DWLPA and DWLPB PCI Adapter Technical Manual

Order Number: EK–DWLPX–TM. A01

DWLPA and DWLPB are PCI adapters (PCIA) developed for the AlphaServer I/O subsystems. The adapters include a PCI to EISA bridge that generates an EISA bus from the PCIA. The PCIA connects to TLSB based systems through an I/O port module.

digital equipment corporation
maynard, massachusetts

First Printing, July 1996

Contents

Preface ix

Chapter 1 Overview

1.1 Applicable Documents 1-1
1.2 Features 1-1
1.3 TLSB I/O General Description 1-2
1.4 PCIA General Description 1-3
1.5 Communication Overview 1-5
1.5.1 DMA Transactions 1-5
1.5.2 Mailbox Transactions 1-6
1.5.3 CSR Transactions 1-6
1.5.4 Interrupt Transactions 1-7

Chapter 2 Addressing

2.1 TLSB System Addressing 2-1
2.2 System Bus to PCI Addressing 2-2
2.2.1 PCI Memory Space 2-4
2.2.1.1 Dense Mapping 2-4
2.2.1.2 Sparse Mapping 2-5
2.2.2 PCI I/O Space 2-8
2.2.3 PCI Configuration Space 2-8
2.2.3.1 PCIA CSR and Map RAM Space 2-12
2.2.3.2 EISA Subtractive Decode Window 2-13
2.2.3.3 PCIA Reserved Addresses 2-13
2.3 PCI to System Bus Addressing 2-13
2.3.1 PC Compatibility Holes 2-14
2.3.2 DMA Address Windows 2-15

Chapter 3 Registers

SCYCLE — PCIA Special Cycle Register 3-5
IACK — PCIA Interrupt Acknowledge Register 3-6
PRESENT — PCIA Slot Present Register 3-7
TBIT — PCIA TBIT Register 3-9
MCTL — PCIA Module Control Register 3-11
IBR — PCIA Information Base Repair Register 3-12
CTLx — PCIx Bus Control Register 3-13
MRETRYx — PCIx Master Retry Limit Register 3-16
GPRx — PCIx General Purpose Register 3-17

ERRx — PCIx Error Summary Register 3-18
FADRx — PCIx Failing Address Register 3-21
IMASKx — PCIx Interrupt Mask Register 3-22
DIAGx — PCIx Diagnostic Register 3-25
IPENDx — PCIx Interrupt Pending Register 3-27
IPROGx — PCIx Interrupt in Progress Register 3-29
WMASK_xx — PCIx Window Mask Registers 3-31
WBASE_xx — PCIx Window Base Registers 3-32
TBASE_xx — PCIx Translated Base Registers 3-33
ERRVEC - PCIx Error Interrupt Vector Registers 3-34
Scatter/Gather RAM Map Entry Format 3-36

Chapter 4 Functional Description

4.1 PCI Interface 4-1
4.1.1 CPU-Initiated PCI Cycles 4-1
4.1.1.1 CPU-Initiated PCI Commands 4-1
4.1.1.2 PCI Addressing 4-2
4.1.1.3 CPU Command Decode 4-4
4.1.1.4 CPU Request Queue 4-4
4.1.1.5 Requesting PCI Bus 4-4
4.1.1.6 Default Bus Parking 4-5
4.1.1.7 Address Phase 4-5
4.1.1.8 Data Phase 4-5
4.1.1.9 Transaction Termination 4-6
4.1.1.10 Exclusive Access Support 4-7
4.1.2 Device-Initiated PCI Cycles 4-7
4.1.2.1 Device-Initiated PCI Commands 4-7
4.1.2.2 PCI Memory Read Multiple Command 4-8
4.1.2.3 PCI Address Decode 4-9
4.1.2.4 PCI Address Translation 4-9
4.1.2.5 Address Queuing 4-10
4.1.2.6 Address Phase 4-10
4.1.2.7 Address Burst Order 4-11
4.1.2.8 Memory Block Boundary 4-11
4.1.2.9 Address Parity Errors 4-11
4.1.2.10 Write Data Transfers 4-11
4.1.2.11 Read Data Transfers 4-12
4.1.2.12 PCI Target Latency Timeout 4-13
4.1.2.13 Device-Initiated Exclusive Access 4-13
4.2 Hose Interface 4-14
4.2.1 Down Hose 4-14
4.2.1.1 Down Hose Commands 4-14
4.2.1.2 Down Hose to PCI Synchronization 4-14
4.2.1.3 Down Hose Signals 4-15
4.2.2 Up Hose 4-15
4.2.2.1 Up Hose Commands 4-15
4.2.2.2 Up Hose Arbitration 4-17
4.2.2.3 Up Hose Command FIFO 4-17
4.2.2.4 Up Hose Signals 4-18
4.2.2.5 Map RAM Structure 4-18
4.2.2.6 Special Programming Considerations for the Up Hose 4-19
4.2.3 Serial EEPROM 4-20
4.2.4 PCI to EISA Bridge Hardware 4-20
4.2.5 CPU-Initiated Transactions 4-20

4.2.6 DMA Transactions ... 4-21
4.3 PCIA Interrupts ... 4-23
4.3.1 PCI Interrupts ... 4-23
4.3.1.1 PCI Device Interrupts ... 4-23
4.3.1.2 PCI Error Interrupts ... 4-23
4.3.2 Hose Interrupt Support ... 4-23
4.3.3 Vectored Interrupts ... 4-24
4.3.3.1 PCI Device Interrupts ... 4-24
4.3.3.2 PCI Error Interrupts ... 4-25
4.3.3.3 Masking PCI Interrupts ... 4-26
4.3.3.4 EISA Interrupts ... 4-26
4.3.4 Diagnostic Features ... 4-27
4.3.4.1 Interrupt Pending Registers ... 4-27
4.3.4.2 Interrupt in Progress Registers ... 4-27

Chapter 5 Error Handling

5.1 Error Categories ... 5-1
5.2 Down Hose Header Cycle Parity Error ... 5-1
5.3 Mailbox Errors ... 5-2
5.4 CSR Command Errors ... 5-2
5.5 DMA Errors ... 5-2
5.6 PCI Errors ... 5-4
5.7 PCIA Errors ... 5-5
5.8 Up Hose Packet Count Error ... 5-6
5.9 PCI Peer-to-Peer Errors ... 5-6
5.10 PCI Configuration Errors ... 5-6

Chapter 6 PCI to EISA Bridge

6.1 PCI and EISA Subsystem Block Diagram ... 6-1
6.2 PCI-EISA Bridge Chipset ... 6-2
6.3 Address Mapping ... 6-3
6.3.1 PCI to EISA ... 6-3
6.3.2 EISA to PCI ... 6-4
6.4 Bridge Buffers ... 6-4
6.5 Interrupts ... 6-4
6.6 EISA Bus Lock ... 6-5
6.7 Lockout Prevention ... 6-6
6.8 Configuration Rules ... 6-6
6.9 EISA Bridge Registers ... 6-7

Appendix A PCIA Supported Hose Packets

A.1 Down Hose Packets ... A-1
A.2 Up Hose Packets ... A-8

Figures

1-1 I/O System Block Diagram ... 1-2
1-2 PCIA Module Physical Layout ... 1-3
1-3 PCIA Module Block Diagram ... 1-5

2-1 CPU View of PCI Addresses ... 2-3
2-2 CPU to PCI Address Mapping ... 2-4
2-3 Dense Space Address Mapping ... 2-5
2-4 Sparse Space Address Mapping (Lower 16 Mbytes) ... 2-7
2-5 Sparse Space Address Mapping (Upper 112 Mbytes) ... 2-7
2-6 PCI Configuration Space Addressing - Type 0 Addresses ... 2-9
2-7 PCI Configuration Space - Type 1 Address Fields ... 2-12
2-8 PCI Device View of System Address Space ... 2-14
2-9 PCI Memory Space, Including Compatibility Holes ... 2-15
2-10 Direct DMA Address Translation ... 2-17
2-11 Mapped DMA Address Translation for DWLPA and DWLPB with TBIT=0 ... 2-17
2-12 Mapped DMA Address Translation for DWLPB with TBIT=1 ... 2-18
4-1 INTR/IDENT Packet ... 4-23
4-2 Interrrupt Status Return Packet ... 4-24
6-1 PCI and EISA Subsystem ... 6-2
6-2 PCI-EISA Bridge Chip Function ... 6-3
A-1 Interrupt Status Return Packet ... A-1
A-2 DMA Read Data Return Packet ... A-2
A-3 Mailbox Command Packet ... A-3
A-4 CSR Sparse Read Packet ... A-4
A-5 CSR Sparse Write Packet ... A-5
A-6 CSR Dense Read Packet ... A-6
A-7 CSR Dense Write Packet ... A-7
A-8 DMA Read Packet ... A-8
A-9 IREAD Packet ... A-8
A-10 DMA Write Mask Packet ... A-9
A-11 DMA Unmasked Write Packet ... A-10
A-12 Mailbox Status Return Packet ... A-11
A-13 Interrupt/IDENT Packet ... A-11
A-14 CSR Sparse Read Return Packet ... A-12
A-15 CSR Dense Read Return Packet ... A-13
A-16 CSR Write Status Return Packet ... A-13

Tables

1 AlphaServer 8200 and 8400 Documentation ... x
2 Related Documents ... xii
2-1 TLSB Address Map ... 2-1
2-2 I/O Node Space Assignment ... 2-2
2-3 CPU to PCI Address Mapping for a Single PCIA ... 2-3
2-4 Sparse Access Encoding - PCI Memory Space ... 2-6
2-5 Sparse Access Encoding - PCI I/O Space ... 2-8
2-6 PCI Configuration Space - Type 0 Address Fields ... 2-10
2-7 Type 0 Configuration Space Address Map ... 2-11
2-8 PCI Configuration Space Addressing - Type 1 Address Bits ... 2-12
2-9 PCIA CSR and Map RAM Address Space ... 2-13
2-10 PCI Window Mask Values ... 2-15
3-1 PCIA Registers ... 3-2
3-2 Register Bit Types ... 3-4
3-3 SCYCLE Register Bit Definitions ... 3-5
3-4 IACK Register Bit Definitions ... 3-6
3-5 PRESENT Register Bit Definitions ... 3-8
3-6 TBIT Register Bit Definitions ... 3-9
3-7 MCTL Register Bit Definitions ... 3-11
3-8 IBR Register Bit Definitions ... 3-12

3-9 CTLx Register Bit Definitions 3-13
3-10 MRETRYx Register Bit Definitions 3-16
3-11 GPRx Register Bit Definitions 3-17
3-12 ERRx Register Bit Definitions 3-19
3-13 FADRx Register Bit Definitions 3-21
3-14 IMASKx Register Bit Definitions 3-23
3-15 DIAGx Register Bit Definitions 3-25
3-16 IPENDx Register Bit Definitions 3-28
3-17 IPROGx Register Bit Definitions 3-30
3-18 WMASK_xx Register Bit Definitions 3-31
3-19 WBASE_xx Register Bit Definitions 3-32
3-20 TBASE_xx Register Bit Definitions 3-33
3-21 ERRVEC Register Bit Definitions 3-34
3-22 PCI Interrupt Vector Offsets 3-35
3-23 Scatter/Gather Map Entry Format 3-36
4-1 Supported PCI Commands 4-1
4-2 CPU Address to PCI Address Conversion 4-3
4-3 Supported PCI Target Commands 4-7
4-4 Down Hose Commands 4-14
4-5 Down Hose Signals 4-15
4-6 Up Hose Commands 4-15
4-7 Up Hose Packet Status 4-16
4-8 Command Signal Assertion 4-17
4-9 Up Hose Signals 4-18
4-10 Map RAM Logic Command 4-19
4-11 PCI Interrupt Priority with and without the Standard I/O Module 4-25
4-12 Interrupt Pending Register Bit Definitions 4-27
6-1 PCI to EISA Address Mapping 6-3
6-2 EISA Bridge and Standard I/O Interrupt Connections 6-4
6-3 Standard I/O EISA Interrupt Assignments 6-5
6-4 Slot Usage with Standard I/O Module Installed 6-6

Preface

Intended Audience

This manual describes the hardware of the PCI I/O subsystem in AlphaServer 8200 and 8400 systems. It discusses the operations of the DWLPA and DWLPB PCI adapters, and provides detailed information on the subsystem registers. The manual is intended for technical professionals such as operating system programmers and customer service engineers.

Document Structure

This manual has six chapters and one appendix:

- **Chapter 1, Overview,** gives an overview of the PCIA adapter.
- **Chapter 2, Addressing,** describes system to PCI and PCI to system addressing schemes.
- **Chapter 3, Registers,** describes each register in the PCIA.
- **Chapter 4, Functional Description,** discusses CPU and PCI commands and transactions in detail.
- **Chapter 5, Error Handling,** describes errors caused during subsystem operations.
- **Chapter 6, PCI to EISA Bridge,** describes the PCI to EISA bridge.
- **Appendix A, PCIA Supported Hose Packets,** shows the supported Up and Down Hose data packets.

Documentation Titles

Table 1 lists the books in the Digital AlphaServer 8200 and 8400 documentation set. Table 2 lists other documents that you may find useful.

Table 1 AlphaServer 8200 and 8400 Documentation

Title	Order Number
Hardware User Information and Installation	
Operations Manual	EK–T8030–OP
Site Preparation Guide	EK–T8030–SP
AlphaServer 8200 Installation Guide	EK–T8230–IN
AlphaServer 8400 Installation Guide	EK–T8430–IN
Service Information Kit	QZ–00RAC–GC
Service Manual (hard copy)	EK–T8030–SV
Service Manual (diskettes)	AK–QKNFA–CA AK–QUW7A–CA AK–QUW6A–CA
Reference Manuals	
System Technical Manual	EK–T8030–TM
System Technical Manual Supplement	EK–T8030–TS
DWLPA/DWLPB PCI Adapter Technical Manual	EK–DWLPX–TM
Upgrade Manuals for Both Systems	
KN7CC CPU Installation Card	EK–KN7CC–IN
KN7CD CPU Installation Card	EK–KN7CD–IN
KN7CE CPU Installation Card	EK–KN7CE–IN
MS7CC Memory Installation Card	EK–MS7CC–IN
KFTHA System I/O Module Installation Guide	EK–KFTHA–IN
KFTIA Integrated I/O Module Installation Guide	EK–KFTIA–IN

Table 1 AlphaServer 8200 and 8400 Documentation (Continued)

Title	Order Number
Upgrade Manuals: 8400 System Only	
AlphaServer 8400 Upgrade Manual	EK–T8430–UI
BA654 DSSI Disk PIU Installation Guide	EK–BA654–IN
BA655 SCSI Disk and Tape PIU Installation Guide	EK–BA655–IN
DWLAA Futurebus+ PIU Installation Guide	EK–DWLAA–IN
DWLMA XMI PIU Installation Guide	EK–DWLMA–IN
DWLPA/DWLPB PCI PIU Installation Guide	EK–DWL84–IN
H7237 Battery PIU Installation Guide	EK–H7237–IN
H7263 Power Regulator Installation Card	EK–H7263–IN
KFMSB Adapter Installation Guide	EK–KFMSB–IN
KZMSA Adapter Installation Guide	EK–KXMSX–IN
RRDCD Installation Guide	EK–RRDRX–IN
Upgrade Manuals: 8200 System Only	
DWLPA/DWLPB PCI Shelf Installation Guide	EK–DWL82–IN
H7266 Power Regulator Installation Card	EK–H7266–IN
H7267 Battery Backup Installation Card	EK–H7267–IN

Table 2 Related Documents

Title	Order Number
General Site Preparation	
Site Environmental Preparation Guide	EK–CSEPG–MA
System I/O Options	
BA350 Modular Storage Shelf Subsystem Configuration Guide	EK–BA350–CG
BA350 Modular Storage Shelf Subsystem User's Guide	EK–BA350–UG
BA350-LA Modular Storage Shelf User's Guide	EK–350LA–UG
CIXCD Interface User Guide	EK–CIXCD–UG
DEC FDDIcontroller 400 Installation/Problem Solving	EK–DEMFA–IP
DEC FDDIcontroller/Futurebus+ Installation Guide	EK–DEFAA–IN
DEC FDDIcontroller/PCI User Information	EK–DEFPA–IN
DEC LANcontroller 400 Installation Guide	EK–DEMNA–IN
DSSI VAXcluster Installation/Troubleshooting Manual	EK–410AA–MG
EtherWORKS Turbo PCI User Information	EK–DE435–OM
KZPSA PCI-to-SCSI Storage Adapter User's Guide	EK–KZPSA–UG
RF Series Integrated Storage Element User Guide	EK–RF72D–UG
StorageWorks RAID Array 200 Subsystem Family Installation and Configuration Guide	EK–SWRA2–IG
StorageWorks RAID Array 200 Subsystem Family Software User's Guide for OpenVMS AXP	AA–Q6WVA–TE
StorageWorks RAID Array 200 Subsystem Family Software User's Guide for DEC OSF/1	AA–Q6TGA–TE
Operating System Manuals	
Alpha Architecture Reference Manual	EY–L520E–DP
DEC OSF/1 Guide to System Administration	AA–PJU7A–TE
Guide to Installing DEC OSF/1	AA–PS2DE–TE
OpenVMS Alpha Version 6.2 Upgrade and Installation Manual	AA–PV6XC–TE
MEMORY CHANNEL Manuals	
MEMORY CHANNEL User's Guide	EK–PCIRM–UG
MEMORY CHANNEL Service Information	EK–PCIRM–SV

Chapter 1

Overview

The PCI adapter (PCIA) module provides a complete PCI and EISA bus subsystem for systems that implement the TLSB I/O hose interface.

1.1 Applicable Documents

- Intel 82420/82430 PCIset ISA and EISA Bridges Databook (April 1993)
- AlphaServer 8200 and 8400 documentation
- AlphaServer 2100 documentation

1.2 Features

Software Compatibility - The PCIA is architecturally compatible with other CPU-PCI, and PCI-EISA bridges, allowing software drivers to run on the TLSB platforms with no modifications.

Hardware Compatibility - The PCIA supports the TLSB hose protocol. Direct CSR access to the PCI bus is allowed on the TLSB.

Configurability - As seen from the CPU, the three PCI buses on the PCIA share a common I/O address space. Therefore, flexible address space assignments meet varying I/O card address space requirements. There are 12 PCI expansion slots plus one slot for the EISA bridge.

Scatter/Gather Capability - The DWLPA supports a 32K entry scatter/gather address map RAM used to translate PCI memory addresses into TLSB main memory addresses. DWLPB supports a 128K entry map RAM.

Performance - To improve the individual DMA transaction performance, the PCIA implements an on-chip scatter/gather cache and reads of a full host memory block. To improve overall DMA throughput, the PCIA implements three physically separate PCI buses and allows a DMA operation to be pending simultaneously for each bus.

EISA Support - A dedicated slot is used to support a PCI-EISA bus bridge module (KFE70), which provides access to an 8-slot EISA bus. This module is the standard I/O module used on AlphaServer 8200 and 8400 systems. The EISA bridge includes an Intel 8242 keyboard/mouse controller, for operating systems that require an integrated keyboard. The module also provides a PC-style floppy disk port and PCI-based Ethernet.

Self-Test - The PCIA does not implement an on-board self-test. However, the host CPU diagnoses the module.

Cache Coherency - The PCIA does not support any of the PCI cache coherency mechanisms required to maintain coherency between TLSB main memory and memory located on the PCI.

Memory Locks - Since the TLSB system buses do not provide locks, only a single, limited case of PCI locking is supported.

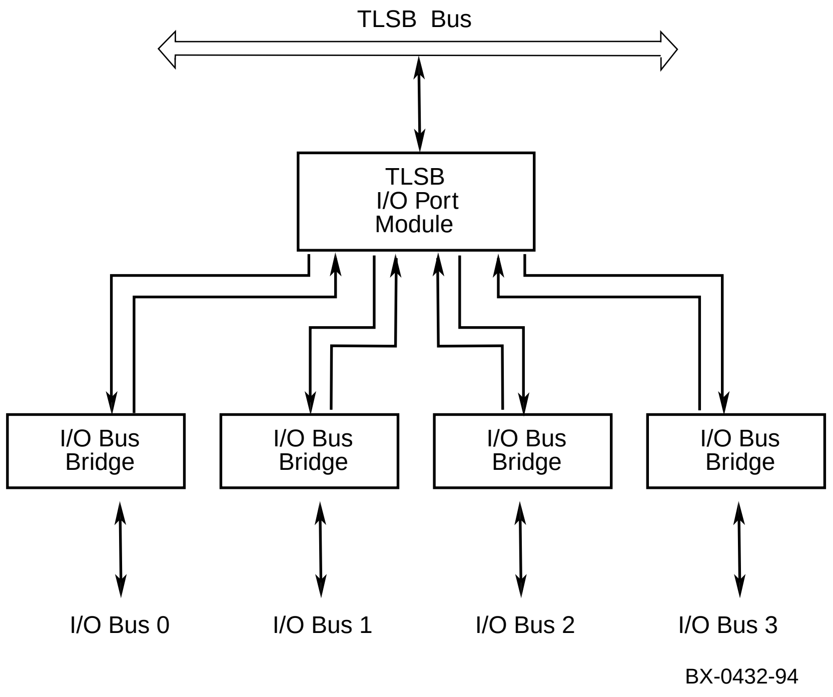
PCI Addressing - The PCIA does not support the 64-bit addressing mode or the 64-bit data path of the PCI bus.

1.3 TLSB I/O General Description

TLSB systems consist of processor modules, memory modules, and I/O port modules, as shown in Figure 1-1. Each I/O module (KFTIA/KFTHA) supports from 1 to 4 bridges to a remote I/O bus such as XMI, Futurebus+, or PCI. The bus bridge, which includes the backplane for the PCI/EISA devices, is housed in a plug-in unit (PIU) in the 8400 or in a PCI shelf in the 8200. The bridge is connected to the I/O port module by a pair of cables called a hose. One cable of the pair carries data and control to the bridge (the Down Hose), and the other carries traffic from the bridge (the Up Hose).

The I/O module converts between transactions on the system bus and hose data packets as interpreted by the I/O bridges. The remote bus address spaces are made visible to the CPU through either I/O mailboxes (XMI and Futurebus+) or as a direct CSR space (PCI). The I/O module does not attempt to maintain coherency between system memory and any memory on its I/O adapters.

Figure 1-1 I/O System Block Diagram



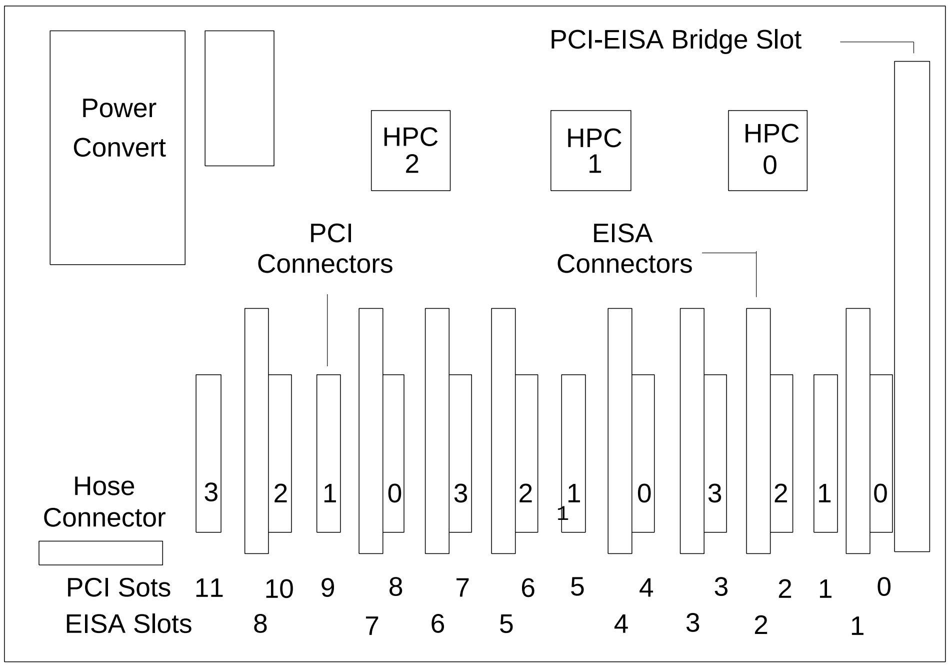
1.4 PCIA General Description

The PCI bus bridge is located on the PCIA. From the CPU viewpoint, the PCI side of the bridge appears as a single, logical PCI bus. The bus supports up to 15 PCI devices. Some of these devices can be placed in the 12 PCI expansion slots on the PCIA motherboard. Others are embedded PCI options on the PCI-EISA bridge module (also called the standard I/O module). From the CPU, the logical PCI appears to have one set of address spaces.

For electrical and performance reasons, the logical PCI is implemented as three physical PCI buses. Adjacent groups of four devices are physically on the same PCI segment and can communicate as peers. The 12 PCI expansion slots are divided evenly, with 4 slots on each physical bus. Each bus operates at 33 MHz and supports a 32-bit data path and 32-bit addresses, for a raw bus bandwidth of 120 Mbytes/sec. Figure 1-2 shows the physical layout of the PCIA module.

The PCI-EISA bridge (standard I/O module) occupies a special slot on one of the physical buses and when installed all of the slots of that physical segment are dedicated to the EISA bus. Eight of the expansion slots include two sets of connectors and can be used for either PCI or EISA modules. The bridge is implemented using the Intel PCEB/ESC (Mercury) chipset and provides a single 33 Mbyte/sec EISA bus using the 8 slots with EISA connectors.

Figure 1-2 PCIA Module Physical Layout



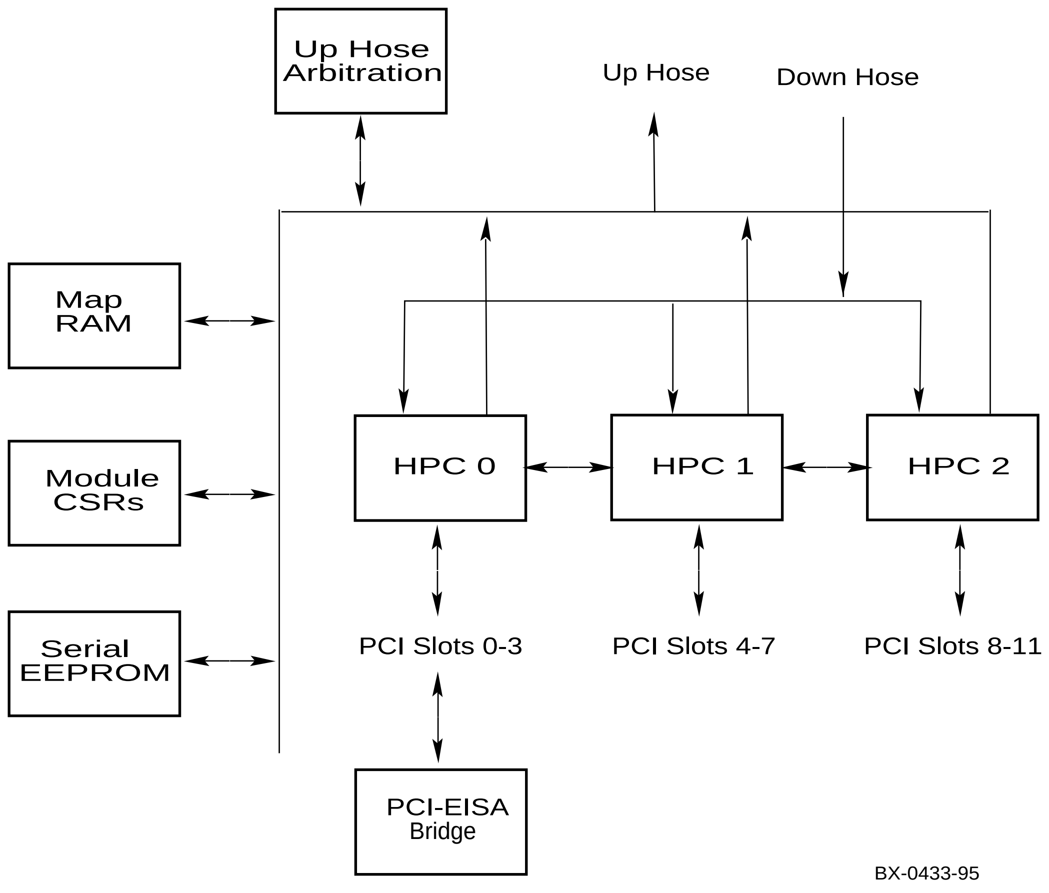
The PCIA provides access to all three PCI address spaces: memory space, I/O space, and configuration space. All are accessible using sparse address mapping that allows for byte access. PCI memory space is also accessible by dense mapping that supports cache block sized bursts.

From the system bus side of the PCIA, a single logical PCI with one set of address spaces is seen. This logical view exists only from the CPU. From the PCI side of the PCIA, three physical PCI bus segments exist. Individual PCI bus masters see only the address spaces local to their physical PCI segment. PCI devices on the same physical bus communicate as peers without involving the PCIA. All three segments normally are configured to see the same view of system memory.

Each of the three physical PCI bus segments is controlled by a PCI bus interface gate array (HPC). Down Hose data from the KFTIA/KFTHA is sent to each of the HPCs in parallel. Up Hose data is driven by each HPC over a tristate bus, UP_BUS<31:0>, to a driver which transmits the data on to the Up Hose. Each HPC arbitrates for the use of the UP_BUS when sending packets over the Up Hose. The UP_BUS also provides access to the scatter/gather map RAM and some module-level control registers. Each HPC has 16 PCI interrupt inputs, which it prioritizes and sends to the KFTIA/KFTHA. Each HPC monitors UP_BUS transactions to maintain a consistent count of the number of transactions that the PCIA has outstanding. Figure 1-3 shows the PCIA module block diagram.

The EISA bus bridge provides decoding for an 8-bit XBUS, which supports PC-style floppy, keyboard, and mouse ports on an additional connector module. The module also includes an Ethernet port. Support for these peripherals is operating system dependent. Refer to the *AlphaServer 2100 and 2200 I/O Specification* for additional information.

Figure 1-3 PCIA Module Block Diagram



The PCIA motherboard and EISA bridge modules, the standard I/O module and the connector module, mount in an enclosure that provides the card cage for the PCI and EISA option modules. The enclosure is the same form factor as a StorageWorks shelf and mounts in a modified version of a plug-in unit or a rackmount shelf.

1.5 Communication Overview

The PCIA hardware communicates with the I/O port using four types of transactions: DMA, mailbox, CSR, and interrupt.

1.5.1 DMA Transactions

All DMA transactions access the PCIA as a PCI target. PCI memory transactions are forwarded to the I/O port if they access one of the address ranges specified by a set of DMA window registers. If required, the PCI DMA address is translated to a system memory address by scatter/gather address mapping. The PCIA can generate DMA read and masked/unmasked write transactions to the I/O port. The PCIA responds to multiple PCI memory read transactions by prefetching a programmable number of cache blocks. The PCIA generates an interlocked read transaction to the I/O port in response to a PCI DMA read with PCI lock asserted.

DMA write transactions are received from the PCI in a store and forward manner. The PCIA generates masked octaword or hexword writes, or unmasked double hexword writes to the I/O port. Each HPC contains two PCI DMA write buffers allowing one transaction's write data to be transferred over the Up Hose while another PCI DMA command is being received from the PCI bus. All DMA read transactions are the size of the system memory block. Memory block sizes of 32 and 64 bytes are supported. DMA read commands sent to the I/O port are tagged with an HPC ID code. Read return data received over the Down Hose bus is identified by its tag and buffered by the appropriate HPC. Each HPC contains one 64-byte read return buffer. The buffer is filled from the hose until a cut-through threshold is reached, at which point the HPC begins transferring data on the PCI. The PCI transfer proceeds in parallel with the remainder of the hose transfer.

1.5.2 Mailbox Transactions

All mailbox transactions are executed by the PCIA hardware as a PCI master. Mailbox transactions forwarded from the I/O port can access a PCI I/O card on the PCI bus, a PCIA CSR, or the map RAM. Mailbox transactions to PCI memory or I/O space on the PCI bus are sent to all three PCI buses, but only one bus responds to the transaction.

Mailbox transactions are byte, word, tribyte, longword, or quadword in length. A Mailbox Command packet is received by the PCIA on the Down Hose. Each of the HPCs decodes the command although only one will execute it. After executing the mailbox read or write command, the executing HPC returns a Mailbox Status packet to the I/O port over the Up Hose. Data is included in the Up Hose packet if the command was a mailbox read.

Mailbox transactions are used for diagnostic and initialization purposes. Although the PCIA can buffer up to four mailbox transactions, it can execute only one transaction at a time. The I/O port sends one mailbox transaction at a time to the PCIA.

1.5.3 CSR Transactions

All CSR commands are issued by the CPU and the transactions are executed by the PCIA as a PCI master. CSR transactions can access a location on the PCI bus, a PCIA CSR, or the map RAM. CSR transactions to PCI memory or I/O space on the PCI bus are sent to all three PCI buses, with only one bus actually responding to the transaction.

CSR transactions are byte, word, tribyte, longword, quadword, or hexword in length. Byte, word, tribyte, longword, and quadword transfers are supported through a sparse CPU-to-PCI address mapping. Hexword transfers are supported through dense CPU-to-PCI address mapping. Byte, word, tribyte, and longword transfers result in a PCI cycle burst length of one. Quadword transfers result in a PCI cycle burst length of two. Hexword transfers result in a PCI cycle burst length of eight. See Chapter 2 for a description of PCI sparse and dense address spaces.

Each HPC decodes the CSR command although only one HPC executes it. After executing the CSR read or write command, the HPC returns a CSR status packet to the TIOP over the Up Hose. Data is included in the Up Hose packet if the command was a CSR read.

The DWLPA can buffer four CSR transactions while the DWPLB can buffer six CSR transactions. The buffering reduces the I/O port wait time between the end of one transaction on the PCI bus and the start of the next transaction on the PCI bus.

1.5.4 Interrupt Transactions

Interrupts are handled by all three of the physical PCI buses. Each HPC accepts 16 PCI device interrupt inputs from the bus it interfaces to. Each HPC has 17 programmable interrupt vector registers. Sixteen of the registers hold hardware device interrupt vectors, and one register holds an error interrupt vector.

The interrupts are latched and prioritized in the HPC. The HPC generates an INTR/IDENT by accessing the CSR that contains the selected interrupt's vector and merging it with a programmable device IPL. The INTR/IDENT is then sent to the I/O port over the Up Hose.

All PCI device interrupts are issued as INTR/IDENTs at the same interrupt priority level. No prioritization is provided between interrupts generated on different PCI buses. Normal Up Hose arbitration is used to select the order in which HPCs issue INTR/IDENTS to the Up Hose. All HPCs monitor the outstanding INTR/IDENTs and inhibit issuing INTR/IDENTs at the outstanding IPLs until an interrupt status packet for that IPL is returned on the Down Hose.

Chapter 2

Addressing

2.1 TLSB System Addressing

The AlphaServer 8200/8400 system bus (TLSB) supports up to 1 terabyte (40 bits) of physical address space. All system memory resides on the TLSB in the lower 512 gigabytes. The system provides direct I/O space access through the upper 512 gigabytes. Up to three KFTHA I/O modules are supported, although TLSB systems can be configured with as many as five KFTHAs. I/O space is divided among the five TLSB nodes that can contain I/O port modules. Within each node I/O space, the address space is divided among the four I/O port hoses. The various address spaces created by this scheme are known as window spaces. Table 2-1 gives the map of TLSB address space. Table 2-2 shows how each node I/O address space is assigned to four hoses. The System Bus Address portion of Figure 2-1 shows the window space.

Table 2-1 TLSB Address Map

TLSB ADDR<39:0>	Description
00 0000 0000 - 7F FFFF FFFF	TLSB memory space
80 0000 0000 - 8F FFFF FFFF	I/O space for node 4
90 0000 0000 - 9F FFFF FFFF	I/O space for node 5
A0 0000 0000 - AF FFFF FFFF	I/O space for node 6
B0 0000 0000 - BF FFFF FFFF	I/O space for node 7
C0 0000 0000 - CF FFFF FFFF	I/O space for node 8
D0 0000 0000 - DF FFFF FFFF	Reserved
E0 0000 0000 - EF FFFF FFFF	Reserved
F0 0000 0000 - FF FFFF FFFF	TLSB CSR space/CPU private space

Table 2-2 I/O Node Space Assignment

Address Range	Assignment
x0 0000 0000 - x3 FFFF FFFF	I/O space for hose 0
x4 0000 0000 - x7 FFFF FFFF	I/O space for hose 1
x8 0000 0000 - xB FFFF FFFF	I/O space for hose 2
xC 0000 0000 - xF FFFF FFFF	I/O space for hose 3

2.2 System Bus to PCI Addressing

A PCIA consumes 16 Gbytes (34 bits) of system I/O address space, which it uses to provide four mappings of a single 4-Gbyte PCI address space. Multiple mappings are required because the PCI uses multiple transaction types and sizes which cannot be directly expressed by an Alpha CPU. The cycle types and transaction sizes are encoded in the I/O space addresses generated by the CPU.

Although the PCIA supports three electrically distinct PCI bus segments, it presents a single logical PCI address space. A device can be configured to a given PCI address, independent of which physical PCI bus it is connected to. This provides for a large number of PCI slots, without splitting the available system I/O address space into regions too small to provide useful mappings. Table 2-3 shows CPU to PCI address mapping for one PCIA. The remainder of the chapter provides the details of each mapping.

It is important to note that this section describes the view of a PCI address space *as seen from the CPU*. The view seen by a PCI device is different and is described in Section 2.4. Figure 2-1 shows the view of PCI addresses, as seen by the CPU.

The responsibility for mapping system (CPU) addresses to PCI addresses is shared by the PCIA and the I/O module. This chapter describes the system mapping on AlphaServer 8200/8400 systems. Refer to Chapter 5 and Appendix A for information on how system addresses are encoded in the hose protocol.

Figure 2-1 CPU View of PCI Addresses

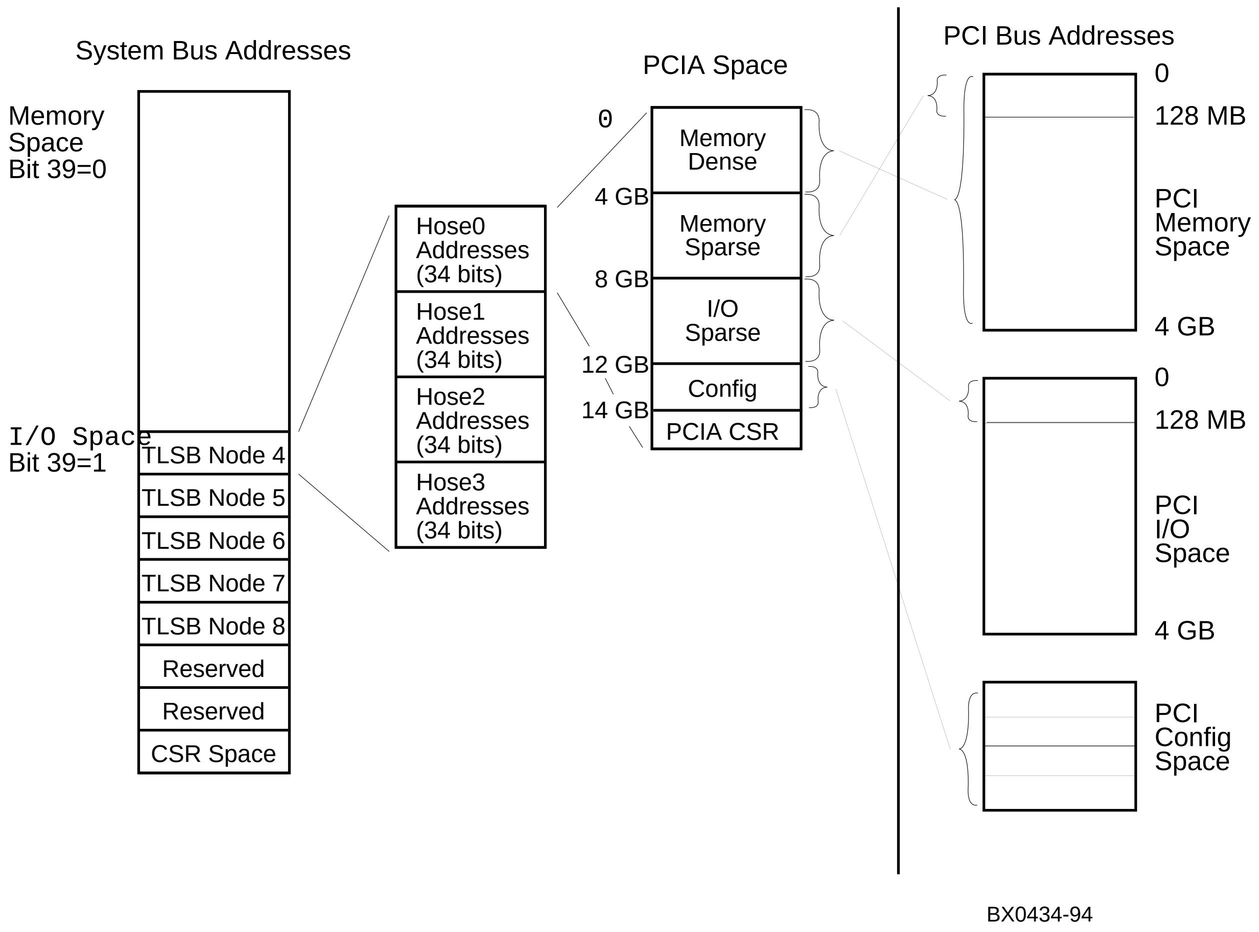


Table 2-3 CPU to PCI Address Mapping for a Single PCIA

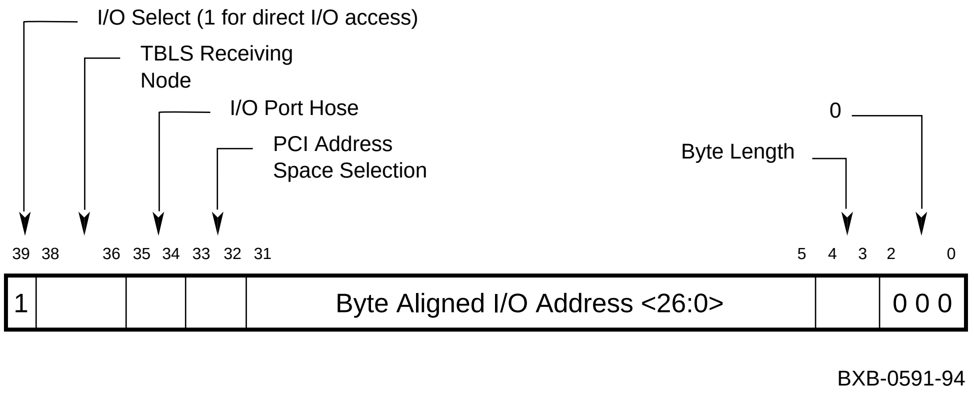
CPU Address Range	PCI Address Range	PCI Address Space
x0 0000 0000 - x0 FFFF FFFF	0000 0000 - FFFF FFFF	PCI memory space - Dense mapping
x1 0000 0000 - x1 1FFF FFFF	0000 0000 - 00FF FFFF	PCI memory space - Sparse mapping (fixed)
x1 2000 0000 - x1 FFFF FFFF	0100 0000 - 07FF FFFF [1]	PCI memory space - Sparse mapping (variable)
x2 0000 0000 - x2 1FFF FFFF	0000 0000 - 00FF FFFF	PCI I/O space - Sparse mapping (fixed)
x2 2000 0000 - x2 FFFF FFFF	0100 0000 - 07FF FFFF [1]	PCI I/O space - Sparse mapping (variable)
x3 0000 0000 - x3 7FFF FFFF	0000 0000 - 0000 00FF [2]	PCI configuration space
x3 8000 0000 - x3 FFFF FFFF	N/A	PCIA CSRs and Map RAM

[1] Variable under the control of the HAE fields of the CTLx register. Shown for HAEs = 0.

[2] For each possible PCI device.

Figure 2-2 shows another view of the CPU to PCI mapping, indicating how the address bits are decoded. Note this is a "software" view of the decoding. It ignores some details of how the low-order address bits are actually passed from the CPU over the system bus to the I/O port.

Figure 2-2 CPU to PCI Address Mapping



I/O Select (1 for direct I/O access)
TBLS Receiving Node
I/O Port Hose
PCI Address Space Selection
0
Byte Length
39 38 36 35 34 33 32 31
5 4 3 2 0
1
Byte Aligned I/O Address <26:0>
0 0 0
BXB-0591-94

2.2.1 PCI Memory Space

2.2.1.1 Dense Mapping

Accesses to the first 4 Gbytes of the PCIA address space are mapped one-to-one to the combined PCI memory space. Software can reference only aligned, longword, and quadword PCI data through the dense mapping, since there is no way to encode byte, word, or tribyte references.

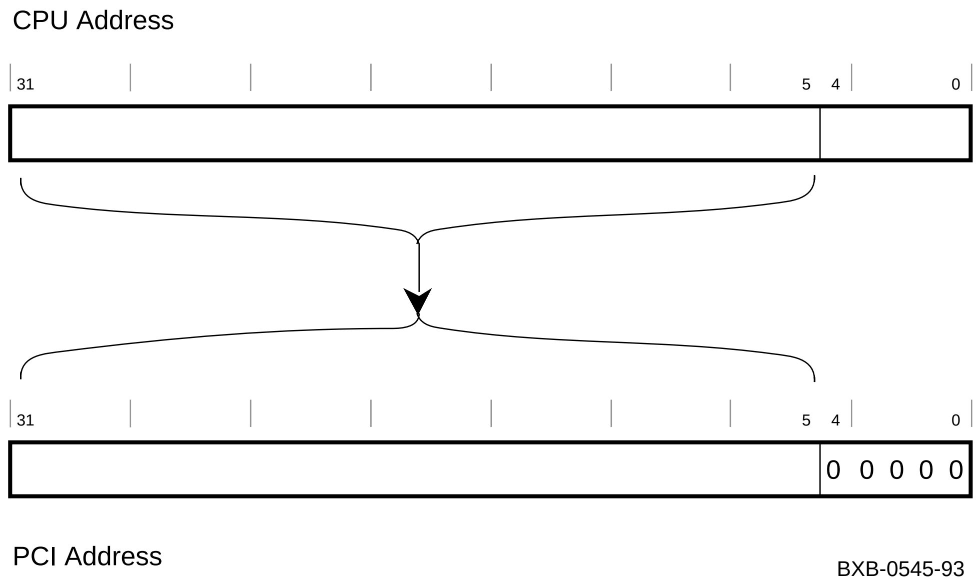
On AlphaServer 8200 and 8400 systems, the KFTIA/KFTHA transfers 32-byte blocks of data to the PCIA. The blocks result from merging of CSR writes in the CPU's write buffers (32-byte blocks). Each block of write data is accompanied by a mask that indicates which longwords in the block contain valid data.

The PCIA transfers a block by generating a burst on the PCI, referencing each longword within the block and transferring data for the longwords indicated by the mask (enables asserted). Unmasked longwords are written in a cycle with all PCI byte enables deasserted. The following software restriction is therefore forced: software must not reference a PCI memory location through dense space unless all locations of the 32-byte block containing the data exist and can be accessed without side effects.

For CPU reads directed to the PCI, the PCIA generates a burst on the PCI for a 32-byte block. Again, this forces the restriction that all locations of the 32-byte block containing the data exist and are readable without side effects. Although a 32-byte block is read and returned over the system bus, the CPU determines how much data is used. For AlphaServer 8200/8400, the read-merge buffer limits the maximum usable data to two quadwords.

For all dense space transfers, PCI addresses are generated as shown in Figure 2-3.

Figure 2-3 Dense Space Address Mapping



2.2.1.2 Sparse Mapping

Accesses to the second 4 Gbytes of the PCIA address space are mapped to the PCI using a sparse addressing scheme. The PCI bus is a longword wide, with four byte enables. Therefore, access to arbitrary groups of bytes within a longword is allowed. The Alpha architecture only provides instructions to access aligned longwords and quadwords. Sparse mapping allows byte, word, tribyte, longword, and quadword accesses to be performed on the PCI, by encoding the size and position of an access in the low-order bits of the CPU address. The PCIA supports the encoding shown in Table 2-4, which is sufficient to produce all the PCI references.

Table 2-4 Sparse Access Encoding - PCI Memory Space

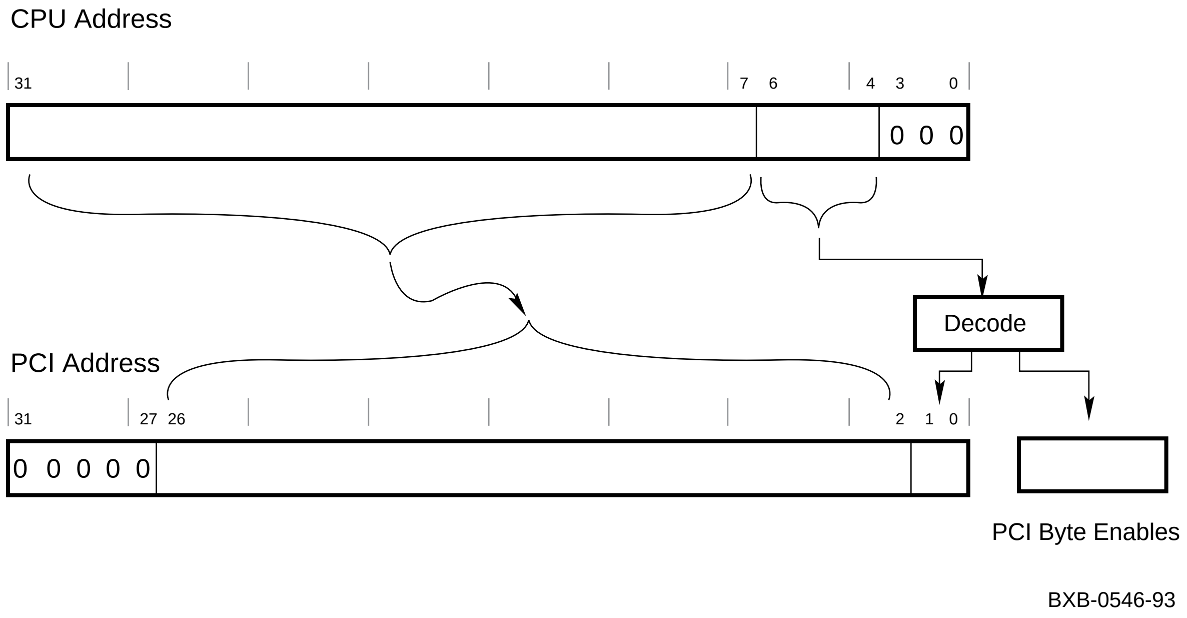
CPU A<6:5>	CPU A<4:3>	Transfer Size	PCIA <1:0>	PCI Byte Enables (C/BE)
00	00	Byte	00	1110 (0 = enabled)
01	00		00	1101
10	00		00	1011
11	00		00	0111
00	01	Word	00	1100
01	01		00	1001
10	01		00	0011
11	01		00	0111 [1]
00	10	Tribyte	00	1000
01	10		00	0001
10	10		00	0011 [1]
11	10		00	0111 [1]
00	11	Longword	00	0000
01	11		00	0001 [1]
10	11		00	0011 [1]
11 [2]	11	Quadword	00	0000 (1st cycle) 0000 (2nd cycle)

[1] Software should consider these combinations of CPU<6:3> as undefined. They are not required to produce the same results on all Alpha systems.

[2] For quadwords, CPU A<7> must be zero.

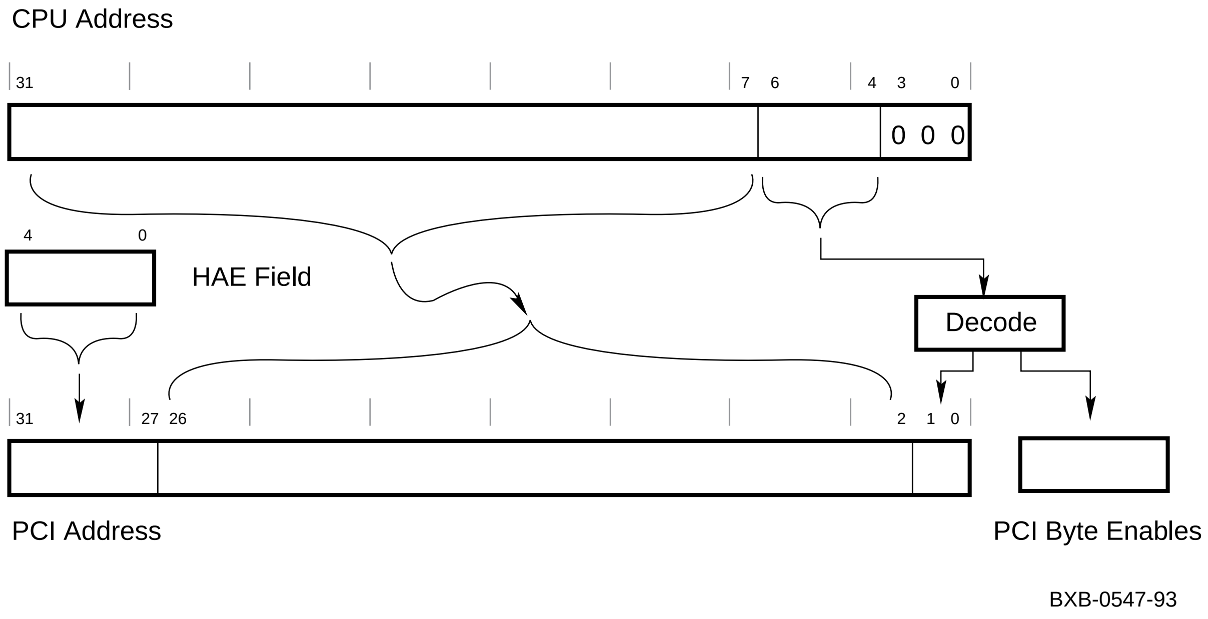
Since five address bits are used by access encoding, a 4-Gbyte range of PCIA address space maps only 128 Mbytes of PCI memory space. The lower 512 Mbytes of CPU sparse address space maps directly to the lower 16 Mbytes of PCI memory space using the translation shown in Figure 2-4.

Figure 2-4 **Sparse Space Address Mapping (Lower 16 Mbytes)**



For the upper 3.5 Gbytes of PCIA space (CPU A<31:29> non-zero), the translation is modified by supplying an additional five high-order PCI address bits obtained from the Memory Hardware Address Extension (MHAE) field in the PCIx Control Register (CTLx). This maps an additional 112 Mbytes of PCI address space, with addresses constructed as shown in Figure 2-5.

Figure 2-5 **Sparse Space Address Mapping (Upper 112 Mbytes)**



To perform a byte, word, or tribyte access to PCI memory space, software must issue an LDL or STL instruction to an address created as described above. The longword quantity is transferred from/to the PCI data lines, without any byte shifting. Software is responsible for positioning the data in the correct byte lanes, corresponding to the byte enables shown in Table 2-4. For quadword transfers, software must issue an LDQ or STQ instruction. The low longword of the quadword will be written to the specified PCI address, and the high longword will be written to the next longword address.

Note that for PCI memory space, the PCI specification requires that PCI A<1:0> encode the address burst order. The PCIA always forces these bits to zero (linear incrementing burst addresses).

2.2.2 PCI I/O Space

Accesses to the third 4 Gbytes of the PCIA address space are sparsely mapped to the PCI I/O space. The mapping is identical to that used for sparse memory space, except for PCI A<1:0>. Table 2-5 shows the mapping for PCI I/O space.

Table 2-5 Sparse Access Encoding - PCI I/O Space

CPU A<6:5>	CPU A<4:3>	Transfer Size	PCIA <1:0>	PCI Byte Enables (C/BE)
00	00	Byte	00	1110 (0 = enabled)
01	00		01	1101
10	00		10	1011
11	00		11	0111
00	01	Word	00	1100
01	01		01	1001
10	01		10	0011
11	01		01[1]	0111 [1]
00	10	Tribyte	00	1000
01	10		01	0001
10	10		10[1]	0011 [1]
11	10		11[1]	0111 [1]
00	11	Longword	00	0000
01	11		01[1]	0001 [1]
10	11		10[1]	0011 [1]
11	11	Quadword	00	0000 (1st cycle PCI A<2> = 0)
				0000 (2nd cycle PCI A<2> = 1)

[1] Software should consider these combinations of CPU<6:3> as undefined. They are not required to produce the same results on all Alpha systems.

For the upper 3.5 Gbytes of PCIA I/O space (CPU A<31:29> non-zero), the I/O Hardware Address Extension (IHAE) field of the PCIx Control Register (CTLx) supplies the upper five bits of the PCI address. All other aspects of sparse I/O space are identical to those of sparse memory space.

2.2.3 PCI Configuration Space

The next 2-Gbyte region of PCIA address space provides access to PCI configuration space. The same sparse mapping used for I/O and memory space is used here, with the following exceptions:

- PCI ADDR<1:0> control the type of configuration cycle being performed. These bits are supplied by bits in the CTLx CSRs.
- PCI ADDR<31:26> are always zero.

Each PCI device has a 256-byte region for its configuration registers. For a PCI device on one of the PCIA's bus segments (that is, not on a bridged bus), the configuration registers are referenced with a Type 0 configuration cycle (PCI ADDR<1:0> = 0), using addresses constructed as shown in Figure 2-6.

Figure 2-6 PCI Configuration Space Addressing - Type 0 Addresses

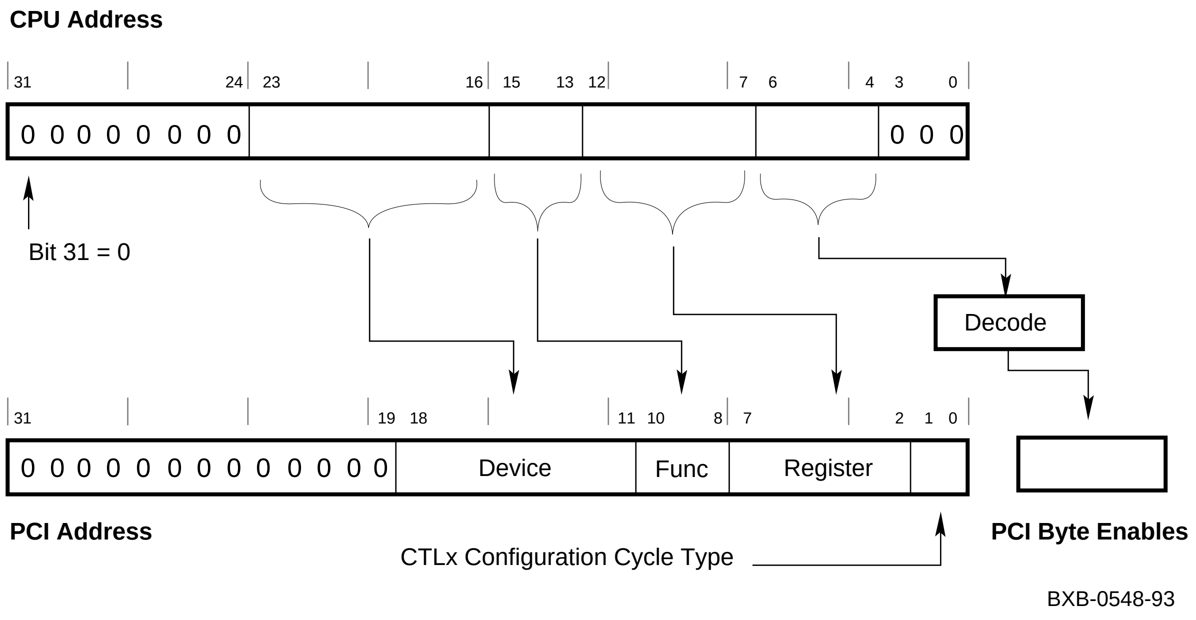


Table 2-6 PCI Configuration Space - Type 0 Address Fields

CPU Address Bits	PCI Address Bits	Definition
	<31:19>	Always zero
<23:16>	<18:11>	PCI device select. Each PCI address line <16:11> is physically tied to one PCI device IDSEL line. PCI<18:17> identify the HPC. 00 - HPC 0 01 - HPC 1 10 - HPC 2 11 - HPC 3 PCI<16:11> identify the IDSEL line.
<15:13>	<10:8>	Device function number
<12:7>	<7:2>	Configuration register
<6:3>		Sparse space encoding of byte enables
<2:0>		Must be zero.
	<1:0>	Configuration cycle type from CTLx register; zero for Type 0 cycles.

PCI device is selected by PCI Address <18:11>. The resulting memory map is shown in Table 2-7. Each HPC can address up to five PCI devices. This capability is used to access the PCI devices located on the standard I/O module, and when the HPC is used in other products.

No slots of PCI bus 0 can contain PCI modules, if the standard I/O module is installed, because these slots share PCI bus signals with PCI devices on the standard I/O module. Those physical backplane positions can still be used for EISA modules, except for the second slot, which holds the standard I/O connector module.

Type 0 configuration cycles are sent only on the physical PCI bus segment containing the selected device.

Table 2-7 Type 0 Configuration Space Address Map

CPU ADDR<31:0>	Destination Addressed	
	Without Standard I/O	With Standard I/O
0000 0000 - 0000 FFFF	Not used	Not used
0001 0000 - 0001 FFFF	Not used	Standard I/O Ethernet port
0002 0000 - 0003 FFFF	Not used	Not used
0004 0000 - 0007 FFFF	PCI Bus 0 Slot 0	Standard I/O EISA bridge chip
0008 0000 - 000F FFFF	PCI Bus 0 Slot 1	Not used
0010 0000 - 001F FFFF	PCI Bus 0 Slot 2	Not used
0020 0000 - 003F FFFF	PCI Bus 0 Slot 3	Not used
0040 0000 - 0040 FFFF	Not used	Not used
0041 0000 - 0041 FFFF	Not used	Not used
0042 0000 - 0043 FFFF	Not used	Not used
0044 0000 - 0047 FFFF	PCI Bus 1 Slot 0	PCI Bus 1 Slot 0
0048 0000 - 004F FFFF	PCI Bus 1 Slot 1	PCI Bus 1 Slot 1
0050 0000 - 005F FFFF	PCI Bus 1 Slot 2	PCI Bus 1 Slot 2
0060 0000 - 007F FFFF	PCI Bus 1 Slot 3	PCI Bus 1 Slot 3
0080 0000 - 0080 FFFF	Not used	Not used
0081 0000 - 0081 FFFF	Not used	Not used
0082 0000 - 0083 FFFF	Not used	Not used
0084 0000 - 0087 FFFF	PCI Bus 2 Slot 0	PCI Bus 2 Slot 0
0088 0000 - 008F FFFF	PCI Bus 2 Slot 1	PCI Bus 2 Slot 1
0090 0000 - 009F FFFF	PCI Bus 2 Slot 2	PCI Bus 2 Slot 2
00A0 0000 - 00BF FFFF	PCI Bus 2 Slot 3	PCI Bus 2 Slot 3
00C1 0000 - 00FF FFFF	Not used	Not used
0100 0000 - 07FF FFFF	Above 4 sub-regions Wrapped 7 more times	Above 4 sub-regions Wrapped 7 more times

Type 1 configuration cycles (PCI ADDR<1:0>=1) are used to access configuration registers on PCIs bridged from the PCIA. Type 1 accesses are not decoded by the PCIA and are sent to all three PCI buses. Addresses for type 1 cycles are translated as shown in Figure 2-7.

Figure 2-7 PCI Configuration Space - Type 1 Address Fields

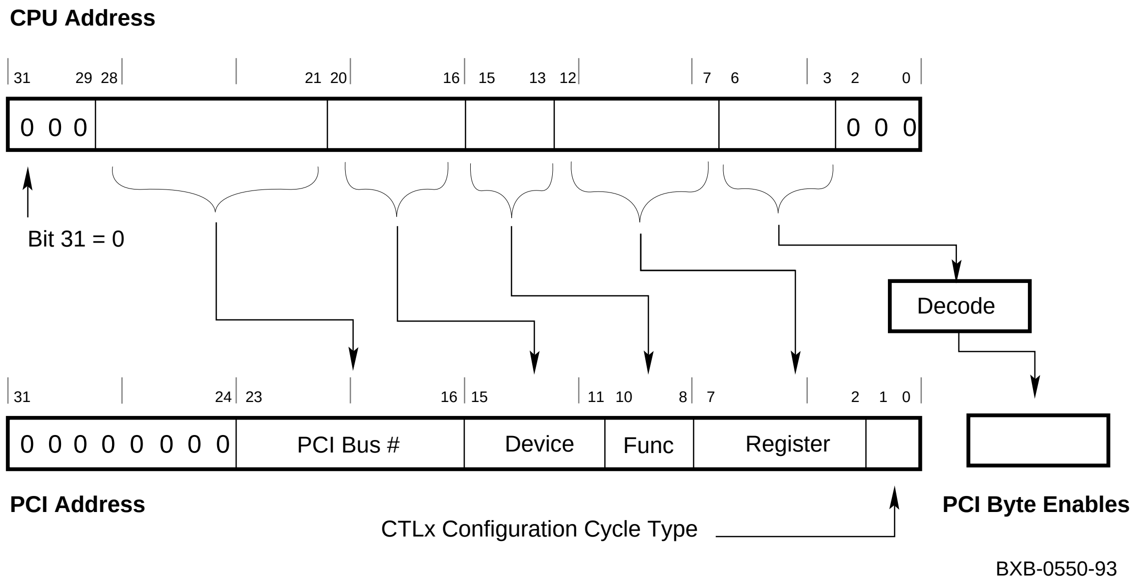


Table 2-8 PCI Configuration Space Addressing - Type 1 Address Bits

CPU Address Bits	PCI Address Bits	Definition
	<31:24>	Always zero
<28:21>	<23:16>	PCI bus number
<20:16>	<15:11>	PCI device select
<15:13>	<10:8>	PCI device function number
<12:7>	<7:2>	Configuration register
<6:3>		Sparse space encoding of byte enables
<2:0>		Must be zero.
	<1:0>	Configuration cycle type from CTLx register - one for Type 1 cycles.

2.2.3.1 PCIA CSR and Map RAM Space

The final 2-Gbyte region of PCIA address space contains the control and status registers for the PCIA and the map RAM used for address translation. References to this region do not result in transactions on the PCI. The PCIA registers reside in each of the HPCs and on the PCIA module. There is one bank of registers for each HPC as shown in Table 2-9. The module registers are addressed through HPC0.

The address translation map RAM consists of 32K longword entries for the DWLPA and 128K longword entries for the DWLPB. These entries are used to perform scatter/gather mapping and to translate 32-bit PCI DMA addresses into 40-bit TLSB system addresses. The CSR and map RAM regions can be read and written using either longword- or quadword-width accesses. Quadword accesses must be aligned on quadword boundaries.

Table 2-9 **PCIA CSR and Map RAM Address Space**

CPU Address<31:0>	Destination Addressed
8000 0000 - 801F FFFF	HPC 0 CSRs
8020 0000 - 803F FFFF	HPC 1 CSRs
8040 0000 - 805F FFFF	HPC 2 CSRs
8060 0000 - 807F FFFF	Reserved. Access causes PCIA Illegal CSR Error
8080 0000 - 80FF FFFF	Module CSRs. Writes to unused addresses ignored, reads return the PRESENT register.
8100 0000 - 81FF FFFF	Map RAM
8200 0000 - 83FF FFFF	Reserved
8400 0000 - FFFF FFFF	Aliased copies of 8000 0000 - 83FF FFFF

2.2.3.2 EISA Subtractive Decode Window

The PCI-EISA bridge uses a technique called subtractive decode to determine which PCI addresses to forward to the EISA bus. The PCIA only supports subtractive decode in the first 8 Mbytes of PCI memory. Therefore, all EISA memory space devices must be configured in the first 8 Mbytes of EISA (and therefore PCI) memory space. When the standard I/O module (KFE70) is installed, no PCI devices may be configured in the first 8-Mbyte segment of PCI memory.

2.2.3.3 PCIA Reserved Addresses

The preceding sections describe various address ranges as Reserved. Also reserved are addresses in PCI memory, I/O, or configuration space that are not occupied by any device on the PCI bus. If these addresses are accessed by a CPU, an error is reported.

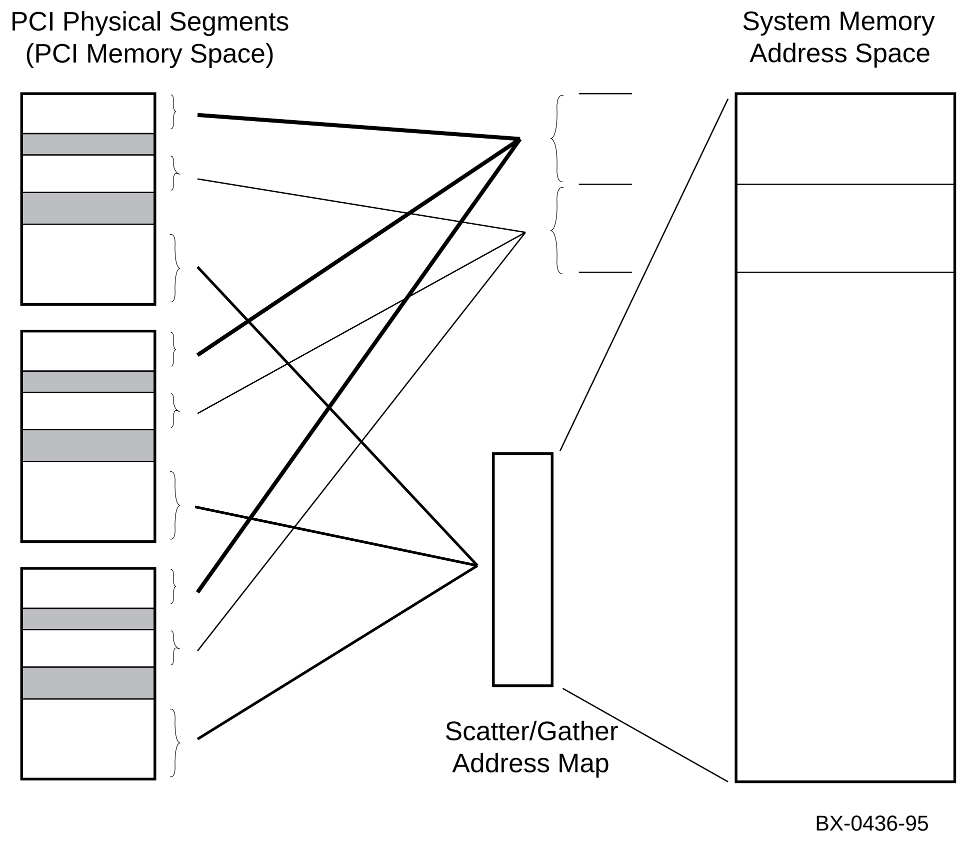
On a read to a nonexistent address, the PCIA will return a hose packet with the error bit set. The result of this is system dependent. On AlphaServer systems, the I/O port responds by writing to the CSR Read Return Error Register in TLSB CSR Broadcast space. This results in a machine check interrupt being delivered to the CPU through vector 0x660. On a write to a nonexistent address, the PCIA reports an interrupt through the PCIA Error Interrupt Vector. Refer to Chapter 5 for details on which HPC determines that an error occurred and what data is captured in the error CSRs.

2.3 PCI to System Bus Addressing

The PCIA serves as a bridge to allow PCI devices to perform direct memory access (DMA) to system memory. A given PCI device sees a single memory address space, which must contain both system memory and the memory space registers of other PCI devices. The PCIA provides three programmable address windows to make system memory addresses visible from the PCI. Memory space references generated by a PCI bus master that fall within these windows are translated and forwarded up the hose to the system memory bus. Memory references that do not hit in the windows are not forwarded, and may be decoded by another PCI target device on the same physical PCI bus segment. The PCIA does not forward memory references from one PCI bus segment to the other two.

Figure 2-8 shows a possible view of the system address space, as seen by a PCI device. The shaded regions are only accessible by devices on the same PCI physical segment. The unshaded regions indicate where PCI memory space accesses are forwarded to system memory space, either directly or through the scatter/gather address map. The exact mapping of PCI to system memory regions is dependent on the programming of the address window registers. The registers should be programmed identically in each HPC, producing the same mapping from each PCI physical segment.

Figure 2-8 PCI Device View of System Address Space

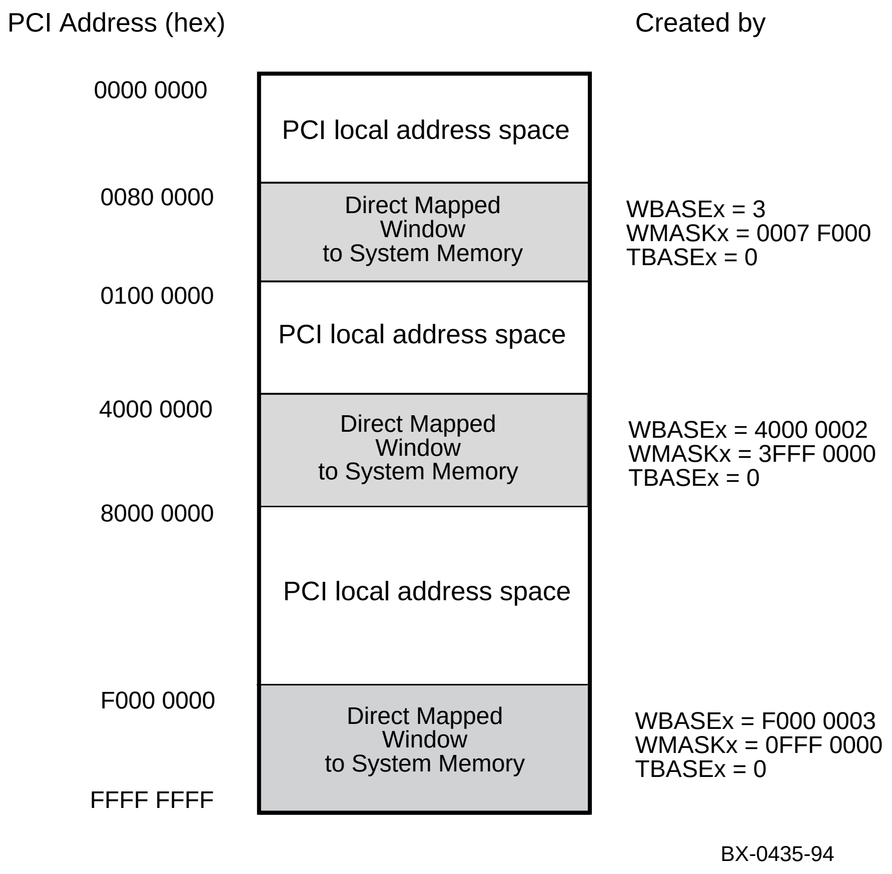


2.3.1 PC Compatibility Holes

Certain EISA devices respond to hardwired memory space addresses. PCI memory references to those addresses must not be forwarded to main memory. These nonforwarding regions are called "PC compatibility holes." There is typically one fixed hole covering the range 512 Kbytes to 1 Mbyte, and one variable hole in the range between 1 Mbyte and 16 Mbytes. Software defines the location of the holes by excluding the necessary regions from the DMA address windows. Address regions that do not hit in an address window are ignored by the PCIA and may be responded to by a PCI or EISA device.

Figure 2-9 shows how the PCI memory space might be programmed. PC compatibility holes are created by placing PCI local address space over the appropriate region. DMA normally takes place through the third (mapped) window to allow all of system memory to be addressed.

Figure 2-9 PCI Memory Space, Including Compatibility Holes



2.3.2 DMA Address Windows

Each address window is specified by three registers, a Window Base (WBASEx), Window Mask (WMASKx), and a Translated Base (TBASEx) register. Refer to Chapter 3 for the format of these registers. For normal operation, the corresponding registers in each HPC must be programmed identically. This causes system memory to appear at the same address range for every PCI device.

The Window Mask Registers provide a mask that selects the size of the windowed region. The size must be one of the values shown in Table 2-10.

Table 2-10 PCI Window Mask Values

WMASK<31:0>	Size of Window
0000 0000	64 kilobytes
0001 0000	128 kilobytes
0003 0000	256 kilobytes
0007 0000	512 kilobytes
000F 0000	1 megabyte
001F 0000	2 megabytes
003F 0000	4 megabytes
007F 0000	8 megabytes
00FF 0000	16 megabytes
01FF 0000	32 megabytes
03FF 0000	64 megabytes
07FF 0000	128 megabytes
0FFF 0000	256 megabytes
1FFF 0000	512 megabytes
3FFF 0000	1 gigabyte
7FFF 0000	2 gigabytes
FFFF 0000	4 gigabytes

The Window Base Registers indicate the starting PCI address of the window. The address must be aligned on a natural boundary for the size of the window. For example, the base address for an 8-Mbyte window must specify an address that is an even multiple of 8 Mbytes. A Window Base Register contains an enable bit that enables translation and forwarding for the corresponding window. The Window Base Register also contains a Scatter/Gather enable bit that controls how the translation is done.

When a PCI address hits in the window, the PCIA generates a translated address using the Translated Base Register. If the Scatter/Gather enable bit is clear, a direct translation is performed using the mapping shown in Figure 2-10. The low-order bit of the Translated Base Register is ignored, and the remaining bits are concatenated with the offset within the window to form a 40-bit system bus address.

In Figure 2-10 the value of *n* is controlled by the Window Mask value. For example, to create a 1-Mbyte window starting at 0010 0000 and ending at 001F FFFF, the following values would be programmed into the registers named:

WMASK value:000F 0000   (n = 19)
WBASE value: 0010 0000
TBASE value: 0000 0010

PCI DMA addresses in this range will be forwarded to main memory, translated to the CPU address range 00 0080 0000 to 00 008F FFFF.

If the Scatter/Gather bit is set, the PCI address is interpreted as a page plus offset address, using an 8-Kbyte page size. The page portion of the address is used to index into the address translation map RAM, which provides a translated page number. The translation is shown in Figure 2-11. If the translation points to map a RAM entry that is marked as invalid, an error is reported.

Figure 2-10 Direct DMA Address Translation

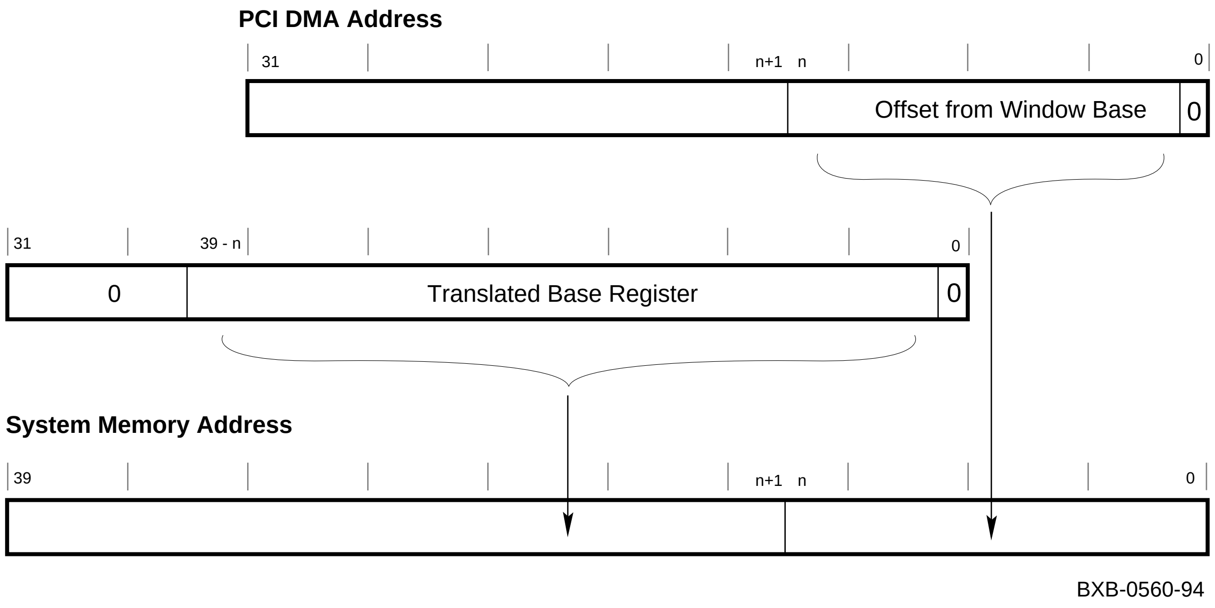


Figure 2-11 Mapped DMA Address Translation for DWLPA and DWLPB with TBIT=0

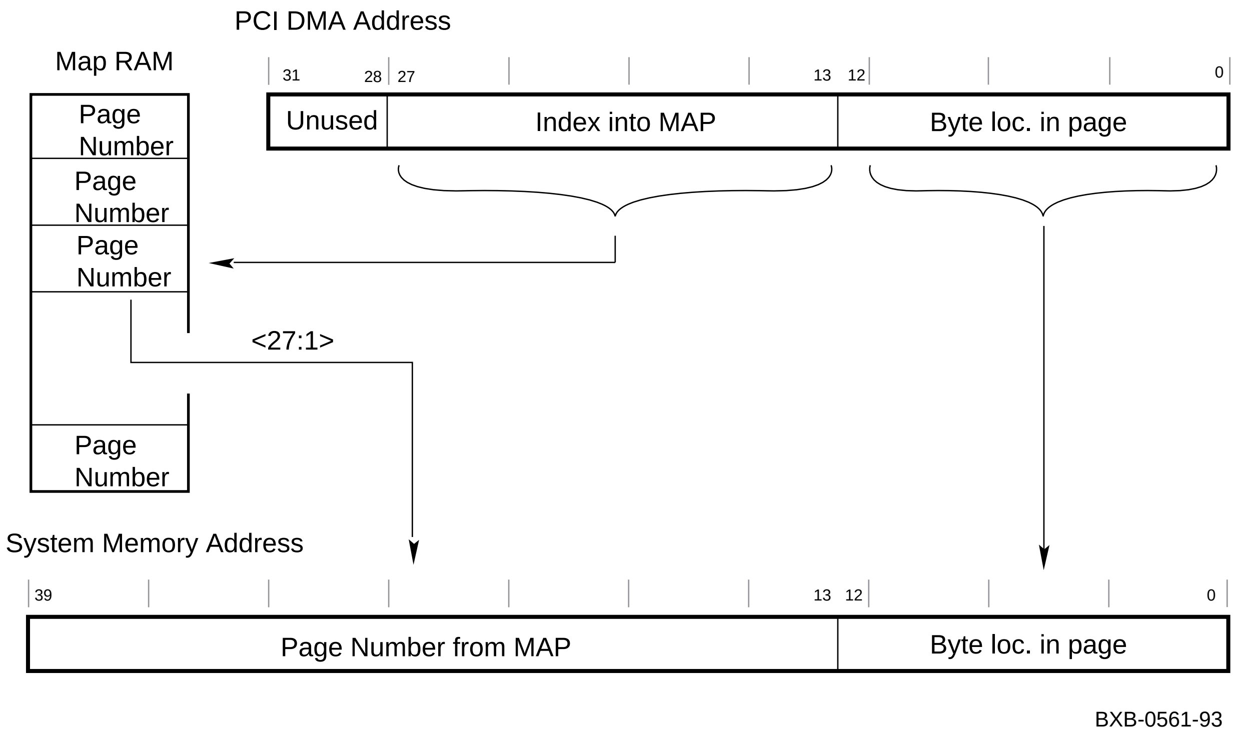
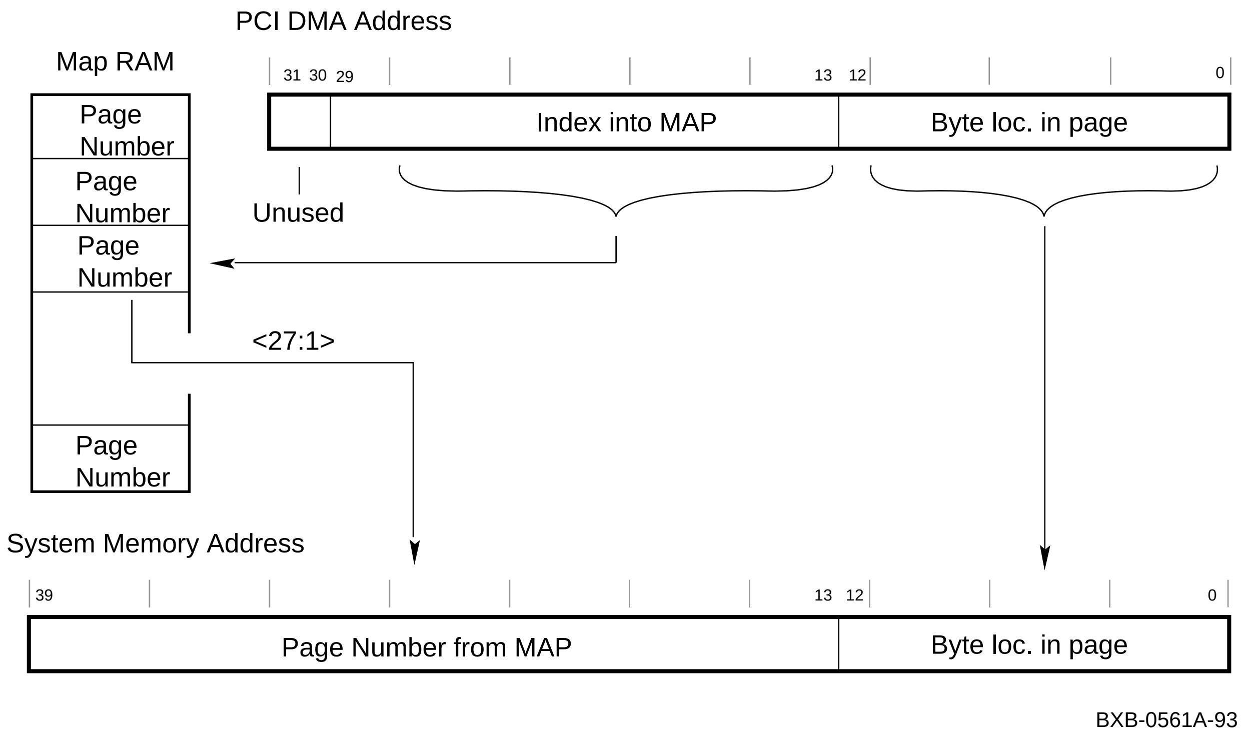


Figure 2-12 Mapped DMA Address Translation for DWLPB with TBIT=1



Chapter 3

Registers

The control and status registers for the PCI adapter are implemented by three HPCs and are on the PCIA module. The registers are accessible at addresses in the range used for PCI configuration space access. Each of the three HPC chips has a set of these registers. In general, the registers implemented in an HPC control one PCI bus. Since the registers are all aligned longwords in a sparse space, the addresses shown below would be OR'ed with A<6:0>=0011000=0x18 when referenced.

Table 3-1 PCIA Registers

CPU Addr <33:0>	Register Name	Mnemonic
3 8000 0000	PCI 0 Bus Control Register	CTL0
3 8000 0080	PCI 0 Master Retry Limit	MRETRY0
3 8000 0100	PCI 0 General Purpose Register	GPR0
3 8000 0180	PCI 0 Error Summary Register	ERR0
3 8000 0200	PCI 0 Failing Address Register	FADR0
3 8000 0280	PCI 0 Interrupt Mask Register	IMASK0
3 8000 0300	PCI 0 Diagnostic Register	DIAG0
3 8000 0380	PCI 0 Interrupt Pending Register	IPEND0
3 8000 0400	PCI 0 Interrupt in Progress Register	IPROG0
3 8000 0480	PCI 0 Window Mask Register A	WMASK_A0
3 8000 0500	PCI 0 Window Base Register A	WBASE_A0
3 8000 0580	PCI 0 Translated Base Register A	TBASE_A0
3 8000 0600	PCI 0 Window Mask Register B	WMASK_B0
3 8000 0680	PCI 0 Window Base Register B	WBASE_B0
3 8000 0700	PCI 0 Translated Base Register B	TBASE_B0
3 8000 0780	PCI 0 Window Mask Register C	WMASK_C0
3 8000 0800	PCI 0 Window Base Register C	WBASE_C0
3 8000 0880	PCI 0 Translated Base Register C	TBASE_C0
3 8000 0900	PCI 0 Error Interrupt Vector Register	ERRVEC0
3 8000 0980 - 3 8000 0FFF	Reserved	
3 8000 1000 - 3 8000 17FF	PCI 0 Device Interrupt Vector Registers	DEVVEC0
3 8000 1800 - 3 8000 1FFF	Reserved	
3 8000 2000	PCIA Special Cycle Register	SCYCLE
3 8000 2080	PCIA Interrupt Acknowledge Register	IACK
3 8000 2100 - 3 801F FFFF	Reserved	
3 8020 0000	PCI 1 Bus Control Register	CTL1
3 8020 0080	PCI 1 Master Retry Limit	RETRY1
3 8020 0100	PCI 1 General Purpose Register	GPR1
3 8020 0180	PCI 1 Error Summary Register	ERR1
3 8020 0200	PCI 1 Failing Address Register	FADR1
3 8020 0280	PCI 1 Interrupt Mask Register	IMASK1
3 8020 0300	PCI 1 Diagnostic Register	DIAG1
3 8020 0380	PCI 1 Interrupt Pending Register	IPEND1
3 8020 0400	PCI 1 Interrupt in Progress Register	IPROG1
3 8020 0480	PCI 1 Window Mask Register A	WMASK_A1
3 8020 0500	PCI 1 Window Base Register A	WBASE_A1
3 8020 0580	PCI 1 Translated Base Register A	TBASE_A1
3 8020 0600	PCI 1 Window Mask Register B	WMASK_B1
3 8020 0680	PCI 1 Window Base Register B	WBASE_B1
3 8020 0700	PCI 1 Translated Base Register B	TBASE_B1
3 8020 0780	PCI 1 Window Mask Register C	WMASK_C1
3 8020 0800	PCI 1 Window Base Register C	WBASE_C1
3 8020 0880	PCI 1 Translated Base Register C	TBASE_C1
3 8020 0900	PCI 1 Error Interrupt Vector Register	ERRVEC1
3 8020 0980 - 3 8020 0FFF	Reserved	
3 8020 1000 - 3 8020 17FF	PCI 1 Device Interrupt Vector Registers	DEVVEC1
3 8020 1800 - 3 803F FFFF	Reserved	

Table 3-1 PCIA Registers (Continued)

CPU Addr <33:0>	Register Name	Mnemonic
3 8040 0000	PCI 2 Bus Control Register	CTL2
3 8040 0080	PCI 2 Master Retry Limit	MRETRY2
3 8040 0100	PCI 2 General Purpose Register	GPR2
3 8040 0180	PCI 2 Error Summary Register	ERR2
3 8040 0200	PCI 2 Failing Address Register	FADR2
3 8040 0280	PCI 2 Interrupt Mask Register	IMASK2
3 8040 0300	PCI 2 Diagnostic Register	DIAG2
3 8040 0380	PCI 2 Interrupt Pending Register	IPEND2
3 8040 0400	PCI 2 Interrupt in Progress Registers	IPROG2
3 8040 0480	PCI 2 Window Mask Register A	WMASK2
3 8040 0500	PCI 2 Window Base Register A	WBASE2_A1
3 8040 0580	PCI 2 Translated Base Register A	TBASE2_A1
3 8040 0600	PCI 2 Window Mask Register B	WMASK_B1
3 8040 0680	PCI 2 Window Base Register B	WBASE_B1
3 8040 0700	PCI 2 Translated Base Register B	TBASE_B1
3 8040 0780	PCI 2 Window Mask Register C	WMASK_C1
3 8040 0800	PCI 2 Window Base Register C	WBASE_C1
3 8040 0880	PCI 2 Translated Base Register C	TBASE_C1
3 8040 0900	PCI 2 Error Interrupt Vector Register	ERRVEC2
3 8040 0980 - 3 8040 0FFF	Reserved	
3 8040 1000 - 3 8040 17FF	PCI 2 Device Interrupt Vector Registers	DEVVEC2
3 8040 1800 - 3 807F FFFF	Reserved	
3 8080 0000	PCIA Slot Present Register	PRESENT
3 8080 0080 - 3 809F FFFF	Aliases of PRESENT	
3 80A0 0000	PCIA TBIT Register	TBIT
3 80A0 0080 - 3 80BF FFFF	Aliases of TBIT	
3 80C0 0000	PCIA Module Control Register	MCTL
3 80C0 0080 - 3 80DF FFFF	Aliases of MCTL	
3 80E0 0000	PCIA Information Base Repair Register	IBR
3 80E0 0080 - 3 80FF FFFF	Aliases of IBR	
3 8100 0000 - 3 813F FFFF	Address Map RAM, 128 KB (32K entries) DWLPA: DMA_ADDR<29:28>=00,01,10,11 DWLPB: DMA_ADDR<29:28>=00	
3 8140 0000 - 3 817F FFFF	Address Map RAM, 128 KB (32K entries) DWLPA: DMA_ADDR<29:28>=00,01,10,11 (3 8100 0000 area aliased) DWLPB: DMA_ADDR<29:28>=01	
3 8180 0000 - 3 81BF FFFF	Address Map RAM, 128 KB (32K entries) DWLPA: DMA_ADDR<29:28>=00,01,10,11 (3 8100 0000 area aliased) DWLPB: DMA_ADDR<29:28>=10	
3 81C0 0000 - 3 81FF FFFF	Address Map RAM, 128 KB (32K entries) DWLPA: DMA_ADDR<29:28>=00,01,10,11 (3 8100 0000 area aliased) DWLPB: DMA_ADDR<29:28>=11	
3 8200 0000 - 3 83FF FFFF	Reserved. 1 MB (byte accesses only)	
3 8400 0000 - 3 FFFF FFFF	Aliases of 3 8000 0000 to 3 83FF FFFF	

The register descriptions use notations as defined in Table 3-2.

Table 3-2 Register Bit Types

Type of Bit	Description
R/W	Read/write.
RO	Read only. Writes ignored. The value is only changed by hardware.
WO	Write only. Read value not defined.
R/W1C	Read. Write one to clear.
MBZ	Fields marked MBZ (Must Be Zero) are currently unused. When writing, such fields should be written with zeros. When reading, the contents of such fields should be treated as unpredictable.

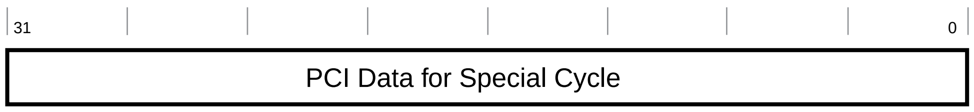
NOTE: Except as noted in Table 3-1 access to regions marked "Reserved" results in a PCIA illegal CSR address error.

The hardware initialization values of the bits and bit fields are given next to the type of the bit.

SCYCLE — PCIA Special Cycle Register

Address	3 8000 2000
Access	WO

The SCYCLE register is used to perform a PCI special cycle bus transaction. A value written to this register is transmitted on all three PCI buses using a PCI special cycle.



BXB-0541-93

Table 3-3 SCYCLE Register Bit Definitions

Bit(s)	Type	Description
<31:0>	WO	**Special Cycle Data.** Data to be transmitted during the special cycle.

IACK — PCIA Interrupt Acknowledge Register

Address	3 8000 2080
Access	RO

When an EISA interrupt occurs, the IACK register is used to perform a PCI Interrupt Acknowledge bus transaction. Reading this register causes an IACK transaction on the PCI bus. The interrupt vector is returned by the PCI/EISA bridge.

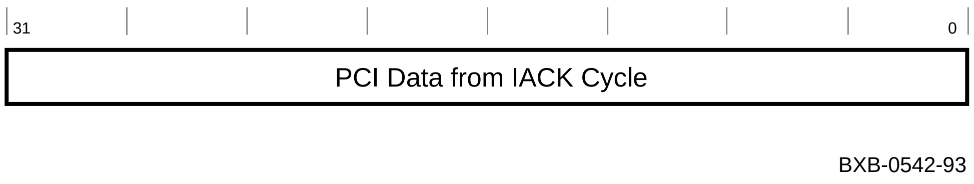


Table 3-4 IACK Register Bit Definitions

Bit(s)	Type	Description
<31:0>	RO	**Interrupt Acknowledge Data.** Data received during the Interrupt Acknowledge cycle.

PRESENT — PCIA Slot Present Register

Address	3 8080 0000
Access	RO

The PRESENT register indicates the presence and power requirements of each PCI module on the PCI as well as the revision level of the PCIA. The register allows firmware to implement configuration restrictions. Operating system software should not use this register to size the PCI, since the correspondence between devices and slots is configuration dependent. To size the PCI space, use PCI configuration space; do not use this register.

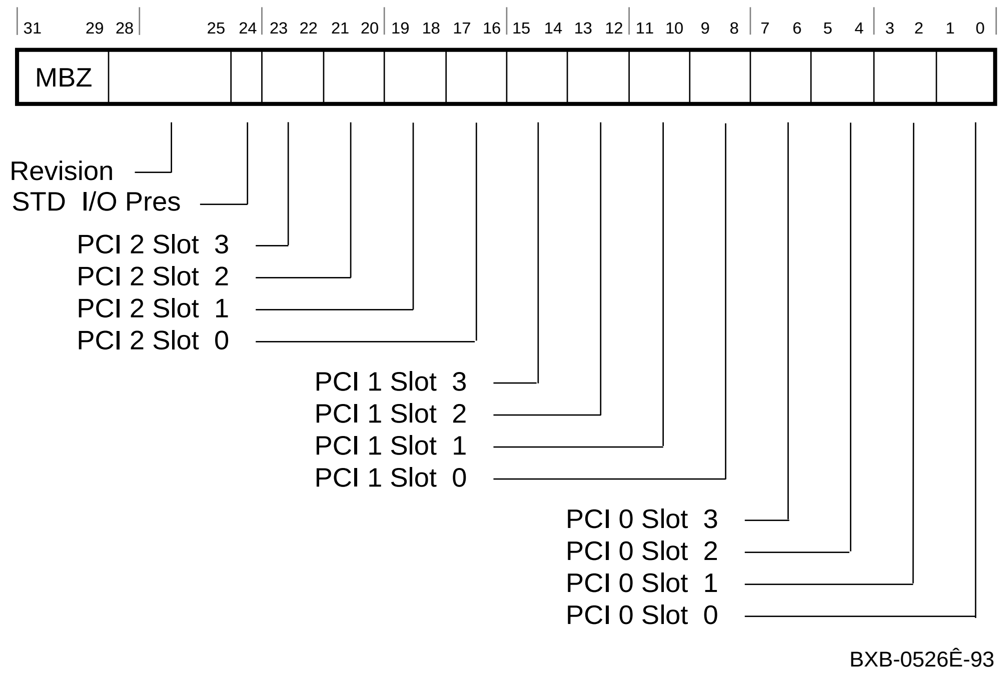


Table 3-5 PRESENT Register Bit Definitions

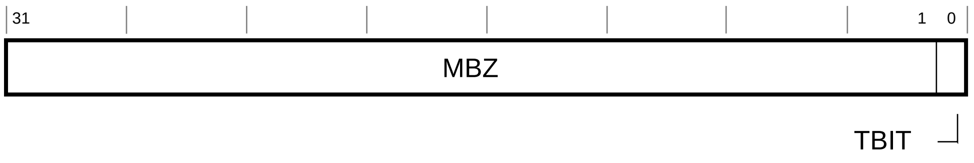
Bit(s)	Type	Description
<31:19>	RO	**Reserved.** Must be zero.
<28:25>	RO	**Module Revision**. Identifies the revision of the PCIA module.
<24>	RO	**Standard I/O Present.** When set, this bit indicates that a standard I/O EISA bridge module is in the dedicated slot. The bit is clear if the module is not installed.
<23:22>	RO	**Contents of PCI 2 - Slot 3**. These bits indicate if a module is present in the slot and its power requirements. The encoding of these fields is based on the corresponding PCI bus signals:
<21:20>	RO	**Contents of PCI 2 - Slot 2**
<19:18>	RO	**Contents of PCI 2 - Slot 1**
<17:16>	RO	**Contents of PCI 2 - Slot 0**
<15:14>	RO	**Contents of PCI 1 - Slot 3**
<13:12>	RO	**Contents of PCI 1 - Slot 2**
<11:10>	RO	**Contents of PCI 1 - Slot 1**
<9:8>	RO	**Contents of PCI 1 - Slot 0**
<7:6>	RO	**Contents of PCI 0 - Slot 3**
<5:4>	RO	**Contents of PCI 0 - Slot 2**
<3:2>	RO	**Contents of PCI 0 - Slot 1**
<1:0>	RO	**Contents of PCI 0 - Slot 0**

Field Value	PCI Module
00	No module present
01	25 watt module present
10	15 watt module present
11	7.5 watt module present

TBIT — PCIA TBIT Register

Address	3 80A0 0000
Access	WO, 0

The TBIT register allows selection of a 128-entry map RAM on the DWLPB motherboard without an update of the operating system. On power-up, TBIT<0>=0 masks out DMA_ADDR<29:28> and CPU_ADDR<23:22> to the map RAM. This register reads an alias of the PRESENT register.



BXB-0524A-96

Table 3-6 TBIT Register Bit Definitions

Bit(s)	Type	Description
<31:1>	MBZ	**Reserved.** Write as zero.
<0>	WO, 0	**Enable New Map.** Selects the 512KB or 128KB map RAM on the DWLPB module. Not implemented on the DWLPA. If zero, a 128KB map RAM is selected on the DWLPB as follows:

DMA_ADDR <29:28>	CPU Address (128KB Areas) (TBIT<0>=0)
00,01,10,11	3 8100 0000 - 3 813F FFFF
00,01,10,11	3 8140 0000 - 3 817F FFFF (3 8100 0000 area aliased)
00,01,10,11	3 8180 0000 - 3 81BF FFFF (3 8100 0000 area aliased)
00,01,10,11	3 81C0 0000 - 3 8FFF FFFF (3 8100 0000 area aliased)

Table 3-6 TBIT Register Bit Definitions (Continued)

Bit(s)	Type	Description
<0>	WO, 0	When one, a 512KB map RAM is selected on the DWLPB as follows:

DMA_ADDR <29:28>	CPU Address (128KB Areas) (TBIT<0>=1)
00	3 8100 0000 - 3 813F FFFF
01	3 8140 0000 - 3 817F FFFF
10	3 8180 0000 - 3 81BF FFFF
11	3 81C0 0000 - 3 8FFF FFFF

MCTL — PCIA Module Control Register

Address 3 80C0 0000
Access WO, 0

The MCTL register provides control bits that affect the entire PCIA module. This register reads an alias of the PRESENT register.

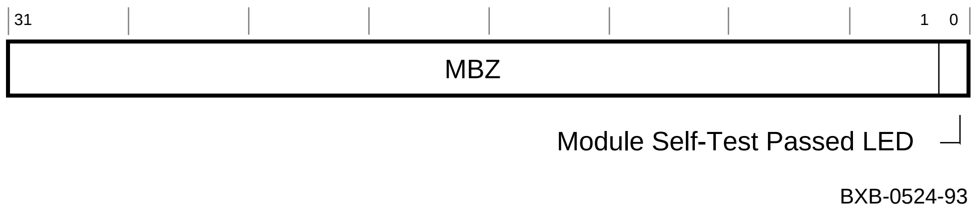


Table 3-7 MCTL Register Bit Definitions

Bit(s)	Type	Description
<31:1>	MBZ	**Reserved.** Write as zero.
<0>	WO, 0	**Module Self-Test Passed LED**. If this bit is set, the module self-test passed LED is lit. If this bit is clear, the LED is not lit.

IBR — PCIA Information Base Repair Register

Address 3 80E0 0000
Access R/W

The IBR register provides access to the serial EEPROM used to store the module serial number and repair information. To access the EEPROM, software updates the IBR register to transfer command, address, and data to and from the device. Writing an alternating one and zero pattern to the Serial EEPROM Clock bit implements the serial data clock used for the EEPROM protocol.

Command, address, and write data are serially transferred to the EEPROM by writing to the Transmit Data bit in conjunction with the Clock bit. EEPROM read data and responses are read serially from the Receive Data bit in accordance with the Clock bit. When receiving EEPROM read data and responses, the Transmit Data bit must be set to one.

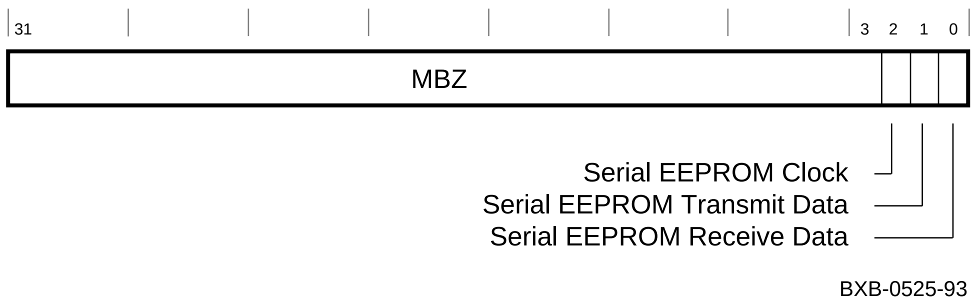


Table 3-8 IBR Register Bit Definitions

Bit(s)	Type	Description
<31:3>	MBZ	**Reserved.** Write as zero. When read, returns bits <31:3> of PRESENT.
<2>	R/W, 0	**Serial EEPROM Clock**. This bit is used to control the clock input to the serial EEPROM. When this bit is set, the EEPROM serial clock input is forced high. When this bit is cleared, the serial clock input is forced low.
<1>	R/W, 1	**Serial EEPROM Transmit Data.** This bit is used to control the data input line to the serial EEPROM. When this bit is set, the line is asserted high; when the bit is clear, the line is asserted low.
<0>	RO, 1	**Serial EEPROM Receive Data.** This bit is used to read the value of the EEPROM serial data line. Bit <1> must be set in order to receive data on this line.

CTLx — PCIx Bus Control Register

Address 3 80x0 0000
Access R/W

The CTLx register contains read/write control bits for one HPC and its corresponding PCI bus. There are three such registers, one for each PCI. Each must be programmed identically in all HPCs on the same hose before access to PCI devices is attempted.

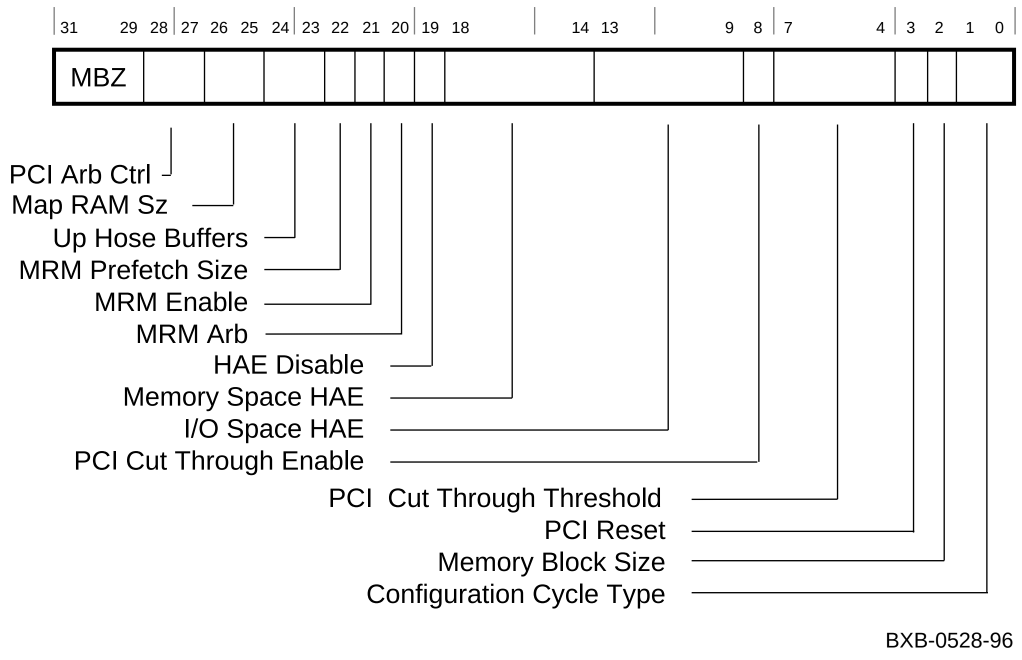


Table 3-9 CTLx Register Bit Definitions

Bit(s)	Type	Description
<31:29>	RO, 0	**Reserved**. Must be zero.
<28:27>	R/W, 0	**PCI Arbitration Control.** These bits are initialized to zero and select the round robin mode for arbitration. These bits must be written as zeros.

Table 3-9 CTLx Register Bit Definitions (Continued)

Bit(s)	Type	Description
<26:25>	R/W, 0	**Scatter/Gather Map RAM Size.** These bits must be set to indicate the size of the scatter/gather map RAM attached to the PCIA. The possible values are: CTLx<26:25> / Map RAM Size: 00 — 128 KB (32K entries – DWLPA) 01 — 256 KB (64K entries) 10 — Disable S/G cache in HPC 11 — 512 KB (128K entries – DWLPB)
<24:23>	R/W, 01	**I/O Port Up Hose Buffers.** These bits must be programmed to indicate the number of buffers available for Up Hose packets in the I/O port at the TLSB end of the hose. The possible values are: CTLx<26:25> / Up Hose Buffers: 00 — 4 buffers 01 — 1 buffer 10 — 2 buffers 11 — 3 buffers (KFTHA and KFTIA) This field defaults to 01, which is safe for all implementations. Software must program this field to 11 (3 buffers) to obtain full PCIA performance.
<22>	R/W, 0	**Memory Read Multiple Prefetch Size.** This bit controls the number of cache blocks the PCIA prefetches in response to a DMA memory read multiple transaction on the PCI. If set, the PCIA fetches 4 cache blocks in addition to the originally requested block. If clear, the PCIA fetches two additional blocks. This bit is ignored if bit <21> is clear.
<21>	R/W, 0	**Memory Read Multiple Enable.** When this bit is set, the PCIA responds to a PCI memory read multiple transaction by prefetching additional cache blocks of data from main memory. If this bit is clear, a memory read multiple transaction is handled the same as a PCI memory read. By default, memory read multiple support is disabled. Enabling it increases throughput for the requesting device, at the cost of increased latency for other devices on the bus. Normally, software should enable this function by setting this bit.
<20>	R/W, 0	**Memory Read Multiple Arbitration.** When this bit is set, an HPC transmits all Up Hose packets associated with a memory read multiple transaction after successful arbitration for the PCIA UP BUS. Both the initial read and all prefetch packets are transmitted. If this bit is clear, an HPC transmits only one Up Hose packet per successful arbitration. If bit <21> is clear, this bit is ignored, and only one Up Hose packet is transmitted per arbitration.

Table 3-9 CTLx Register Bit Definitions (Continued)

Bit(s)	Type	Description
<19>	R/W, 0	**Hardware Address Extension Disable.** This bit disables the use of the Memory Space and I/O Space Hardware Address Extensions. When this bit is zero, the PCIA uses the HAE values to map sparse space addresses. When this bit is one, the PCIA ignores the HAE values and uses the addresses directly as passed in the hose packets. This allows the system bus interface to implement a different size sparse mapping, by providing its own HAE registers.
<18:14>	R/W, 0	**Memory Space Hardware Address Extension.** This field provides an extension for sparse space mapping of PCI memory space. When CPU address bits <31:29> are non-zero, this field supplies PCI Address bits <31:27>.
<13:9>	R/W, 0	**I/O Space Hardware Address Extension.** This field provides an extension for the sparse mapping of PCI I/O space. When CPU address bits <31:29> are non-zero, this field supplies PCI address bits <31:27>.
<8>	R/W, 0	**PCI Cut-through Enable.** Setting this bit enables cut through of DMA read data transfers. If this bit is clear, DMA read transfers are buffered in the PCIA before it is transmitted on the PCI. Software should set this bit for normal operation.
<7:4>	R/W, 0	**PCI Cut-through Threshold.** This field sets the number of longwords of DMA read data the PCIA will buffer before beginning to transmit data on the PCI. This only affects transfers not aligned on 64-byte boundaries. Software should initialize this field to zero for normal operation.
<3>	WO, 0	**PCI Reset.** When this bit is set, the PCIA asserts PCI_RESET_L to reset the PCI bus. The signal is deasserted when the bit is cleared. Software must time the assertion of this bit to assert reset for at least 1 ms.
<2>	R/W, 0	**Memory Block Size.** This bit selects the maximum size of a DMA read or write transfer. When clear, the PCIA allows DMA transfers of up to 64 bytes before disconnecting. When set, the PCIA limits DMA transfers to a maximum of 32 bytes.
<1:0>	R/W, 0	**Configuration Cycle Type.** These bits select the type of PCI configuration cycle to be performed on accesses to configuration space. They supply bits <1:0> of the PCI address. Legal values are: 00 - PCI Type 0 Configuration Cycle 01 - PCI Type 1 Configuration Cycle

MRETRYx — PCIx Master Retry Limit Register

Address	3 80x0 0000
Access	R/W

The MRETRY register controls the number of times the PCIA will attempt to restart a burst transfer that has been disconnected by the target. If this count is exceeded, the PCIA reports an error. Software must initialize this field to a value of 400000 (hex) before performing any PCI accesses. This value corresponds to about 1 second of retries.

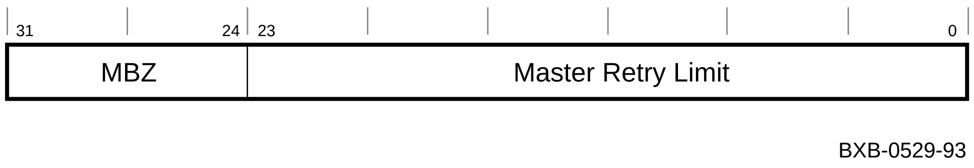


Table 3-10 MRETRYx Register Bit Definitions

Bit(s)	Type	Description
<31:24>	MBZ	**Reserved**. Must be zero.
<23:0>	R/W	**Master Retry Limit.** This value specifies the number of times the PCIA will attempt to restart a burst transfer that has been disconnected by the target.

GPRx — PCIx General Purpose Register

Address 3 80x0 0100
Access R/W

The GPR register is available for diagnostics and scratch use. Reads and writes to this register have no effect on the PCI bus.

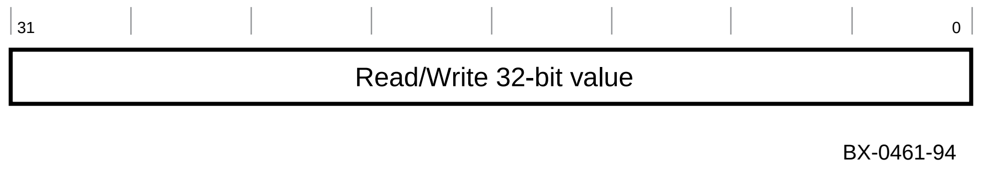


Table 3-11 GPRx Register Bit Definitions

Bit(s)	Type	Description
<31:0>	R/W	**GPR Contents.** This register can be read or written with any 32-bit value. There is no other effect on the HPC or the PCI bus.

ERRx — PCIx Error Summary Register

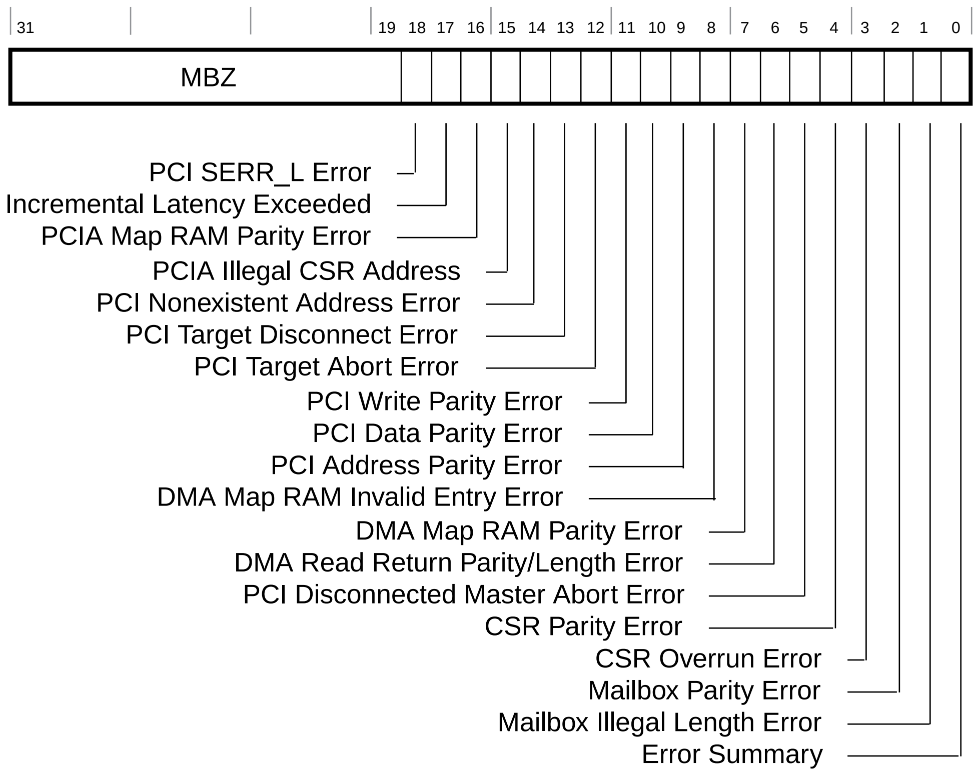
Address 3 80x0 0180
Access R/W

When a nonfatal error is detected, one or more bits in this register are set. An error interrupt is generated only when the first error bit is set in this register. No further error interrupts are generated until all pending errors are handled and the error bits cleared.

When an error interrupt is received, take the following steps

1. **Read the ERRx register to determine the cause of the interrupt.**
2. **Perform appropriate logging or recovery for the errors indicated in the register.**
3. **Write back the contents of the ERRx register to clear the interrupts.**
4. **Read the ERRx register again and if bits are set, go to step 2.**
5. **Return from the interrupt.**

Fatal errors are not reported in this register since PCIA registers are not accessible following a fatal error.



BX-0462-94

Table 3-12 ERRx Register Bit Definitions

Bit(s)	Type	Description
<31:19>	MBZ	**Reserved.** Must be zero.
<18>	R/W1C, 0	**PCI SERR_L Error.** This bit is set if any device on the PCI asserts the PCI SERR_L signal. This signal is asserted when a PCI device detects an address parity error or other serious error condition.
<17>	R/W1C, 0	**Incremental Latency Exceeded.** This bit is set if the HPC, as a PCI bus target, delays more than 8 PCI cycle times between supplying contiguous longwords of a PCI burst transaction. This bit is a performance monitoring aid and does not indicate an error condition. The PCIA completes the transaction normally, and no interrupt is generated.
<16>	R/W1C, 0	**PCIA Map RAM Parity Error.** A CPU-initiated read of a scatter/gather map RAM location received data with a parity error. This bit is set in ERR0 only.
<15>	R/W1C, 0	**PCIA Illegal CSR Address.** A command packet was received referencing a reserved address used for HPC CSRs. This bit will only be set in register ERR0.
<14>	R/W1C, 0	**PCI Nonexistent Address Error.** No PCI device responded when the HPC issued a valid address cycle. This bit will only be set in register ERR0.
<13>	R/W1C, 0	**PCI Target Disconnect Error.** A PCI target exceeded the limit of target disconnects specified in the MRETRYx register.
<12>	R/W1C, 0	**PCI Target Abort Error.** A PCI target issued a target abort to the HPC.
<11>	R/W1C, 0	**PCI Write Parity Error.** The HPC detected a parity error during a PCI DMA write data cycle. The PCIA was the target of the PCI operation. The FADRx register is valid when this bit is set.
<10>	R/W1C, 0	**PCI Data Parity Error.** The HPC detected a parity error while master of a PCI read data cycle or detected the assertion of the PCI PERR_L signal during a PCI write data cycle.
<9>	R/W1C, 0	**PCI Address Parity Error.** The HPC, while a target, detected a parity error on the PCI during an address phase. If other devices detect the error, this bit may be set in conjunction with the PCI SERR_L Error bit <18>. The HPC does not assert PCI SERR_L in response to this error.
<8>	R/W1C, 0	**DMA Map RAM Invalid Entry Error.** During a DMA operation, the HPC accessed a scatter/gather map RAM entry which did not have its valid bit set. The FADRx register is valid when this bit is set.

Table 3-12 ERRx Register Bit Definitions (Continued)

Bit(s)	Type	Description
<7>	R/W1C, 0	**DMA Map RAM Parity Error.** During a DMA operation, the address translation data received from the scatter/gather map RAM contained a parity error. The FADRx register is valid when this bit is set.
<6>	R/W1C, 0	**DMA Read Return Data Parity/Length Error.** A DMA read return packet was received down the hose with bad parity on one or more longwords or a packet was received that contained fewer than the expected number of longwords. The FADRx register is valid when this bit is set.
<5>	R/W1C, 0	**PCI Disconnected Master Abort Error.** The HPC detected a master abort on a PCI bus during a CSR window or mailbox transaction, and the transaction was previously claimed and disconnected by a node on the PCI bus.
<4>	R/W1C, 0	**CSR Parity Error.** A CSR command packet was received down the hose with bad parity on one or more longwords.
<3>	R/W1C, 0	**CSR Overrun Error.** A CSR command packet was received down the hose and contained too many longwords.
<2>	R/W1C, 0	**Mailbox Parity Error.** A Mailbox Command packet was received down the hose with bad parity on one or more longwords.
<1>	R/W1C, 0	**Mailbox Illegal Length Error.** A Mailbox Command packet was received down the hose and contained the wrong number of longwords.
<0>	R/W1C, 0	**Error Summary.** This bit is set if any of the other bits are set in this register.

FADRx — PCIx Failing Address Register

Address 3 80x0 0200
Access RO

When an error occurs during a DMA operation, this register is latched with the address from the corresponding PCI address phase. The error that caused the register to latch is recorded in the ERRx register.

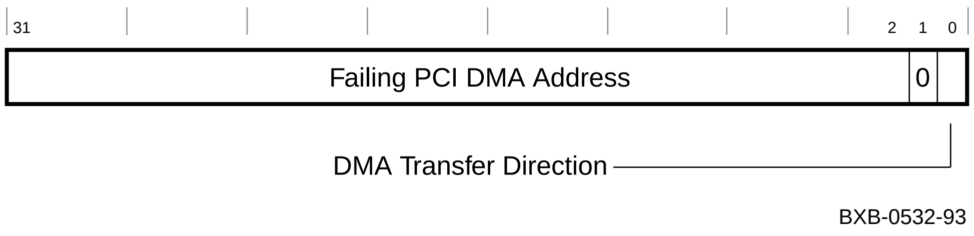


Table 3-13 FADRx Register Bit Definitions

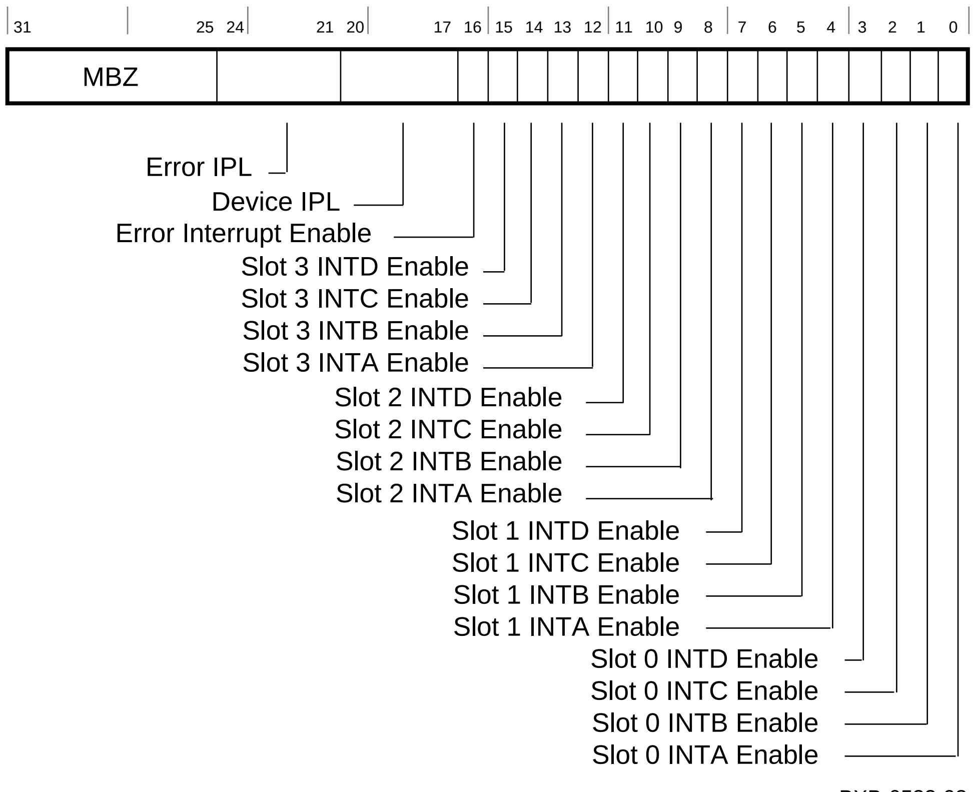
Bit(s)	Type	Description
<31:2>	RO, 0	**Failing PCI DMA Address.** When a DMA error occurs, this register latches the address value from the corresponding PCI address phase. When the ERRx register does not indicate a DMA error, the contents of this register are unpredictable. Note that PCI memory addresses are always longword aligned.
<1>	RO, 0	**Reserved.**
<0>	RO, 0	**DMA Transfer Direction.** When this register is latched, this bit is set to zero if the DMA transfer was a read from memory or set to one if the DMA transfer was a write to memory.

IMASKx — PCIx Interrupt Mask Register

Address	3 80x0 0280
Access	R/W

The IMASKx register selects which PCI interrupt sources can generate interrupt requests to the CPU. The HPC has four interrupt inputs from each PCI slot. The register also sets the interrupt priority level of error interrupts generated by the PCIA and device interrupts generated by PCI devices. All PCI devices interrupt at the same IPL. The error and device IPL fields must be programmed the same in each HPC.

Although some HPC interrupt inputs are also used for the PCI devices on the standard I/O module, no conflict exists because the affected slots cannot be populated when the standard I/O module is installed.



BXB-0533-93

Table 3-14 IMASKx Register Bit Definitions

Bit(s)	Type	Description
<31:25>	MBZ	**Reserved**. Must be zero.
<24:21>	R/W, 0	**Error Interrupt Priority Level.** This field selects the interrupt priority level to be used for interrupts requested by the HPC in response to errors. This 4-bit field uses the following encoding:

IMASKx<24:21>	Priority Level
1000	IPL 17
0100	IPL 16
0010	IPL 15
0001	IPL 14

Bit(s)	Type	Description
		Software must initialize this field so that only one bit is set. Setting multiple bits may cause spurious interrupts to be reported. Software must set the Device Interrupt Priority Level and the Error Interrupt Priority Level to different values to ensure that interrupts are not lost by the I/O port. It is recommended that the error interrupt priority be programmed to IPL 17.
<20:17>	R/W, 0	**Device Interrupt Priority Level.** This field selects the interrupt priority level to be used for interrupts requested by all PCI devices. This 4-bit field uses the following encoding:

IMASKx<20:17>	Priority Level
1000	IPL 17
0100	IPL 16
0010	IPL 15
0001	IPL 14

Bit(s)	Type	Description
		Software must initialize this field so that one bit is set. Setting multiple bits may cause spurious interrupts to be reported. Software must set the Device Interrupt Priority Level and the Error Interrupt Priority Level to different values to ensure that interrupts are not lost by the I/O port.
<16>	R/W, 0	**Error Interrupt Enable.** Enables recognition of interrupt requests generated by the HPC in response to error conditions. The cause of the error interrupt is reported in the ERRx register.
<15>	R/W, 0	**Slot 3 - Interrupt D Enable.** Enables recognition of interrupt requests from PCI interrupt line D in this slot.
<14>	R/W, 0	**Slot 3 - Interrupt C Enable.** Enables recognition of interrupt requests from PCI interrupt line C in this slot.
<13>	R/W, 0	**Slot 3 - Interrupt B Enable.** Enables recognition of interrupt requests from PCI interrupt line B in this slot.

Table 3-14 IMASKx Register Bit Definitions (Continued)

Bit(s)	Type	Description
<12>	R/W, 0	**Slot 3 - Interrupt A Enable.** Enables recognition of interrupt requests from PCI interrupt line A in this slot.
<11>	R/W, 0	**Slot 2 - Interrupt D Enable.** Enables recognition of interrupt requests from PCI interrupt line D in this slot.
<10>	R/W, 0	**Slot 2 - Interrupt C Enable.** Enables recognition of interrupt requests from PCI interrupt line C in this slot.
<9>	R/W, 0	**Slot 2 - Interrupt B Enable.** Enables recognition of interrupt requests from PCI interrupt line B in this slot.
<8>	R/W, 0	**Slot 2 - Interrupt A Enable.** Enables recognition of interrupt requests from PCI interrupt line A in this slot.
<7>	R/W, 0	**Slot 1 - Interrupt D Enable.** Enables recognition of interrupt requests from PCI interrupt line D in this slot.
<6>	R/W, 0	**Slot 1 - Interrupt C Enable.** Enables recognition of interrupt requests from PCI interrupt line C in this slot.
<5>	R/W, 0	**Slot 1 - Interrupt B Enable.** Enables recognition of interrupt requests from PCI interrupt line B in this slot or from the standard I/O module's SCSI device.
<4>	R/W, 0	**Slot 1 - Interrupt A Enable.** Enables recognition of interrupt requests from PCI interrupt line A in this slot or from the standard I/O module's Ethernet device.
<3>	R/W, 0	**Slot 0 - Interrupt D Enable.** Enables recognition of interrupt requests from PCI interrupt line D in this slot.
<2>	R/W, 0	**Slot 0 - Interrupt C Enable.** Enables recognition of interrupt requests from PCI interrupt line C in this slot.
<1>	R/W, 0	**Slot 0 - Interrupt B Enable.** Enables recognition of interrupt requests from PCI interrupt line B in this slot or from the standard I/O module's EISA interrupt controllers.
<0>	R/W, 0	**Slot 0 - Interrupt A Enable.** Enables recognition of interrupt requests from PCI interrupt line A in this slot, or from the EISA Bus Bridge Non-Maskable Interrupt.

DIAGx — PCIx Diagnostic Register

Address	3 80x0 0300
Access	R/W

The DIAGx register contains control bits that alter the normal behavior of the PCIA for test purposes. The register also contains control bits for the special feature that supports Memory Channel on the PCI. This register should not be modified during normal operation.

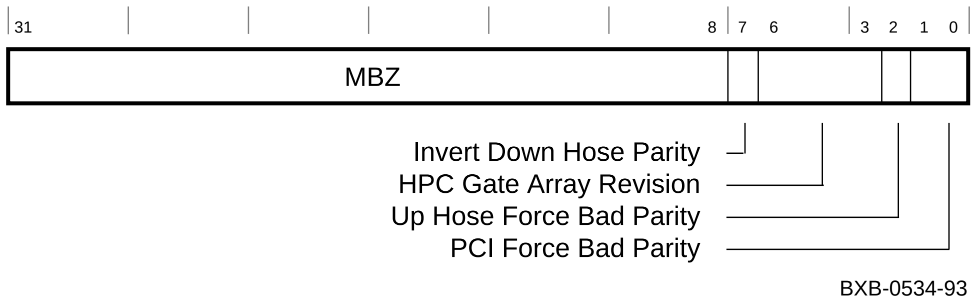


Table 3-15 DIAGx Register Bit Definitions

Bit(s)	Type	Description
<31:8>	MBZ	**Reserved**. Must be zero.
<7>	R/W, 0	**Invert Down Hose Parity.** When set, the HPC inverts its parity test on the fourth cycle of a mailbox write or CSR write packet. The bit is automatically cleared once a parity error has been detected, thus restoring access to the HPC registers.
<6:3>	R/W, 0	**HPC Gate Array Revision.** This field contains the revision of the HPC gate array. It is 0 at release.
<2>	R/W, 0	**Up Hose Force Bad Parity.** When this bit is set, the HPC forces bad parity on data cycles on the Up Hose bus. When the bit is clear, correct parity will be generated.

Table 3-15 DIAGx Register Bit Definitions (Continued)

Bit(s)	Type	Description
<1:0>	R/W, 0	**PCI Force Bad Parity.** This field controls the HPC's generation of parity when driving the PCI. The field is encoded as follows:

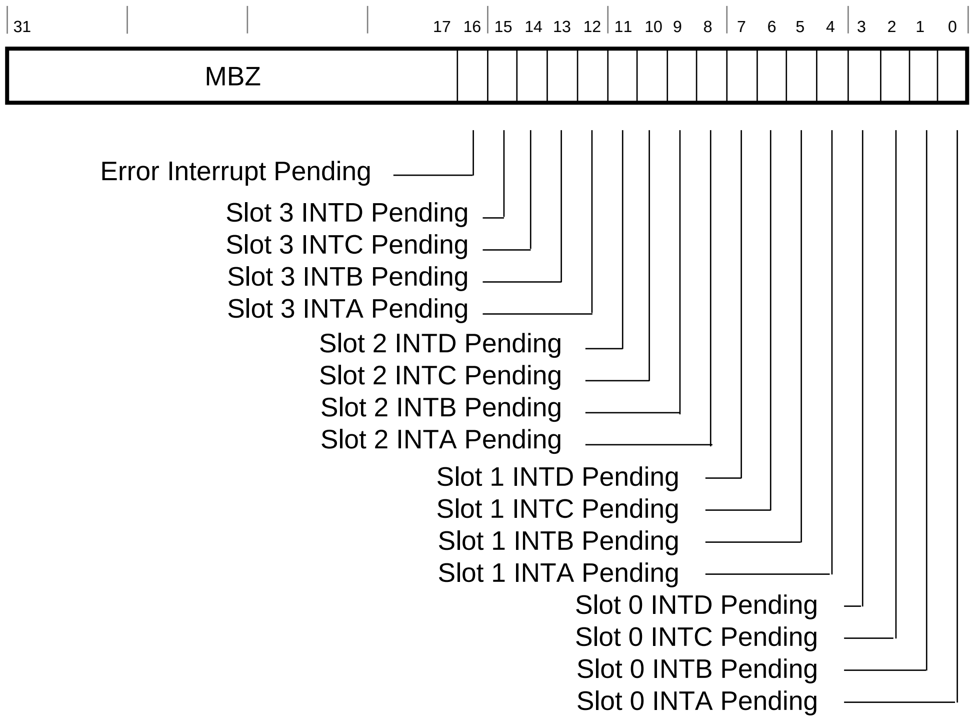
DIAG<1:0>	Action
00	Generate correct parity
01	Force bad parity on address phases
10	Force bad parity on data phases
11	Undefined

IPENDx — PCIx Interrupt Pending Register

Address 3 80x0 0380
Access RO

The IPENDx register indicates the presence of interrupt requests that have not yet resulted in INTR/IDENT packets being sent up the hose for the DWLPB. When written, all interrupt requests (except the one indicated in the IPROGx) are resampled. A write to this register of any value must be performed at the end of every interrupt service to detect any devices that are requesting a new interrupt. This register is intended for diagnostic purposes. It is possible for operating system software to read this register and anticipate an interrupt, but the interrupt will still be delivered. Some of the interrupt lines are also used for PCI devices on the standard I/O module, as described for the IMASKx register.

DWLPA supports only transition sensitive interrupts for device drivers that do not recheck device interrupt status. DWLPB supports level sensitive interrupts for drivers that do not recheck with PALcode by writing to this register.



BXB-0535-93

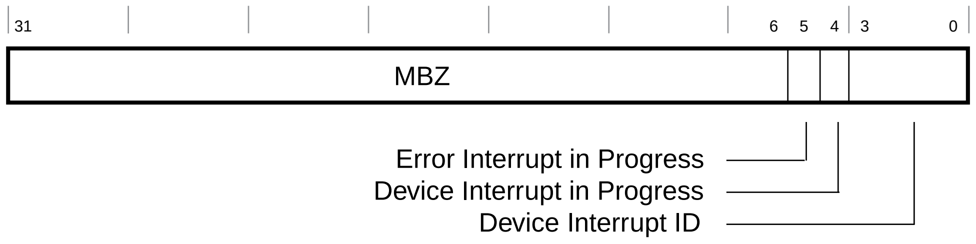
Table 3-16 IPENDx Register Bit Definitions

Bit(s)	Type	Description
<31:17>	MBZ	**Reserved.** Must be zero.
<16>	RO, 0	**Error Interrupt Pending.** An error interrupt is pending from this HPC.
<15>	RO, 0	**Slot 3 Interrupt D Pending.** An interrupt request on the INTD line is pending from this slot or not used when Standard I/O module is present.
<14>	RO, 0	**Slot 3 Interrupt C Pending.** An interrupt request on the INTC line is pending from this slot or not used when standard I/O module is present.
<13>	RO, 0	**Slot 3 Interrupt B Pending.** An interrupt request on the INTB line is pending from this slot or not used when standard I/O module is present.
<12>	RO, 0	**Slot 3 Interrupt A Pending.** An interrupt request on the INTA line is pending from this slot or not used when standard I/O module is present.
<11>	RO, 0	**Slot 2 Interrupt D Pending.** An interrupt request on the INTD line is pending from this slot or not used when standard I/O module is present.
<10>	RO, 0	**Slot 2 Interrupt C Pending.** An interrupt request on the INTC line is pending from this slot or not used when standard I/O module is present.
<9>	RO, 0	**Slot 2 Interrupt B Pending.** An interrupt request on the INTB line is pending from this slot or not used when standard I/O module is present.
<8>	RO, 0	**Slot 2 Interrupt A Pending.** An interrupt request on the INTA line is pending from this slot or not used when standard I/O module is present.
<7>	RO, 0	**Slot 1 Interrupt D Pending.** An interrupt request on the INTD line is pending from this slot or not used when standard I/O module is present.
<6>	RO, 0	**Slot 1 Interrupt C Pending.** An interrupt request on the INTC line is pending from this slot or not used when standard I/O module is present.
<5>	RO, 0	**Slot 1 Interrupt B Pending.** An interrupt request on the INTB line is pending from this slot or standard I/O SCSI interrupt pending when standard I/O module is present.
<4>	RO, 0	**Slot 1 Interrupt A Pending.** An interrupt request on the INTA line is pending from this slot or standard I/O Ethernet interrupt pending when standard I/O module is present.
<3>	RO, 0	**Slot 0 Interrupt D Pending.** An interrupt request on the INTD line is pending from this slot or not used when standard I/O module is present.
<2>	RO, 0	**Slot 0 Interrupt C Pending.** An interrupt request on the INTC line is pending from this slot or not used when standard I/O module is present.
<1>	RO, 0	**Slot 0 Interrupt B Pending.** An interrupt request on the INTB line is pending from this slot or standard I/O EISA bridge interrupt pending when standard I/O module is present.
<0>	RO, 0	**Slot 0 Interrupt A Pending.** An interrupt request on the INTA line is pending from this slot or standard I/O. Non-Maskable Interrupt pending when standard I/O module is present.

IPROGx — PCIx Interrupt in Progress Register

Address 3 80x0 0400
Access RO

The IPROGx register indicates current interrupt request(s) that are outstanding between the DWLPA/DWLPB and the KFTIA/KFTHA. Only one device interrupt and one error interrupt can be outstanding. This register is intended for diagnostic purposes. Some of the interrupt IDs are used by PCI devices on the standard I/O module, as described for the IMASKx register.



BXB-0536-93

Table 3-17 IPROGx Register Bit Definitions

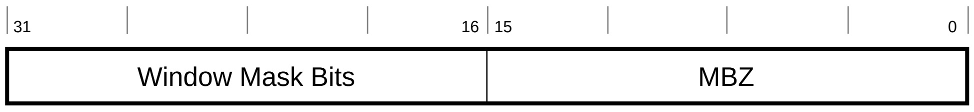
Bit(S)	Type	Description
<31:6>	MBZ	**Reserved.** Must be zero.
<5>	RO, 0	**Error Interrupt in Progress.** This bit is set when the HPC sends an error interrupt request up the hose. It is cleared when the corresponding interrupt status packet is received.
<4>	RO, 0	**Device Interrupt in Progress.** This bit is set when the HPC sends a device interrupt request up the hose. It is cleared when the corresponding interrupt status packet is received. When this bit is set, the contents of bits <3:0> are valid.
<3:0>	RO, 0	**Device Interrupt ID.** While a PCI device interrupt is in progress, this field indicates which PCI device initiated the interrupt. The contents of this field is not valid unless bit <4> is set. The field is encoded as follows:

Bits<3:0>	Interrupt Source
1111	Slot 3 INTD
1110	Slot 3 INTC
1101	Slot 3 INTB
1100	Slot 3 INTA
1011	Slot 2 INTD
1010	Slot 2 INTC
1001	Slot 2 INTB
1000	Slot 2 INTA
0111	Slot 1 INTD
0110	Slot 1 INTC
0101	Slot 1 INTB
0100	Slot 1 INTA
0011	Slot 0 INTD
0010	Slot 0 INTC
0001	Slot 0 INTB
0000	Slot 0 INTA

WMASK_xx — PCIx Window Mask Registers

Address 3 80x0 0480
Access R/W

Each Window Mask Register controls the size of a region of PCI memory space addresses that are translated and forwarded to the system memory bus during a DMA operation. The window size varies from 64 kilobytes to 4 gigabytes. Each of the three HPC mask registers (WMASK_Ax, WMASK_Bx, WMASK_Cx) defines a DMA region. At power-up, the contents are undefined. Software should initialize this register before enabling the window with the corresponding Window Base Register.



BXB-0537-93

Table 3-18 WMASK_xx Register Bit Definitions

Bit(s)	Type	Description
<31:16>	R/W	**Window Mask.** Specifies the window size as follows:

WMASK<31:16>	Window Size	WMASK<31:16>	Window Size
0000	64 Kbytes	01FF	32 Mbytes
0001	128 Kbytes	03FF	64 Mbytes
0003	256 Kbytes	07FF	128 Mbytes
0007	512 Kbytes	0FFF	256 Mbytes
000F	1 Mbyte	1FFF	512 Mbytes
001F	2 Mbytes	3FFF	1 Gbyte
003F	4 Mbytes	7FFF	2 Gbytes
007F	8 Mbytes	FFFF	4 Gbytes
00FF	16 Mbytes		

Bit(s)	Type	Description
<15:0>	MBZ	**Reserved.** Must be zero.

WBASE_xx — PCIx Window Base Registers

Address	3 80x0 0500
Access	R/W

Each Window Base Register sets the base of a range of PCI memory space addresses that are translated and forwarded to the system memory bus. The size of the window is set with the Window Mask Register. The window base address must be aligned on an even multiple of the window size. The HPC contains three base registers (WBASE_Ax, WBASE_Bx, WBASE_Cx), each of which defines the base of a DMA region.

Software must initialize the base address before enabling the window. Software must also ensure that the range of addresses in the window does not overlap with any memory space addresses recognized by PCI devices on the bus.

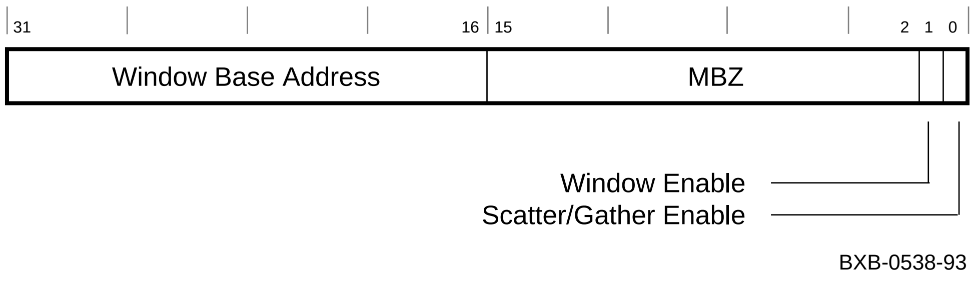


Table 3-19 WBASE_xx Register Bit Definitions

Bit(s)	Type	Description
<31:16>	R/W	**Window Base Address.** This field specifies the upper 16 bits of the PCI base address of the window.
<15:2>	MBZ	**Reserved**. Must be zero.
<1>	R/W, 0	**Window Enable.** When this bit is set, PCI memory references that hit in the window are recognized. When this bit is clear, the PCIA does not respond to addresses in this window.
<0>	R/W, 0	**Scatter/Gather Enable.** When this bit is set, PCI memory references that hit in the window are translated through the PCIA address translation map. When this bit is clear, PCI memory references that hit in the window are translated directly using the Translated Base Register.

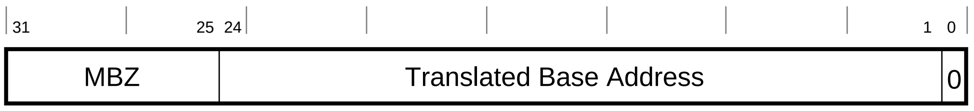
TBASE_xx — PCIx Translated Base Registers

Address 3 80x0 0580
Access R/W

The Translated Base Registers provide the system memory base address of the window from PCI memory space. This register is only used for direct translations, as selected by the Scatter/Gather Enable bit in the Window Base Register. When a PCI address is translated, the offset of the PCI address from the PCI window base address is concatenated to the value in this register to obtain the system memory address.

The HPC contains three base registers (TBASE_Ax, TBASE_Bx, TBASE_Cx) corresponding to the three possible DMA windows.

Software must initialize this register before enabling the window. The base address must be aligned on an even multiple of the window size.



BXB-0539-93

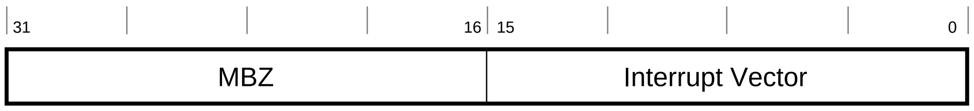
Table 3-20 TBASE_xx Register Bit Definitions

Bit(s)	Type	Description
<31:25>	MBZ	**Reserved**. Must be zero.
<24:1>	R/W	**Translated Base Address.** This field supplies some of the bits of the system memory base address corresponding to the window from PCI address space. The exact range of bits from <39:16> is determined by the WMASKxx register.
<0>	MBZ	**Reserved.** Must be zero.

ERRVEC - PCIx Error Interrupt Vector Registers

Address 3 80x0 0900 — 3 80x0 17FF
Access R/W

Each HPC contains an Error Interrupt Vector Register (ERRVECx) and an array of 16 PCI Device Interrupt Vector Registers (DEVVECx). All registers have the same format. These registers provide the vector information sent to the CPU when either a PCI interrupt or a PCIA error interrupt occurs. On power-up, the contents of these registers is not defined. They must be initialized by software before the corresponding interrupt is enabled.



BXB-0540-93

Table 3-21 ERRVEC Register Bit Definitions

Bit(s)	Type	Description
<31:16>	MBZ	**Reserved.** Must be zero.
<15:0>	R/W	**Interrupt Vector.** The interrupt vector to be sent to the CPU. For OpenVMS, PALcode must be able to identify which HPC generated the interrupt. The interrupt vector consists of: (see nested table below) For Digital UNIX, this is the value to be supplied in register "a1" when the "entInt" system entry point is called.

Bits	Definition
<15:4>	The byte offset of the entry in the SCB used to dispatch the interrupt.
<3:2>	The KFTIA or KFTHA hose port used (0-3).
<1:0>	00 = not a PCI interrupt 01 = HPC 0 10 = HPC 1 11 = HPC 2

The Error Interrupt Vector Register in each HPC is at offset 900 (hex). Each HPC has a block of Device Interrupt Vector Registers, which corresponds to the interrupt sources on the associated PCI bus. Within each block, the registers are assigned as shown in Table 3-22.

Table 3-22 PCI Interrupt Vector Offsets

Offset	Without Standard I/O	With Standard I/O
1000	Vector for Slot 0 INTA	Vector for Non-Maskable Int.
1080	Vector for Slot 0 INTB	Vector for EISA bridge
1100	Vector for Slot 0 INTC	Not used
1180	Vector for Slot 0 INTD	Not used
1200	Vector for Slot 1 INTA	Vector for Ethernet
1280	Vector for Slot 1 INTB	Vector for SCSI
1300	Vector for Slot 1 INTC	Not used
1380	Vector for Slot 1 INTD	Not used
1400	Vector for Slot 2 INTA	Not used
1480	Vector for Slot 2 INTB	Not used
1500	Vector for Slot 2 INTC	Not used
1580	Vector for Slot 2 INTD	Not used
1600	Vector for Slot 3 INTA	Not used
1680	Vector for Slot 3 INTB	Not used
1700	Vector for Slot 3 INTC	Not used
1780	Vector for Slot 3 INTD	Not used

Scatter/Gather RAM Map Entry Format

Address	3 8100 0000–3 813F FFFF
Access	R/W

For address translation, the DWLPA implements a 32K entry map and the DWLPB implements a 128K entry map. Each map entry is one longword in length and longword aligned. The register contents are undefined on power-up and must be initialized by software before use.

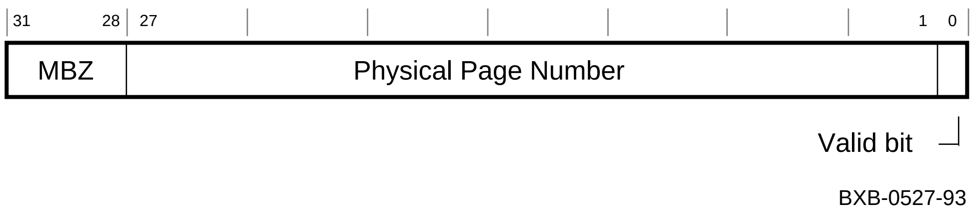


Table 3-23 Scatter/Gather Map Entry Format

Bit(s)	Type	Description
<31:28>	MBZ	**Reserved.** Must be zero.
<27:1>	R/W	**Physical Page Number.** This field provides the physical page translation to be supplied when this entry is referenced. This field provides bits <39:13> of the resulting physical address.
<0>	R/W	**Valid.** When this bit is set, the entry contains a valid address translation. If the bit is clear and an attempt to use the entry occurs, an error is reported.

Chapter 4

Functional Description

4.1 PCI Interface

The PCI interface allows for communication between the CPU and devices on the PCI. Both the CPU and devices on the PCI can initiate communication and cause the PCI bus to cycle.

4.1.1 CPU-Initiated PCI Cycles

CPU-initiated PCI cycles are transactions originating from the CPU.

4.1.1.1 CPU-Initiated PCI Commands

All CPU-initiated PCI commands are received by the PCIA in Down Hose command packets. The HPCs decode the command packet and operate as a PCI bus master while executing the command. As a PCI master, the HPC supports a subset of the full PCI command set. Table 4-1 shows the commands supported by the HPC as a PCI master.

Table 4-1 Supported PCI Commands

PCI Command	C_BE<3:0>
Interrupt Acknowledge	0000
Special Cycle	0001
I/O Read	0010
I/O Write	0011
Memory Read	0110
Memory Write	0111
Configuration Read	1010
Configuration Write	1011

A PCI interrupt acknowledge cycle is generated by reading an HPC's IACK CSR. A read to this register causes the HPC to issue the interrupt acknowledge cycle on to the PCI bus. To date, the only PCI device capable of receiving this command is the PCI to EISA bridge. The bridge returns an interrupt vector that is then sent over the Up Hose as read return data by the HPC.

A PCI special cycle is generated by writing data to the PCIA SCYCLE CSR. Any write to this register causes the unmodified write data to be broadcast to all three PCI buses with a command code of 0001. Accesses to PCI memory, I/O, and configuration spaces are specified through command and address bits in the Down Hose command packets.

All other PCI commands generate a read or write on the PCI bus at an address specified by the CPU.

The type of read or write command depends on the region of PCI address space being accessed.

4.1.1.2 PCI Addressing

PCI address space consists of PCI memory space, PCI I/O space, and PCI configuration space. PCI memory space is implemented as one 32-bit contiguous address space across all three PCI buses. Similarly, PCI I/O space is implemented as one 32-bit contiguous address space across all three PCI buses. All accesses to PCI memory and I/O space are sent to all three PCI buses. PCI configuration space is logically contiguous across all three PCI buses, but accesses may be physically sent to only one PCI bus. A type 0 access to configuration space is decoded by the HPCs and is sent to only one of the PCI buses. A type 1 access to configuration space is decoded by the HPCs and is sent to all three PCI buses.

PCI memory space can be accessed in two ways. The first method accesses memory space as a dense memory space. Accesses to this space are unmasked longword transfers. No provision for byte addressing is made with this type of access although masking of entire longwords of data can be done. Byte addressing of PCI memory space is provided by accessing it as a sparse address space. Access to sparse memory space can be byte, word, tribyte, longword, or quadword in length.

PCI I/O and configuration space can only be accessed as a sparse address space. The HPCs use the packet type and the space type bits of the Down Hose command packet to determine the type of PCI address space being accessed.

During sparse space accesses, the PCIA only receives bits <26:0> of the PCI address in the Down Hose command packet. Each HPC obtains bits <31:27> from one of the two HAE<4:0> fields (Hardware Address Extension) of its CTL CSR. The memory HAE is used during PCI sparse memory space accesses and the I/O HAE is used during PCI sparse I/O space accesses. The HAE fields are used to provide extended PCI addresses to regions of PCI sparse address space which the CPU is unable to access through its normal system address map. The PCIA also provides direct access to the lower 16 Mbytes of PCI memory and I/O space which is in the normal TLSB system addressing capability by forcing zeros on to PCI byte address bits <31:27>. The PCIA decodes PCI address bits <26:24> to determine whether the access is a direct one or uses an HAE. The PCIA uses the following translation to generate PCI address bits <31:27>.

```
if (PCI address<26:24> = 000)
        PCI address <31:27> = 00000;
else
        PCI address<31:27> = HAE<4:0>;
```

Each of the HPCs implements the above address translation if the Force HAE bit, bit <9> of the CTL CSR, is clear. If the bit is not clear, the HPCs use the contents of bits <31:27> of the address field of the Down Hose command packet as PCI address bits <31:27>. This allows other systems on which the PCIA may be used to implement their own translation and to bypass the HAE translation implemented on the PCIA.

Byte, word, and tribyte accesses to any of the PCI sparse address spaces require that a byte mask be generated for the longword of data to be transferred. CPU address bits <4:3> of the Down Hose command packet address are used as transfer size bits, with 00 indicating a byte transfer, 01 indicating a word transfer, and 10 indicating a tribyte tranfer. CPU address bits <6:5> are used to determine the starting byte of the transfer. Longword accesses are unmasked and have all byte enables asserted. Table 4-2 shows the byte enables and PCI address bits <1:0> generated for the supported byte, word, tribyte, and longword combinations of CPU address bits <6:3>.

Table 4-2 CPU Address to PCI Address Conversion

CPU Address		Resulting PCI Address and Byte Enables			
CPU ADDR <6:5>	CPU ADDR <4:3>	PCI MEM ADDR<1:0>	PCI I/O ADDR<1:0>	PCI CONF ADDR<1:0>	PCI C_BE <3:0> L
00	00	00	00	CTL<1:0>	1110
01	00	00	01	CTL<1:0>	1101
10	00	00	10	CTL<1:0>	1011
11	00	00	11	CTL<1:0>	0111
00	01	00	00	CTL<1:0>	1100
01	01	00	01	CTL<1:0>	1001
10	01	00	10	CTL<1:0>	0011
00	10	00	00	CTL<1:0>	1000
01	10	00	01	CTL<1:0>	0001
00	11	00	00	CTL<1:0>	0000

A quadword transfer is indicated when CPU address bits <6:3> = 1111. All quadword transfers are unmasked transfers and have all byte enables asserted during all data phases of the PCI transfer. PCI address bits <2:0> are forced to 000 during the PCI address phase. Combinations of CPU address bits <6:3> that are not shown in Table 4-2 or are not equal to 1111 are not supported, and the associated byte enables are unspecified.

For accesses to PCI dense or sparse memory space, regardless of the transfer size, the HPC forces PCI address bits <1:0> to 00 to indicate a linear incrementing burst addressing mode. Accesses to PCI I/O space use a shifted CPU address as the PCI address with CPU address bits <6:5> being used as PCI address bits <1:0> for byte, word, tribyte, and longword accesses. Quadword accesses to PCI I/O space force PCI address bits <1:0> to 00. For accesses to configuration space, the HPC forces bits<1:0> of its CTL CSR into bit <1:0> of the PCI address. Bits<31:2> are the longword or quadword aligned address bits. A value of 00 as PCI address bits <1:0> indicates a type 0 configuration access, while a value of 01 indicates a type 1 configuration access.

4.1.1.3 CPU Command Decode

Masked Transfers

Masked transfers are supported through sparse address space. A masked transfer is indicated when Down Hose command packet address bits <4:3> are 00, 01, or 10. These bits along with address bits <6:5> determine the PCI byte mask. All masked transfers, read or write, are one longword in length. The type of access, read or write, is specified by the R/W bit in the Down Hose command packet. Byte masks are generated for both read and write operations.

Unmasked Transfers

Unmasked transfers are supported through both dense and sparse addressing. Unmasked transfers are indicated when address bits <4:3> of a sparse address access are 11, or during any dense address space access. All PCI byte enables are asserted for these transfers, PCI C_BE L<3:0> = 0000.

Unmasked accesses to sparse space are longword or quadword in length, resulting in a PCI burst length of one or two cycles. Accesses to dense space are quadword or hexword in length, although the masking of entire longwords is possible. Dense space transfers result in a burst length of 2 or 8 longwords being generated on the PCI.

4.1.1.4 CPU Request Queue

The DWLPA can buffer four and DWLPB can buffer six outstanding CPU initiated Down Hose command packets. The Down Hose packets may be Mailbox Command packets, dense command packets, or sparse command packets. When the number of outstanding Down Hose packets reaches the maximum, subsequent command packets can only be sent after a status packet associated with the first command is returned on the Up Hose. Mailbox packets have a Mailbox Status packet returned, dense and sparse commands have a CSR read return packet or a CSR write acknowledge packet returned.

Each DWLPA HPC contains four 8 word x 32-bit buffers used to store CPU write data. Similarly, a DWLPB HPC contains six 8 word x 32-bit buffers. Each HPC also contains one 8 word x 32-bit buffer used to store CPU read return data. These buffers are used to store data for all longword, quadword, and hexword CPU-initiated transactions. Note that revision 1 of the HPC gate array, used on DWLPB (DIAGx register bits <6:3> = 001) has six buffers rather than four for CPU-initiated Down Hose command packets and six return buffers for CSR Write Status Return packets.

4.1.1.5 Requesting PCI Bus

Each HPC contains a PCI arbiter for the PCI bus it interfaces. Each arbiter operates independently though in the same manner. If a Down Hose command is outstanding and a PCI bus is not granted to the HPC, the HPC requests it. Upon receiving a grant from its arbiter, the HPC transfers its command and address to the bus in the cycle following the next idle cycle. The HPC deasserts its request on the cycle it sends the command/address to the bus.

4.1.1.6 Default Bus Parking

When the PCI bus is idle, the HPC is granted the bus even though no Down Hose command packets are outstanding. This condition is referred to as bus parking. If a Down Hose command is received and the bus is parked at an HPC, then the HPC initiates a PCI transfer without asserting a request to its arbiter.

4.1.1.7 Address Phase

During the PCI address phase, the HPC drives the target address on PCI AD<31:0> H and the command on PCI C_BE<3:0> L. Even parity across these signals is driven on PCI PAR H in the next cycle. The HPC also asserts FRAME L indicating that a transfer has been started. If a parity error is detected on the address phase, SERR L is asserted to the PCIA two cycles after the address phase.

4.1.1.8 Data Phase

Longword Reads

In the cycle following the address phase of a longword read, the HPC asserts IRDY L to indicate it can accept data, and it starts sampling TRDY L. FRAME L is also deasserted in this cycle. When TRDY L is sampled asserted, the HPC latches in the read data present on PCI_AD<31:0>. In the next cycle the parity is sampled on PCI PAR, and IRDY L is deasserted to indicate that the transfer is complete. If a parity error is detected, it is reported to the I/O port by setting the error bit in the Up Hose status packet.

Longword Writes

In the cycle following the address phase of a longword write, the HPC drives the write data on to PCI AD<31:0>, the byte mask on to C_BE<3:0> L, and asserts IRDY L indicating that valid write data is on the PCI bus. FRAME L is also deasserted in this cycle to indicate that this is the last data transfer of the burst. The HPC also starts sampling TRDY L in this cycle. When TRDY L is sampled true, the HPC asserts IRDY L to indicate that the transfer is complete. Two cycles after the assertion of TRDY L, the HPC samples PERR L and if it is asserted, the HPC reports a parity error to the I/O port by setting the error bit in the Up Hose status packet. If error interrupts are enabled, an error interrupt is generated to the I/O port.

Quadword Reads

Quadword reads are executed in the same manner as longword reads, except two longwords are transferred. In the cycle following the address phase of a quadword read, FRAME L is not deasserted indicating that at least one more longword of data is to be transferred. For a quadword transfer, FRAME L is deasserted in the cycle following the valid transfer of the first longword. IRDY L and TRDY L are used to handshake each of the two data transfers. IRDY L is deasserted after the second longword is latched from the bus to indicate the transfer is complete. PCI PAR is sampled one cycle after each data transfer, and if a parity error is detected, the HPC reports it to the I/O port by setting the error bit in the Up Hose status packet.

Quadword Writes

Quadword writes are executed in the same manner as longword writes, except two longwords are transferred instead of one. FRAME L is not deasserted until the second longword is driven valid on the bus, and IRDY L and TRDY L are used to handshake each of the data transfers. PERR_L is sampled two cycles after each data transfer, and if it is asserted the HPC reports the parity error to the I/O port by setting the error bit in the Up Hose status packet. If error interrupts are enabled, an error interrupt is generated to the I/O port.

Hexword Reads

Hexword reads are executed in the same manner as longword reads, except eight longwords are transferred. In the cycle following the address phase of a hexword read, FRAME L is not deasserted indicating that at least one more longword of data is to be transferred. For a hexword transfer, FRAME L is deasserted in the cycle following the valid transfer of the seventh longword. IRDY L and TRDY L are used to handshake each of the eight data transfers. IRDY L is deasserted after the last longword is latched from the bus to indicate the transfer is complete. PCI PAR is sampled one cycle after each data transfer, and if a parity error is detected, the HPC reports it to the I/O port by setting the error bit in the Up Hose status packet.

Hexword Writes

Hexword writes are executed in the same manner as longword writes, except eight longwords are transferred instead of one. FRAME L is deasserted when the eighth longword is driven valid on the bus, and IRDY L and TRDY L are used to handshake each of the eight data transfers. PERR L is sampled two cycles after each data transfer, and if it is asserted the HPC reports the parity error to the I/O port by setting the error bit in the Up Hose status packet. If error interrupts are enabled, an error interrupt is generated to the I/O port.

4.1.1.9 Transaction Termination

PCI Master Timeout

Since the maximum HPC burst transfer is 16 longwords and the HPC imposes no initiator delay between successive longwords that it transmits or receives, the HPC does not implement a master latency timer. If implemented, this timer would count the number of cycles since the assertion of FRAME L. If a cycle count limit had been exceeded and the HPC's grant had been removed, then the HPC would be forced to relinquish the bus.

PCI Target Disconnect

The HPC samples the TRDY L and STOP L signals each cycle and releases the bus when a target disconnect is issued by a target PCI device. If a target abort is detected, an error has occurred that prevents the transaction from being retried; the HPC ends the PCI transfer and does not attempt a retry. An error is reported to the I/O port in the Up Hose status packet. If the command is a write and error interrupts are enabled, an error is generated to the CPU.

If a target disconnect is detected and the burst transfer was not completed, the HPC retries the command starting at the address of the next untransferred longword. The HPC retries the command until the data transfer has completed or until the retry limit specified in its RETRY CSR has been reached. If the retry limit is reached and the transfer is not complete, an error is reported to the I/O port in the Up Hose status packet. If the command is a write and error interrupts are enabled, an error interrupt is generated and sent to the CPU.

PCI Master Abort

The HPC aborts any access it initiates if DEVSEL L is not asserted within five cycles after the assertion of FRAME L. A master abort occurs when no PCI device decodes the command and address as residing within its own address space. The HPC reports the error to the I/O port in the Up Hose status packet and generates an interrupt if the command is a write and if error interrupts are enabled.

4.1.1.10 Exclusive Access Support

The HPC as a PCI master does not support locking PCI devices. All completed PCI transactions in which the HPC is a PCI master terminate with the target device unlocked. The HPC supports accesses to locked target devices through the normal PCI retry mechanism.

4.1.2 Device-Initiated PCI Cycles

Device-initiated PCI cycles are transactions originating from a device on the the PCI.

4.1.2.1 Device-Initiated PCI Commands

The PCIA serves as a bridge to system memory for all peripheral devices on the PCI. Commands generated by a PCI peripheral to system memory are decoded and serviced by the HPCs, acting as PCI targets. The HPCs support a subset of the full PCI command set as a PCI target. Table 4-3 shows the PCI commands that are supported.

Table 4-3 Supported PCI Target Commands

PCI Command	C_BE_L<3:0>	Command Aliasing
Memory Read	0110	
Memory Write	0111	
Memory Read Multiple	1100	Aliased to memory read
Memory Read Line	1110	Aliased to memory read
Memory Write & Invalidate	1111	Aliased to memory write

PCI memory reads and writes are directly supported. The PCI memory read line is supported but is executed as a normal memory read. Similarly, the memory write and invalidate command is supported but is executed only as a memory write command. Commands not shown in Table 4-3 are not supported by the HPC as a PCI target, and no action is taken by the HPC when they appear on the PCI bus.

4.1.2.2 PCI Memory Read Multiple Command

When CTLx CSR MRM_EN, bit <21> is clear, PCI Memory Read Multiple on the PCI bus is treated as a PCI memory read. When CTLx CSR MRM_EN (bit 21) is set, PCI Memory Read Multiple on the PCI bus is enabled, CTLx<22>, the Memory Read Multiple Prefetch Size bit in the CTLx register, controls the number of cache blocks the PCIA prefetches in response to a Memory Read Multiple on the PCI bus. If CTLx<22>=0, up to one additional fetch is made. For example, if the cache block size is 64 bytes (determined by the Memory Block Size bit, bit <2>, in CTLx), the reads issued on the Up Hose are:

DMA_ADDR<7:6> (CTLx<22> = 0)	**Blocks Fetched for DMA_ADDR<7:6>**
00	00, 01
01	01
10	10, 11
11	11

If CTLx<22>=1, up to three additional fetchs are made. For example, if the cache block size is 64 bytes (determined by the Memory Block Size bit, bit <2>, in CTLx), the reads issued on the Up Hose are:

DMA_ADDR<7:6> (CTLx<22> = 1)	**Blocks Fetched for DMA_ADDR<7:6>**
00	00, 01, 10, 11
01	01, 10, 11
10	10, 11
11	11

The MRM_ARB bit, CTLx<20>, controls the number of Up Hose DMA read transactions that a HPC drives on the Up Hose bus of the PCIA in one Up Hose bus tenure. (The Up Hose bus is shared by 3 HPC's on the PCIA. Normally an HPC requests tenure on the Up Hose bus, gets tenure from the up hose arbiter, completes one Up Hose transaction, and then surrenders the bus.) When MRM_EN is enabled, and MRM_ARB = 0, the HPC drives out 1, 2, 3, or 4 Up Hose DMA transactions. When MRM_EN is enabled, and MRM_ARB = 1, prefetching is disabled and only one transaction per Up Hose bus tenure is allowed.

In addition to the control bits just described, one status bit is provided. The INC_LTO bit, ERRx register bit <17>, is set during the execution of a Memory Read Multiple if the number of PCI bus data wait states between data cycles is greater than the PCI limit of 8 cycles. Detection of this condition does not result in an error interrupt generated to the CPU. If INC_LTO, Incremental Latency Timeout is zero, no timeout has been detected. If INC_LTO is one, a timeout has been detected.

4.1.2.3 PCI Address Decode

The PCIA provides three address windows, A, B, and C, through which PCI devices can access system memory. The PCIA decodes 32-bit PCI addresses and services supported target commands if its PCI address falls within any of the PCIA's address windows. Addresses that do not fall in an address window are not serviced by the PCIA since they should decode to a PCI peripheral device's memory space.

The base PCI address of each window is defined in its Window Base Register in each HPC and falls on an address boundary naturally aligned to its window size. Window sizes of 64 Kbytes to 4 Gbytes are supported and are selected through each window's Window Mask Register of each HPC. See Chapter 3 for a description of each register.

Each PCI address window can be individually enabled or disabled by bit <1> of its Window Base Register. When disabled, all accesses addressed to the window are not serviced by the HPC. The type of translation performed on the address, direct or scatter/gather, is selected by bit <0> of the Window_A Base Register.

4.1.2.4 PCI Address Translation

Direct Mapped Address Translation

Direct address translation sets up a window in system memory of the same size that is used in PCI memory space. The window can be located anywhere in system memory address space as long as it starts on an address boundary that is naturally aligned to its window size. The base address of the window in system memory is defined in the Translated Base Register in each HPC. Each PCI address window can be independently mapped to its own address window in system memory.

Scatter/Gather Address Translation

Scatter/gather translation uses an address mapping RAM to translate the 32-bit PCI address into a 40-bit system memory address. Figure 4-1 shows the scatter/gather address translation implemented by the PCIA.

For DWLPA or DWLPB with TBIT=0, bits <27:13> of the PCI address are used to index into the map RAM. Map RAM data bits <27:1>, the translated page number, are then merged with PCI address bits <12:0> to generate a 40-bit DMA system memory address. The map RAM parity is odd. For DWLPB with TBIT=1, the PCI address bits used are <29:13>.

Scatter/Gather Cache

Each HPC maintains a fully associative five-entry cache of the map RAM. This cache provides lower latency translation than is available through physical RAM access. Each cache entry contains a 15-bit tag, a 27-bit page number, a valid bit, and a parity bit. The contents of the tag are compared against bits <27:13> (DWLPA, and DWLPB with TBIT = 0) or bits <29:13> (DWLPB with TBIT = 1) of received PCI DMA addresses. The cache fill algorithm is random.

Cache entries are invalidated when a map RAM CSR write is detected to an entry in the scatter/gather cache. Writes to a map RAM entry should not occur while a DMA transaction using that entry is in progress, since no attempt is made to stop a DMA transaction in progress.

Map RAM Parity Errors

Access to the map RAM during a DMA read that results in a parity error causes the HPC to abort the transaction. The HPC issues a target abort on the PCI bus and sets bit <7> of its ERR CSR. An error interrupt is generated to the CPU if the Error Interrupt Enable bit is set in the HPC's IMASK register.

Access to the map RAM during a DMA write that results in a parity error does not cause the HPC to abort the transaction. Instead the HPC continues receiving write data and drops the transaction and sets bit <7> of its ERR CSR. An error interrupt is then generated to the CPU if the Error Interrupt Enable bit is set in the HPC's IMASK register. Software must force a system crash if a map RAM parity error is detected during a DMA write transaction, since the write is disconnected from its PCI master.

Map RAM Invalid Page Errors

Access to the map RAM during a DMA read that results in the valid bit not being set also causes the HPC to abort the transaction. The HPC issues a target abort on the PCI bus and sets bit <8> of its ERR CSR. An error interrupt is sent to the CPU if the Error Interrupt Enable bit is set in the HPC's IMASK register.

Access to the map RAM during a DMA write that results in the valid bit not being set does not cause the HPC to abort the transaction. Instead the HPC drops the transaction and sets bit <8> of its ERR CSR. An error interrupt is sent to the CPU if the Error Interrupt Enable bit is set in the HPC's IMASK register. Software must force a system crash if a map RAM parity error is detected during a DMA write transaction, since the write is disconnected from its PCI master.

4.1.2.5 Address Queuing

The HPC can queue up to two PCI DMA transactions per PCI bus if the first transaction stored is a DMA write. The second transaction stored can be a DMA read or a DMA write. This allows a DMA write transaction already received on one PCI bus to be transmitted on to the Up Hose while a second DMA transaction is being received on the same PCI bus. Since PCI reads are nonpended, no additional DMA transactions can be received over the bus until the read data is returned to the PCI master. If the HPC's DMA queue is full, the HPC continues to decode all subsequent incoming PCI addresses and issues a PCI target disconnect if it is the target of a transaction.

4.1.2.6 Address Phase

The HPC samples FRAME L to detect a PCI address phase. When FRAME L is asserted, the PCIA latches PCI AD<31:0> H and C_BE<3:0> L. In the next cycle, the HPC decodes the PCI command and address and determines if the access is to system memory; it also samples PAR during this cycle. If the access is to system memory, the HPC asserts DEVSEL_L to the PCI bus two cycles after the assertion of FRAME L.

4.1.2.7 Address Burst Order

Only linear incrementing burst order addressing is supported and is indicated by bits <1:0> of a PCI memory address being equal to 00. An HPC terminates the burst when either the transaction is complete or a memory block boundary is reached, whichever comes first.

If PCI memory address bits <1:0> are not equal to 00, then the HPC terminates the transaction after one longword of data has been transferred. This will cause the burst to be broken up into multiple single longword transfers with the PCI master generating a new longword address for each transfer.

4.1.2.8 Memory Block Boundary

The HPC target disconnects any DMA transaction that crosses a memory block boundary. The size of the memory block boundary is selected by HPC CTL CSR bit <2>. Memory block sizes of 32 and 64 bytes depends upon the state of the Memory Block Size bit, bit <2>, in the CTLx register. Target disconnects are done at memory block boundaries to meet target latency timeout restrictions of the PCI bus. If prefetching is enabled for the PCI Memory Read Multiple command, target disconnects occur at two times or four times the cache block boundary for memory block sizes of 64 bytes or 32 bytes. Target disconnects at 2x or 4x is set by the state of CTLx<22>.

4.1.2.9 Address Parity Errors

If an address parity error is detected on a DMA read or write transaction, the HPC asserts SERR_L and issues a target abort on the PCI bus. No data is written to system memory. The HPC sets bit <8> of its ERR CSR and if error interrupts are enabled, an error interrupt is sent to the CPU.

4.1.2.10 Write Data Transfers

DMA write transfers are completely received by the HPC before being sent on to the Up Hose. All write transfers are limited in maximum size to the memory block size. If a 64-byte memory block is used and all bytes of the block are valid, the PCIA issues an unmasked double hexword write transaction to system memory. If a 32-byte memory block size is selected or if any of the byte enables of the selected 64-byte block is not set, a masked write transaction is issued to system memory. Masked writes result in Read-Modify-Write operations to system memory and thus consume twice the memory bandwidth. The HPC optimizes the size of the Up Hose DMA write packet based on the byte enable bits to reduce the amount of Up Hose bus bandwidth required. Reduced Up Hose packet transfer sizes of 32- and 16-byte transfers are supported. A single Up Hose DMA write is generated for each separate DMA write transaction on the PCI bus. Multiple PCI write transactions are not packed into a single Up Hose packet.

PCI Write Data Buffers

Each HPC contains two 16-word by 32-bit data buffers for storage of PCI write data. Also contained along with each buffer is a 64-bit register used to store the associated byte enable bits. PCI write data parity is checked as it is received from the PCI bus. Parity is regenerated over the Up Hose command and data when it is sent to system memory.

The HPC write data buffers are capable of receiving a longword every cycle from the PCI. Once both data buffers have been filled, the PCIA target disconnects any DMA transaction received on the PCI bus.

Write Data Errors

If a write data parity error occurs during a DMA transaction, the HPC does not issue a target abort of the transaction on the PCI. Instead the HPC continues to receive data until the transaction is complete or a memory block boundary is reached and then drops the transaction and sets bit <10> of its ERR CSR. An error interrupt is generated to the CPU if error interrupts are enabled. The PCI address of the failing DMA transaction is also stored in the HPC's FADR CSR.

Map parity or invalid entry errors also do not affect the reception of write data. The packet is dropped and bit <6> or <7> of the HPC's ERR CSR is set to indicate an error. The PCI address of the failing DMA transaction is stored in the HPC's FADR CSR. If error interrupts are enabled, an error interrupt is generated.

4.1.2.11 Read Data Transfers

The HPC, upon receiving a DMA read request from the PCI, requests a 32-byte or a 64-byte read from system memory. The size of the read request is the same as the memory block size as defined in each HPC's CTL CSR. IREAD transactions (see Section 4.1.2.13) request a 16-byte read from system memory. When returning data to the PCI bus, the HPC is capable of bursting a longword of data to the PCI in consecutive cycles until the transaction is complete.

All read transactions that start at the beginning of a naturally aligned 64-byte block have their read return data returned to the PCI in a cut-through fashion: the data is returned to the PCI while the HPC is still receiving data over the Down Hose. This is possible because the first longword of data in the Down Hose Read Data Return packet is always valid, and the associated delay to the first valid longword is not variable. Read transactions that are not 64-byte naturally aligned return data in a store and forward fashion.

Since the PCIA interfaces to Down Hoses that can run at different clock speeds, the HPCs provide a cut-through threshold field in the CTL CSR. Bits <7:4> of this register define the number of longwords that the HPC will receive and load into its read return buffer before transmitting any data to the PCI bus. Cut-through operation can be disabled by clearing the Cut-through Enable bit, bit <8>, of the CTL CSR.

When cut through is enabled, CTLx register bit <8> is set, and CTLx register bits <7:4> should be initialized according to the following equation:

$$(a * d - a * p) = x \text{ (rounded up)}$$

where x = the setting. If x is a non-integer, then round up.
d = Down Hose clock cycle time.
p = PCI clock cycle time.
a = 8 if memory block size = 32 bytes.
a = 16 if memory block size = 64 bytes.
(Memory block size is controlled by bit <2> of the CTLx register.)

Example 1: a = 16 for memory block size = 64 bytes.
p = 30 ns
d = 30 ns
then x = 0x0 and bits <7:4> = 0000

Example 2: a = 16 for memory block size = 64 bytes.
p = 30 ns
d = 40 ns
x = 5.33333
then x = 0x6 after rounding up and bits <7:4> = 0110.

PCI Read Return Buffer

Each HPC contains a 16-word by 32-bit dual port data buffer used to store a double hexword of read return data received on the Down Hose bus. Read data parity is checked as it is received from the Down Hose bus. Parity is regenerated over the PCI data and PCI byte enables when the data is returned to the PCI bus. The PCI read return buffer is not used as cache for subsequent DMA read transactions on the PCI bus.

Read Data Errors

If a map parity or invalid page error is detected during the translation of the DMA read address, the HPC issues a target abort to the PCI master and terminates the transaction. The HPC sets bit <7> or <6> of its ERR CSR and generates an error interrupt if error interrupts are enabled. The PCI address of the failing DMA transaction is stored in the HPC's FADR CSR.

If a Down Hose parity error is detected during the reception of read return data from the I/O port the HPC target aborts the transaction. The HPC sets bit <5> of its ERR CSR and generates an error interrupt if error interrupts are enabled. The PCI address of the failing DMA transaction is stored in the HPC's FADR CSR.

If a Down Hose Read Data Return packet with an error is received by an HPC, it issues a target abort to the PCI master and terminates the transaction. The HPC does not set any error bit since the error occurred upstream in system memory or in the I/O port module and is flagged there.

4.1.2.12 PCI Target Latency Timeout

The HPC is capable of returning read data to the PCI in consecutive cycles up to the end of a memory block at which time a target disconnect will be issued if necessary. Since the HPC does not introduce any delay between data transfers, no target latency timer is implemented.

4.1.2.13 Device-Initiated Exclusive Access

Since the PCIA contains three PCI buses that operate independently, a PCI device is not capable of locking out PCI devices on another PCI bus from system memory. In addition, LSB and TLSB systems do not provide a mechanism for locking out CPUs from system memory when a PCI lock is established. As a result, the PCIA does not implement the PCI lock function, and an HPC as a target does not lock when it receives a lock request from a PCI device.

The HPC does provide a limited lock mechanism to support the VAXport I/O architecture. When an HPC detects the assertion of the PCI LOCK signal in the cycle after a PCI address phase and the transaction type is a

DMA read and the HPC is the target of the transaction, it issues an IREAD command over the Up Hose to the I/O port for that transaction. The HPC does not generate any additional IREAD commands while PCI LOCK is asserted. PCI LOCK must be deasserted and then reasserted to generate another Up Hose IREAD transaction. See Chapter 6 for more information on EISA bus locks.

4.2 Hose Interface

The hose interface consists of two parts and supports data transfers between the PCIA and the I/O port.

The I/O port provides the interface to the CPU and system memory. A Down Hose interface transfers data from the I/O port to the PCIA. An Up Hose interface transfers data from the PCIA to the I/O port.

4.2.1 Down Hose

The Down Hose can transfer 32 bits of data each cycle to the PCIA. Data is latched by a PCIA module transceiver and is then bused to the HPCs over the DN_BUS<31:0> L signals. Each longword of data is covered with odd parity. Parity error detection is performed by logic in each HPC.

4.2.1.1 Down Hose Commands

The PCIA accepts Down Hose commands and associated packets from the I/O port. The Mailbox Command packet, the Dense CSR Command packet, and the Sparse CSR Command packets are CPU-initiated commands. The DMA Read Data Return packet is a response to a DMA read transaction, and the Interrupt Status Return packet is a response to an Up Hose INTR/IDENT packet. Table 4-4 lists the supported Down Hose packet types and shows the Down Hose header cycle bits that determine the packet type.

Table 4-4 Down Hose Commands

DND<13:12>	DND<1:0>	Down Hose Command Packet Type
10	xx	Mailbox Command
11	00	Dense CSR Command
11	01,10,11	Sparse CSR Command
01	xx	Read Data Return Packet
00	xx	Interrupt Status Return Packet

4.2.1.2 Down Hose to PCI Synchronization

All Down Hose commands sent by the I/O port are synchronous to the Down Hose clock, DNCLK H. Each HPC loads the appropriate Down Hose command, address, and data into its buffers synchronous to DNCLK H. The command is then synchronized to the PCI clock.

4.2.1.3 Down Hose Signals

Table 4-5 lists the 38 Down Hose signals.

Table 4-5 Down Hose Signals

Signal	Signal Count	Description
DND<31:0> L	32	Down Hose data
DNP L	1	Down Hose parity
DNDATAVAL L	1	Down Hose data valid
DNCLK H	1	Down Hose clock
DECPKTCNT L	1	Decrement packet count
DNRST L	1	Down Hose reset
ERROR L	1	Fatal Error

4.2.2 Up Hose

The Up Hose transfers 32 bits of data each cycle to the I/O port. Data is driven from each HPC over the UP_BUS to a module transceiver that drives the data on to the Up Hose. Command and byte mask bits are also driven with the data along with odd parity covering all signals.

4.2.2.1 Up Hose Commands

DMA commands are initiated by PCI peripherals. CSR commands and the Mailbox Status command are responses to CPU-initiated CSR transactions. The INTR/IDENT command can be initiated by an HPC for an error interrupt or by a PCI peripheral for a device interrupt. Supported Up Hose commands are listed in Table 4-6.

Table 4-6 Up Hose Commands

UPCTL<3:0>	Up Hose Command Packet Type
0001	DMA Read
0010	IREAD
0100	Mailbox Status Return
0101	DMA Unmasked Write
0111	DMA Masked Write
1000	INTR/IDENT
1100	CSR Write Status Return
1101	Dense CSR Read Return
1110	Sparse CSR Read Return

Flow control between the I/O port and the PCIA is maintained across all three PCI buses during DMA and INTR/IDENT transactions by signaling to each HPC the status of the current Up Hose command. Two signals, UPSTS<1:0>, are sent to each HPC indicating that status. The signals are driven by a module PAL that decodes UPCTL<3:0>. UP_BUS<31> of INTR/IDENT cycle is used to indicate to the PAL whether the interrupt is a device or error interrupt. Table 4-7 describes the Up Hose Command Status packets. Appendix A describes each of the Up Hose packet structures.

Table 4-7 Up Hose Packet Status

UPSTS<1:0>	Description	Action
00	No command in progress	None
01	CSR/DMA/MBX in progress	Increment packet count.
10	Device interrupt in progress	Increment packet count. Set device interrupt outstanding.
11	Error interrupt in progress	Increment packet count. Set error interrupt outstanding.

Flow control is also maintained between the I/O port and the PCIA on CSR and mailbox transactions by providing the Mailbox Status Return, CSR Read Return, and CSR Write Status Up Hose commands. Before any of these status packets are sent over the Up Hose, all HPCs are synchronized to ensure that storage exists in each HPC for another CPU-initiated Down Hose command.

Synchronization is maintained by requiring each HPC to generate and receive three open drain signals, CMD_OUT L, CMD_DEC_OUT L, and CMD_DS L. All HPCs accept the next Down Hose command when they sample CMD_OUT L as deasserted. Upon starting command execution, each HPC asserts CMD_OUT L to indicate that a command is outstanding and currently being executed. No subsequent Down Hose commands are executed until CMD_OUT L is deasserted for one cycle by all HPCs. If an HPC finishes its command execution before CMD_OUT L is deasserted, it enters a wait state.

CMD_DEC_OUT L is driven low by all HPCs to indicate that the HPC has not yet completed a decode of the command. CMD_DEC_OUT L is deasserted by each HPC when the decode is completed. The HPC that decodes and executes the command asserts CMD_DS L after it finishes the decode. If CMD_DS L is asserted when CMD_DEC_OUT L is deasserted, one HPC has decoded the command and responds by sending the return status packet to the I/O port. If CMD_DS L is deasserted when CMD_DEC_OUT L is deasserted, none of the HPCs has decoded the command as its own and HPC 0 returns error status to the I/O port. If the command is a write and error interrupts are enabled, an interrupt is generated to the CPU.

The HPC that returns the command's Up Hose status return is the last HPC to deassert CMD_OUT L. When an HPC determines it will not execute a command, it deasserts its CMD_OUT L immediately after the decode. However, if no other HPC claimed the command, HPC 0 as the default keeps CMD_OUT L asserted. After status is returned and CMD_OUT L is deasserted for one cycle, all HPCs start executing the next command. Table 4-8 describes the function of CMD_DEC_OUT L and CMD_DS L.

The HPC gate array allows back-to-back CSR write dense packets on the Down Hose and the PCI bus to complete on the PCI bus. Up to six CSR dense write status packets can accumulate among the three HPC's Up

Hose CSR dense write status packet queue. This queue empties its contents onto the Up Hose.

Here, CMD_OUT deasserts when a CSR dense write transaction completes on the PCI bus and another CSR dense write transaction is in the Down Hose queue waiting for service. CMD_OUT release tells:

- the three HPC PCI bus initiator state machines on the PCIA to start servicing the next CPU-initiated transaction.
- the Down Hose queue control that one CSR command was just serviced so one Down Hose CSR buffer is freed for reuse.

Table 4-8 Command Signal Assertion

CMD_DEC_OUT L	CMD_DS L	Description
0	0	MBX or CSR command decode outstanding - no device selected
0	1	MBX or CSR command decode outstanding - device selected
1	0	MBX or CSR command decode done - no device selected
1	1	MBX or CSR command decode done - device selected

4.2.2.2 Up Hose Arbitration

Each of the HPCs must arbitrate for the Up Hose before sending a command on to it. An HPC starts the arbitration process by issuing a request for the UP_BUS to its Up Hose arbiter. When the UP_BUS is available, the arbiter grants it to the requesting HPC. Arbitration is round robin with each HPC having equal priority.

An HPC may delay its request of the Up Hose based on the status of previously transmitted Up Hose command packets. If an HPC has a MBX/CSR, DMA, or INTR/IDENT transfer to send on to the Up Hose and the I/O port has no buffers available to store the packet, the request is delayed until DECPKTCNT_L is asserted on the Down Hose. If an INTR/IDENT is outstanding and an HPC has an interrupt at that IPL, the HPC delays the request until an INTR status packet is received on the Down Hose.

4.2.2.3 Up Hose Command FIFO

Each HPC contains a 4-entry FIFO used to determine the order in which commands are sent on to the Up Hose. DMA and MBX/CSR Return Status commands received from the PCI are loaded into the FIFO in the order in which they are received from the PCI bus. MBX/CSR Return Status commands are executed in the HPC rather than on the PCI bus and are loaded into the FIFO at the time that the command completes.

An interrupt command is loaded into the FIFO at the time it is received by the HPC if it is the highest priority interrupt that is outstanding. No other interrupts are loaded into the FIFO until the first interrupt is sent over the Up Hose. If the interrupt reaches the top of the FIFO and another interrupt is outstanding at the same IPL in the I/O port, the interrupt is stalled until an interrupt status packet for that IPL is received. DMA and

MBX/CSR transfers can be sent over the Up Hose while an interrupt is stalled.

4.2.2.4 Up Hose Signals

The Up Hose consists of the 44 signals listed in Table 4-9.

Table 4-9 Up Hose Signals

Up Hose Signal	Signal Count	Description
UPD<31:0>	32	Up Hose data
UPP L	1	Up Hose parity
UPDATAVAL L	1	Up Hose data valid
UPCLK H	1	Up Hose clock
UPCTL<3:0> L	4	Up Hose control
UPRST L	1	Up Hose reset
CBLOK L	1	Cable OK
PWROK L	1	Power OK
UP_ERR_IN L	1	Fatal error in
UP_ERR_OUT L	1	Fatal error out

4.2.2.5 Map RAM Structure

The map RAM consists of 32K/128K entries of address translation pages used during DMA transactions that require scatter/gather translation. The map RAM hardware consists of four SRAMs, an address latch, a data transceiver, and a read/write control PAL. The map RAM hardware interfaces to the HPCs by the UP_BUS<31:0> signals. A map RAM entry's parity is stored in the map RAM array also.

The HPCs use the UPCTL<3:0> L and UPD<29:28, 26:25> signals to indicate the type of command, read or write, and the length of the command, longword or quadword. An HPC asserts its MPDATAVAL L output signal in the cycle that the address is driven to the map RAM logic. UPDATAVAL_L is not asserted to the I/O port during map RAM accesses. The map RAM can be read by any HPC during a DMA transaction. CPU-initiated reads and writes to the map RAM are performed by HPC 0 during mailbox commands or CSR sparse space commands.

The map RAM logic also contains the PRESENT register. This register is used to determine if a PCI slot is populated and the type of card, high power or standard power, that resides in the slot as well as the module revision number. This register is read only and each PCI card contributes two bits to the register. One UPCTL code is used when accessing the PRESENT register. See Chapter 3 for a detailed bit description of the PRESENT register.

The map RAM logic also contains the MCTL register. This register is written upon the completion of power diagnostic code. Writing a one to bit <0> of the MCTL register will turn on the MCTL passed LED. This register is write only, and one UPCTL code is used when writing the MCTL register. See Chapter 3 for a detailed bit description of the MCTL register.

Table 4-10 shows the map RAM control codes used during map RAM and PRESENT and MCTL register accesses.

Table 4-10 Map RAM Logic Command

Map Ram Logic Command	UPCTL<3:0>	UPD<29:28>	UPD<26:25>
Map RAM Read - Quadword	0010	00	00
Map RAM Write - Quadword	0011	00	00
Map RAM Read - Longword	0000	00	00
Map RAM Write - Longword	0001	00	00
Serial EEPROM CSR Read	0000	00	01
Serial EEPROM CSR Write	0001	00	01
Present Reg Read	0000	00	10
MCTL Reg Write	0001	00	10

4.2.2.6 Special Programming Considerations for the Up Hose

Programming the I/O Port Up Hose Buffers field, bits <24:23>, of the PCIx Control Register (CTLx) requires special consideration. The hose protocol requires that the PCIA know how many Up Hose buffers exist at the other end of the hose. This number is fixed at three for the I/O port, but may be smaller in other implementations. By default, the PCIA assumes that only one buffer is available. On TLSB systems, the HPCs must be reprogrammed to know that three Up Hose buffers are available.

This PCIA default on TLSB systems presents a complication because:

1. Three separate Down Hose transactions are required to reprogram the three HPCs on the PCIA.
2. The HPCs cooperate to count the number of Up Hose buffers in use.
3. The HPCs report a fatal error if they detect an overflow or underflow count.

Therefore, until all HPCs are programmed to the new value, it it important to ensure that only one Up Hose packet ever be pending from the PCIA. The following sequence will maintain that condition and must be used when reprogramming the I/O Port Up Hose Buffers field:

1. Write CTL0 (the CTLx register of HPC0) with the desired new value. HPC0 will send a Write Status Return packet after completing the write.
2. Execute an MB instruction to ensure that the write and read are ordered.
3. Read CTL1 (the CTLx register of HPC1).
4. Write CTL1 with the desired new value that must be the same as the one written to CTL0.
5. Execute an MB instruction to ensure the write and read are ordered.
6. Read CTL2.
7. Write CTL2 with the desired new value that must be the same as the one written to CTL0 and CTL1.

4.2.3 Serial EEPROM

The PCIA has 2 Kbytes of serial EEPROM used to store console code and field repair data. The EEPROM has its data and clock driven by software through the module serial EEPROM CSR.

4.2.4 PCI to EISA Bridge Hardware

The KFE70 module provides connectivity to eight EISA slots on the PCIA module. The KFE70 uses the INTEL PCI-EISA Bridge Component (PCEB) and the EISA System Component (ESC) chipset, also known as the Mercury chipset. The PCEB is a fully buffered bidirectional interface between the PCI bus and the EISA bus. The PCEB also offers PCI arbitration logic which is not used by the PCIA. The ESC adds EISA bus control and arbitration. The equivalent of two 82C59 interrupt controllers, two 82C54 timers, DMA scatter/gather support, and XBus control round out the chipset's capabilities. Refer to Chapter 6 for additional information on the standard I/O module, the PCI-EISA bridge module.

4.2.5 CPU-Initiated Transactions

All Down Hose mailbox read/writes, sparse CSR read/writes, and dense CSR read/writes are executed in a similar manner. Each of the three HPCs contains the logic to execute the Down Hose command independently. The transaction descriptions below describe the operation of one HPC.

Down Hose read/writes start when the HPC's Down Hose receiver state machine decodes the command and address. If the command is a write, data is written into the CPU write RAM. The CPU write RAM is used to store longword, quadword, and hexword data. After synchronization to the PCI clock, the command is executed differently depending on its destination. Destinations are the PCI bus, an HPC register, an HPC RAM location, the map RAM, and the PRESENT or MCTL CSRs.

Writes to the PCI bus start by the HPC arbitrating for the PCI bus. When a grant is received, the HPC starts an address phase by transmitting the PCI command and address on to the bus. Write data is read from the CPU write RAM and transferred to the bus. The HPC's PCI master state machine continues to get data from the PCI write RAM and transfers it to the bus until all the data has been transferred. Reads from the PCI bus are similar to writes, up to the data phase. During a read data phase, the target drives data on to the bus and the HPC latches it and writes it into the CPU read return RAM.

Writes to an HPC CSR register are executed by reading a longword of data from the CPU write RAM and then loading it into the selected register. Writes to an HPC CSR RAM location are executed similarly with the data loaded by RAM write. The HPC supports unmasked longword and quadword CSR writes only. Byte and word writes to CSR registers and RAM are not supported. CSR register reads are executed by loading the selected CSR's data into the CPU read return RAM while an HPC CSR RAM read first reads the RAM data and then writes it to the CPU read return RAM. Longword and quadword reads are supported to HPC CSR registers and RAM.

All accesses to the map RAM and PRESENT or MCTL registers are performed by HPC 0, which is defined by both of its HPC ID<1:0> H pins being driven low. Map RAM reads and writes and PRESENT or MCTL register reads and writes are executed by first arbitrating for the UP_BUS. When a grant is received, the map address and command (command only if a PRESENT register read or MCTL write) are driven on to the UP_BUS for two cycles. If the command is a map RAM write, the write data is driven on the UP_BUS in the next cycle. Map RAM control logic external to the HPC write the data into the selected map RAM location. The PCIA supports unmasked longword and quadword map RAM writes only. Byte and word writes to the map RAM are not supported and are converted to longword writes. Only longword writes to the MCTL register are supported.

Map RAM reads and PRESENT register reads are executed by first driving the command and address on to the UP_BUS. Longword and quadword reads of the map RAM are supported. Only longword reads of the PRESENT register are supported. Quadword reads will return the PRESENT register data in both longwords of the status packet. In the cycle following the last address cycle on the UP_BUS, HPC 0 tristates its drivers. The map control logic drives read data on to UP_BUS<31:0> three cycles after the command was present on the UP_CTL <3:0> signals. The HPC then writes the data into the CPU read return RAM.

The outstanding MBX/CSR command is loaded on the Up Hose FIFO and when it reaches the top, the HPC arbitrates for the UP_BUS to return the Up Hose status packet. All write commands regardless of the destination return a status packet to the I/O port by the Up Hose after all data has been transferred to/from the destination. This is done to maintain flow control with the I/O port and avoid having data overwritten with new Down Hose write data. All reads that have data stored in the CPU read return RAM return the data in an Up Hose status packet.

4.2.6 DMA Transactions

DMA transactions start with the HPC's target state machine sampling the PCI bus for an address phase. When an address phase is detected, the address is latched and decoded to determine if system memory is the target of the transaction. If system memory is the target and the transaction is a write and if there is a DMA write buffer available, data is written into the buffer with a target disconnect occurring at a memory block boundary if necessary.

For accesses to an address window that is direct mapped, the HPC uses the TBASE CSR contents as the upper DMA address bits as described earlier in this chapter and in Chapters 2 and 3.

For accesses to an address window that has scatter/gather address mapping enabled, the address is compared with all the entries in the scatter/gather cache and if a match is found, the cache data is used as the translation page. If a miss occurred, the map RAM index is extracted from the DMA address and a request for the UP_BUS is made. When a grant is received, the map RAM index and a map RAM read command are driven on to the UP_BUS. The map RAM entry is latched from the UP_BUS and is used as the translation page. The latched map RAM entry and the map RAM index are also loaded into the cache. When the entire DMA command has been fully received from the PCI bus, it is loaded in to the Up

Hose command FIFO where it generates an Up Hose request when it reaches the top of the FIFO.

Upon receiving an UP_BUS grant during a DMA write, the direct or translated command/address is driven on to the bus followed by a longword of write data and byte mask each cycle. The Up Hose DMA address is determined by the PCI DMA address and the size of transfer. The Up Hose DMA address is generated from the PCI address such that a minimum size, single Up Hose packet transfer occurs.

A DMA read will similarly drive the Up Hose command and address on to the UP_BUS. The Up Hose DMA read address is the address of the memory block being accessed. All DMA reads sent to the Up Hose bus are the length of the memory block size. The tag field of the DMA command is loaded with the HPC ID code so that the Read Data Return packet can be detected by the HPC. Read return data is received on the Down Hose bus and is latched into the HPC's PCI read return RAM. When the cut-through threshold is reached, data is returned to the PCI bus up to the memory block boundary, where a target disconnect will occur if necessary.

4.3 PCIA Interrupts

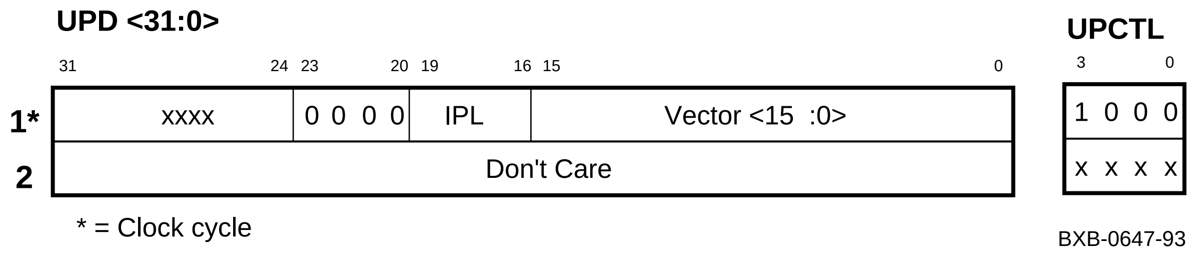
The PCIA supports 48 hardware PCI device interrupts, and it generates error interrupts for PCI bus errors and PCIA-generated errors. Each HPC contains the interrupt handling logic for its PCI bus and executes the interrupt requests independent from the other HPCs. All EISA interrupts are reported by PCI interrupts from the PCI-to-EISA bridge. The bridge module contains logic that combines and prioritizes EISA interrupts into one PCI interrupt.

4.3.1 PCI Interrupts

4.3.1.1 PCI Device Interrupts

The PCIA supports only one interrupt priority level (IPL) for hardware device interrupts. The device IPL is defined in each HPC's IMASK register, bits <20:17>, and is initialized by software. Bits <20:17> must be set to the same value in each of the three HPCs. A PCI device interrupt when driven to an HPC is converted to an Up Hose INTR/IDENT packet. The format of the packet is shown in Figure 4-1.

Figure 4-1 INTR/IDENT Packet



4.3.1.2 PCI Error Interrupts

The PCIA also supports only one IPL for error interrupts. The error IPL is also defined in each HPC's IMASK register <24:21> and is initialized by software. The error IPL, bits <24:21>, must be set to the same value in each of the three HPCs and must not be set to the same value as the device IPL bits <20:17>. Error interrupts are sent over the Up Hose as INTR/IDENT packets. The IPL for error interrupts must be set higher than the IPL for hardware device interrupts.

4.3.2 Hose Interrupt Support

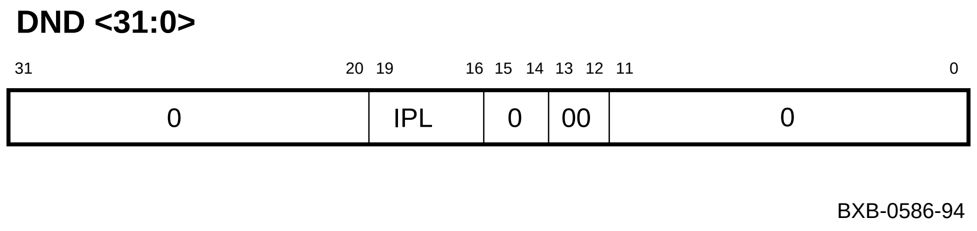
The hose interface provides support for transferring interrupts to the CPU. When the HPC receives a hardware device interrupt, it merges the selected interrupt vector and the IPL level into an INTR/IDENT packet as shown in Figure 4-1. The HPC then inserts the interrupt on an Up Hose command queue. The command queue is processed FIFO, except for error interrupts which have priority. When the interrupt reaches the top of the queue, the UP_BUS is requested if no outstanding INTR/IDENT of the same IPL already exists. When a grant is issued to the HPC, the

INTR/IDENT packet is then sent to the I/O port and status of the outstanding interrupt is sent to each of the HPCs by the UPSTS<1:0> L signals. Bit <31> of the first longword of the INTR/IDENT is set if it is an error interrupt and it is clear if it is a device interrupt. This bit is used by the PCIA module only and is a don't care to the I/O port.

No attempt is made to prioritize hardware interrupts between the HPCs. Up Hose arbitration is used to determine the order of interrupt servicing between PCI buses. When one HPC sends a device interrupt up the hose, the other HPCs are blocked from sending device interrupts until an interrupt status packet is decoded on the Down Hose. To prevent an interrupt from being delayed indefinitely, an HPC that has an interrupt ready for the UP_BUS will count the number of times it fails in an attempt to send an interrupt. When it counts 16 failures, it asserts a lockout signal (HPC_ILCKO L) to the other HPCs. Once the lockout signal is asserted, an HPC will issue no more than one device interrupt and one error interrupt on the Up Hose before ceasing to arbitrate for the UP_BUS. This mechanism allows the "locked out" HPC to successfully arbitrate for the bus, send its interrupt packet, and deassert the lockout signal. Note that the lockout may be asserted simultaneously by more than one HPC.

When an Interrupt Status Return packet is received from the I/O port, it is sent to each HPC. This provides the flow control that allows the PCIA to send another INTR/IDENT packet over the Up Hose at the same IPL. The format of the Interrupt Status Return packet is shown in Figure 4-2.

Figure 4-2 Interrrupt Status Return Packet



4.3.3 Vectored Interrupts

4.3.3.1 PCI Device Interrupts

Each of the four PCI I/O cards on a PCI bus segment can generate four PCI device interrupts; thus, a total of 16 PCI device interrupts can be received from one bus segment. The HPC latches any interrupt signal and stores it in one of 16 interrupt-pending flops.

The pending flop outputs are prioritized to determine which interrupt to service. An interrupt's priority is determined by the PCI slot of the card, and the priority of the interrupts on that card. Slot 0 of a PCI bus generates the highest priority hardware interrupts, while slot 3 generates the lowest. PCI INTA is the highest priority interrupt on a card while PCI INTD is the lowest priority interrupt. Table 4-11 shows the interrupt priority scheme for interrupts from a single PCI bus.

On physical PCI segment 0, four of the HPC interrupt inputs are also connected to interrupts from PCI devices on the standard I/O module, as described in Chapter 6. When the standard I/O module is installed, its devices have the highest priority on PCI 0.

Table 4-11 PCI Interrupt Priority with and without the Standard I/O Module

PCI Slot without Standard I/O	INT without Standard I/O	PCI Slot with Standard I/O	Priority
0	A		Std I/O NMI Highest
0	B		Std I/O EISA bridge
0	C		
0	D	Not used	
1	A	Not used	
1	B	Tulip	
1	C	Not used	
1	D	Not used	
2	A	Not used	
2	B	Not used	
2	C	Not used	
2	D	Not used	
3	A	Not used	
3	B	Not used	
3	C	Not used	
3	D	Not used	Lowest

Each HPC contains 16 hardware interrupt vector registers, with each one assigned to a different interrupt signal. The interrupt vectors are initialized by software.

4.3.3.2 PCI Error Interrupts

Each HPC independently detects errors and generates interrupts based on these errors. To ensure that error conditions are reported before any accompanying device interrupt, the HPC gives error interrupts priority over all other Up Hose traffic. Error interrupts are always inserted at the head of the HPC's internal Up Hose queue.

The IPL settings are provided for the I/O port and CPU's use and to maintain flow control between the PCIA and the I/O port. When a grant is issued to the HPC, the INTR/IDENT is sent to the I/O port and status of the outstanding interrupt is sent to each of the HPCs over UP_STS<1:0>.

No attempt is made to prioritize error interrupts between the PCI buses. Up Hose arbitration is used to determine the order of error interrupt servicing between PCI buses. Since each HPC detects errors independently, each HPC generates its own interrupt for the error. The vector transmitted in the INTR packet is obtained from the HPCs ERRVEC CSR bits <15:0>. Software can distinguish which HPC generated the error interrupt from the interrupt vector and is responsible for clearing all bits of the HPC's error register when it services an error interrupt.

When an Error Interrupt Status Return packet is received over the Down Hose, it is sent to each of the HPCs. This provides the flow control that allows the PCIA to send another INTR/IDENT over the Up Hose at the error IPL. Below is a list of errors that can cause error interrupts to the CPU. Errors that occur asynchronous to the CPU generate an error interrupt. Asynchronous errors occur during the execution of CPU-initiated write commands or during the execution of DMA transactions. Errors that occur during CPU-initiated reads are synchronous to the CPU and do not generate interrupts.

- Mailbox Illegal Length Error
- Mailbox Parity Error
- CSR Overrun Error
- CSR Parity Error
- DMA Read Return Illegal Length Error
- DMA Read Return Data Parity Error
- DMA Map RAM Parity Error
- DMA Map RAM Invalid Entry Error
- PCI Data Parity Error
- PCI Write Parity Error
- PCI Target Abort Error
- PCI Target Disconnect Error
- PCI Nonexistent Address Error
- PCIA Illegal CSR Address Error
- PCIA Map RAM Parity Error
- PCI SERR_L Asserted Error

4.3.3.3 Masking PCI Interrupts

Both PCI device interrupts and PCIA error interrupts can be masked through each HPC's IMASK register. PCI device interrupts are individually masked by clearing any of bits <15:0> of the register. All PCIA error interrupts can be masked through a single error interrupt enable bit, bit <16>. Error interrupts are not generated to the CPU when the interrupt enable bit is clear. See Chapter 3 for a description of the PCIx Error Summary Register and the PCIx Interrupt Mask Register.

4.3.3.4 EISA Interrupts

EISA device interrupts and error interrupts from the PCI-to-EISA bridge are reported through a single PCI interrupt. The device interrupts are masked and prioritized in the standard I/O module. The EISA Non-Maskable Interrupt (NMI) input from the PCI-to-EISA bridge is routed to a second PCI interrupt.

EISA device interrupts are consolidated by a tree of five 8259 interrupt controllers located on the standard I/O module. The controllers are wired in a master/slave configuration. When an EISA interrupt is pending, the master 8259 asserts an interrupt line to the HPC. The HPC sends an INTR/IDENT packet on the Up Hose with a vector corresponding to the asserted HPC input. The interrupt handler responds by reading the standard I/O IACK register. Only one read operation is necessary, since the standard I/O module contains the logic to generate the dual IACK pulses required by the 8259 tree. The data returned by the CSR read indicates which 8259 input was asserted. Refer to the 8259 data sheets for more information.

4.3.4 Diagnostic Features

4.3.4.1 Interrupt Pending Registers

Each HPC contains an Interrupt Pending Register that assigns a bit to each of the 16 hardware interrupts and the error interrupt. When an interrupt is received, the corresponding pending bit is set. The bit remains set until the interrupt is sent to the Up Hose as an INTR/IDENT packet at which time it is cleared. The register bit descriptions of the IPENDx CSR are shown in Table 4-12.

Table 4-12 Interrupt Pending Register Bit Definitions

Bit	Without Standard I/O	With Standard I/O
16	Error	Error
15	Slot 3 INTD	Not used
14	Slot 3 INTC	Not used
13	Slot 3 INTB	Not used
12	Slot 3 INTA	Not used
11	Slot 2 INTD	Not used
10	Slot 2 INTC	Not used
9	Slot 2 INTB	Not used
8	Slot 2 INTA	Not used
7	Slot 1 INTD	Not used
6	Slot 1 INTC	Not used
5	Slot 1 INTB	Not used
4	Slot 1 INTA	Tulip
3	Slot 0 INTD	Not used
2	Slot 0 INTC	Not used
1	Slot 0 INTB	EISA Bridge
0	Slot 0 INTA	NMI

4.3.4.2 Interrupt in Progress Registers

Each HPC contains an Interrupt in Progress Register, which is used to indicate the interrupt currently outstanding to the I/O port for which an interrupt status packet has not yet been returned. The Interrupt in Progress Register is cleared when the Interrupt Status Return packet is received on the Down Hose. Bit <5> of the register is set if an error interrupt is outstanding. Bit <4> is set if a hardware device interrupt is outstanding to the I/O port. Bits <3:0> indicate the PCI device interrupt that is outstanding. See the description of the IPROGx register in Chapter 3.

Chapter 5

Error Handling

5.1 Error Categories

There are two error categories: fatal and nonfatal. A fatal PCIA error is any error that invalidates the mailbox/CSR, DMA, or interrupt flow control mechanisms between the PCIA and the I/O port. Fatal errors can only be cleared by a full reset of the PCIA adapter. The Hose Error LED at the hose connection to the PCIA is lit when a fatal error is detected on the PCIA.

Nonfatal PCIA errors are errors that do not invalidate the flow control mechanisms. Any nonfatal error that occurs asynchronous to the CPU causes an error interrupt to be sent to the CPU. Asynchronous errors can occur during disconnected CPU-initiated write operations and during DMA operations. Errors that occur synchronous to CPU execution during CPU-initiated reads do not generate an interrupt but instead are indicated by sending a Read Data Return packet with error status to the I/O port. If the read was through a direct CSR access, the I/O port reports the error to the CPU by writing to the CSR Read Error Return register, which in turn triggers an interrupt through machine check vector 0x660. If the read was through a mailbox, the mailbox ERR will be set to indicate that an error occurred.

Nonfatal errors that occur during the execution of a CPU-initiated transaction can occur before or after an HPC or PCI destination is selected. Errors that occur before a destination is selected set an error bit in all the HPCs and cause an interrupt to be generated to the CPU. Since HPCs detect errors independently, each HPC generates its own interrupt. However, only a single Up Hose status packet is returned by the default HPC, HPC 0. An error that occurs on a CPU-initiated command after the destination is selected sets an error bit in the destination HPC. In this case, however, the Up Hose status packet is sent by the HPC that detects the error. Additionally, if the command is a write, an interrupt is generated to the CPU.

An HPC, upon detecting a nonfatal error, returns to its normal operating state after reporting the error to the CPU. No action is required by software to reenable PCIA operation after a nonfatal error.

5.2 Down Hose Header Cycle Parity Error

There is only one Down Hose error. The first cycle of any Down Hose packet that contains the command field bits <13:12> is the header cycle.

Any parity error that is detected on this cycle by any of the HPCs causes a fatal error since the command cannot be decoded. The HPC that detects the error asserts ERROR_L on the Up Hose. All HPCs monitor the state of the ERROR_L signal to determine whether packets can be transmitted and received over the hoses. Once ERROR_L is asserted, it remains asserted until a Down Hose reset is performed.

5.3 Mailbox Errors

Mailbox Illegal Length Error

Mailbox packets are eight longwords in length. A mailbox illegal length error occurs when the packet is not eight longwords long. This nonfatal error is detected before a destination is selected. Upon detecting this error, the HPC sets bit <1> of its error register. If error interrupts are enabled, an INTR/IDENT packet is sent to the I/O port.

Mailbox Parity Error

Any parity error detected on the second through eighth cycle of the Mailbox Command packet is a mailbox parity error. This is a nonfatal error, which is detected before a destination is selected. Upon detecting this error, the HPC sets bit <2> of its error register. If error interrupts are enabled, an INTR/IDENT packet is sent to the I/O port.

5.4 CSR Command Errors

CSR Overrun Error

The maximum size of a Down Hose CSR command is 18 cycles. Any CSR packet that exceeds 18 cycles in length causes a CSR overrun error. This is a nonfatal error, which is detected before a destination is selected. Upon detecting this error, the HPC sets bit <3> of its error register. If error interrupts are enabled, an INTR/IDENT packet is sent to the I/O port.

CSR Packet Parity Error

A CSR packet parity error occurs when a parity error is detected on a nonheader CSR cycle. This is a nonfatal error, which is detected before a destination is selected. Upon detecting this error, the HPC sets bit <4> of its error register. If error interrupts are enabled, an INTR/IDENT packet is sent to the I/O port.

5.5 DMA Errors

DMA Read Data Return Parity/Length Error

This error occurs if a parity error is detected on a data longword of a DMA Read Data Return packet or if the packet has fewer than the expected number of longwords. Since the read data is returned to the PCI master in a cut-through fashion, the HPC passes the bad data and parity on to the PCI bus. This is done for both cut-through and store and forward operations. The HPC sets bit <6> of the error register and generates an error interrupt to the CPU if error interrupts are enabled. The PCI address of the failing DMA transaction is stored in the Failing Address CSR.

DMA Read Data Return Error

A DMA read data return error occurs when the error bit is set in the Down Hose Read Data Return packet. This error is not logged in the HPC nor is an error interrupt generated since the error occurred upstream on the read and was detected by the I/O port or a memory module. The HPC issues a target abort to the PCI master for this error.

DMA Map RAM Parity Error

A DMA map RAM parity error occurs when map RAM read data is returned to the HPC with bad parity during the scatter/gather address translation of a DMA access. If the operation is a DMA write, the HPC has already buffered the entire DMA packet, and therefore cannot issue a target abort to the PCI master. The write data is discarded. Bit <7> of its error register is set and an error interrupt is generated to the CPU if error interrupts are enabled. The error interrupt is guaranteed to go up the hose before any subsequent transactions from the PCI device.

Since the DMA write is disconnected from its PCI master, the master reports a successfully completed data transfer after finishing the DMA stream. If the operating system cannot associate the PCIA error interrupt with a subsequent I/O completion, then the system must crash when this error is detected.

If the operation is a DMA read, a target abort is issued to the PCI master. Bit <7> of the error register is set, and if error interrupts are enabled, an interrupt is generated to the CPU. When the PCI master is aborted, it also sets bit <12> in its status register in PCI configuration space. In both read and write transactions, the PCI address of the failing DMA transaction is stored in the Failing Address CSR.

DMA Map RAM Invalid Entry Error

A DMA map RAM invalid entry error occurs if, during a map RAM read, bit <1> of the entry is a zero. If the operation is a DMA write, the HPC has already buffered the entire DMA packet, and therefore cannot issue a target abort to the PCI master. The write data is discarded. Bit <8> of the error register is set, and an error interrupt is generated to the CPU if error interrupts are enabled. The error interrupt is guaranteed to go up the hose before any subsequent transactions from the PCI device.

Since the DMA write is disconnected from its PCI master, the master reports a successfully completed data transfer after finishing the DMA stream. If the operating system cannot associate the PCIA error interrupt with a subsequent I/O completion, then the system must crash when this error is detected.

If the operation is a DMA read, a target abort is issued to the PCI master. Bit <8> of the error register is set, and if error interrupts are enabled, an interrupt is generated to the CPU. When the PCI master is aborted, it also sets bit <12> in its status register in PCI configuration space. In both read and write transactions, the PCI address of the failing DMA transaction is stored in the Failing Address CSR.

5.6 PCI Errors

PCI Address Parity Error

A PCI address parity error occurs if an HPC as a target detects a parity error on a PCI address phase. The HPC that detects this error sets bit <9> of its error register and generates an error interrupt to the CPU if error interrupts are enabled. If other PCI nodes detect the same error, this bit may be set in conjunction with the PCI SERR_L asserted error.

PCI Data Parity Error

A PCI data parity error is detected if an HPC as a master detects a parity error on a PCI read data phase or if the HPC as a master detects the assertion of PERR_L by another PCI node during a write. Bit <10> of its error register is set, and the HPC that detects this error returns an Up Hose status packet with the error bit set to the I/O port. On reads, the I/O port recognizes the error bit, and triggers a machine check error. On writes, the I/O port ignores the error bit. An error interrupt is generated to the CPU if the operation is a write and error interrupts are enabled.

PCI Write Parity Error

A PCI write parity error is detected if an HPC as a target detects a parity error on a PCI write data phase. The HPC allows the DMA logic to receive data from the PCI but asserts PERR L two cycles after the data phase. This causes the transaction to be dropped. The HPC sets bit <11> of its error register and generates an error interrupt to the CPU. The PCI address of the failing DMA transaction is stored in the Failing Address CSR.

PCI Target Abort Error

A PCI target abort error is detected if a transaction with HPC as a master is aborted by a PCI target. Bit <12> of its error register is set, and the HPC that detects this error returns an Up Hose status packet with the error bit set to the I/O port. On reads, the I/O port recognizes the error bit and triggers a machine check error. On writes, the I/O port ignores the error bit. An error interrupt is generated to the CPU if the command is a write and if error interrupts are enabled. This error can indicate corrupted data and should result in a system crash.

PCI Target Disconnect Error

A PCI target disconnect error occurs if an HPC as a master is disconnected by a target in consecutive retries to the same longword up to the limit specified in the HPC's RETRY CSR. Bit <13> of its error register is set, and the HPC that detects this error returns a status packet with the error bit set to the I/O port. On reads, the I/O port recognizes the error bit and triggers a machine check error. On writes, the I/O port ignores the error bit. An error interrupt is generated to the CPU if the command is a write and if error interrupts are enabled.

PCI Nonexistent Address Error

A PCI nonexistent address error occurs when no HPC as master detects the assertion of PCI DEVSEL_L on any of the three PCI buses in response to a valid address phase. In this case, HPC 0 sets bit <14> of its error register and returns a status packet with the error bit set to the I/O port. On reads, the I/O port recognizes the error bit and triggers a machine check error. On writes, the I/O port ignores the error bit. An error interrupt is generated to the CPU if the command is a write and error interrupts are enabled.

PCI SERR_L Asserted Error

A PCI SERR_L asserted error occurs when an HPC detects the assertion of the PCI SERR_L signal by any PCI device. This signal may be asserted when the device detects a PCI address parity error, a data parity error on a PCI special cycle, or any other error condition the device considers catastrophic. Bit <18> is set, and the HPC that detects the signal generates an error interrupt to the CPU if error interrupts are enabled. The PCI device that asserts SERR L sets bit <14> in its status register in PCI configuration space.

PCI Disconnected Master Abort Error

This error occurs when the HPC detects a master abort on a PCI bus during an I/O window or mailbox transaction, and the transaction was previously claimed and disconnected by a node on that PCI bus. The HPC is thus unable to complete a PCI burst that had been successfully started and disconnected. The HPC sets bit <5> of its error register and generates an error interrupt to the CPU if error interrupts are enabled. The most likely cause for this error is an incorrectly configured PCI device or software that performs a multi-longword access across the end of a range of device CSRs.

PCI Incremental Latency Exceeded

This condition is indicated when the PCIA as a PCI target is performing a burst transaction, and the PCIA must stall more than eight PCI cycles between driving adjacent data cycles on the bus. This violates a PCI guideline, and indicates that the PCIA is encountering excessive latency in accessing main memory. This is not considered an error condition. Bit <17> is set by the HPC that detects the condition. The PCI transaction completes normally, and no interrupt is generated. This condition is detected to aid in configuring the DMA behavior of the PCIA.

5.7 PCIA Errors

PCIA Command Overflow Error

A PCIA command overflow error occurs if the PCIA detects a Down Hose CPU-initiated command packet while all of its Down Hose CPU command buffers are full. This is a fatal error. The PCIA asserts ERROR_L on the Up Hose. All HPCs monitor the state of the ERROR_L signal to determine whether packets can be transmitted and received over the hoses. Once ERROR_L is asserted, it remains asserted until a Down Hose reset is performed.

PCIA Illegal CSR Address

A PCIA illegal CSR address occurs when a CPU-initiated command packet addresses an invalid HPC CSR location. Bit <15> of the HPC’s error register is set, and the HPC that detects this error returns a status packet to the I/O port with the error bit set. An error interrupt is generated to the CPU if the command is a write and error interrupts are enabled.

PCIA Map RAM Parity Error

A PCIA map RAM parity error occurs when a CPU-initiated read command to the map RAM returns data with bad parity. Bit <16> of the

HPC0's error register is set and the status packet is returned with the error bit set to the I/O port.

5.8 Up Hose Packet Count Error

There is only one Up Hose error. An Up Hose packet count error occurs when any HPC detects an underflow or overflow condition on its Up Hose packet counter. This is a fatal error. The HPC asserts ERROR_L on the Up Hose. All HPCs monitor the state of the ERROR_L signal to determine whether packets can be transmitted and received over the hoses. Once ERROR_L is asserted, it remains asserted until a Down Hose reset is performed.

5.9 PCI Peer-to-Peer Errors

Since any PCI device can function as both a master and a target, there may be transactions on the PCI that do not involve the HPC. In these cases the HPC is a bystander and does not directly detect some errors.

If a parity error occurs on a PCI address cycle, it is impossible to tell which device was being addressed. The HPC detects the error, asserts SERR L on the bus, and generates an error interrupt as described above. It is likely other PCI devices will also detect the error and assert SERR L, provided they have error reporting enabled.

If a parity error occurs during a PCI data cycle, and the HPC is not participating in the transaction, the error will not be detected by the HPC. Instead, it is detected by the PCI device involved. That device will then report the error through its interrupt, provided the device has parity error reporting enabled.

5.10 PCI Configuration Errors

It is possible for software to erroneously configure two PCI devices to respond to overlapping address ranges. This will cause bad and potentially unpredictable things to happen. If both devices are on the same physical PCI segment, they will both attempt to respond to an address. In the worst case, this could cause physical damage if two devices try to drive a PCI bus signal to opposite values.

If the devices are on different physical PCI segments, a single host-initiated request will receive two replies. Eventually, this will cause an I/O port module error when the hose flow control counters underflow. However, depending on the dynamics of the hose traffic, many transactions could occur before the error is detected.

Chapter 6

PCI to EISA Bridge

The PCIA supports an EISA bus using the standard I/O module. The module provides connectivity to eight EISA slots by an INTEL 82375EB PCI-EISA Bridge Component (PCEB) and the 82374EB EISA System Component (ESC), also known as the Mercury chipset. The EISA bridge module plugs into a special, dedicated slot on PCI segment 0.

The PCEB is a fully buffered bidirectional interface between the PCI bus and the EISA bus. The PCEB also offers PCI arbitration logic which is not used in the PCIA implementation. The ESC adds EISA bus control and arbitration.

The standard I/O module includes an integrated PCI I/O Ethernet port (Tulip chip). The XBus includes chips that provide PC-style serial, parallel, floppy, keyboard, and mouse ports, plus NVRAM, real time clock, and Ethernet address ROM. A cascade of 8259 interrupt conrtrollers prioritize interrupts from EISA and XBus sources.

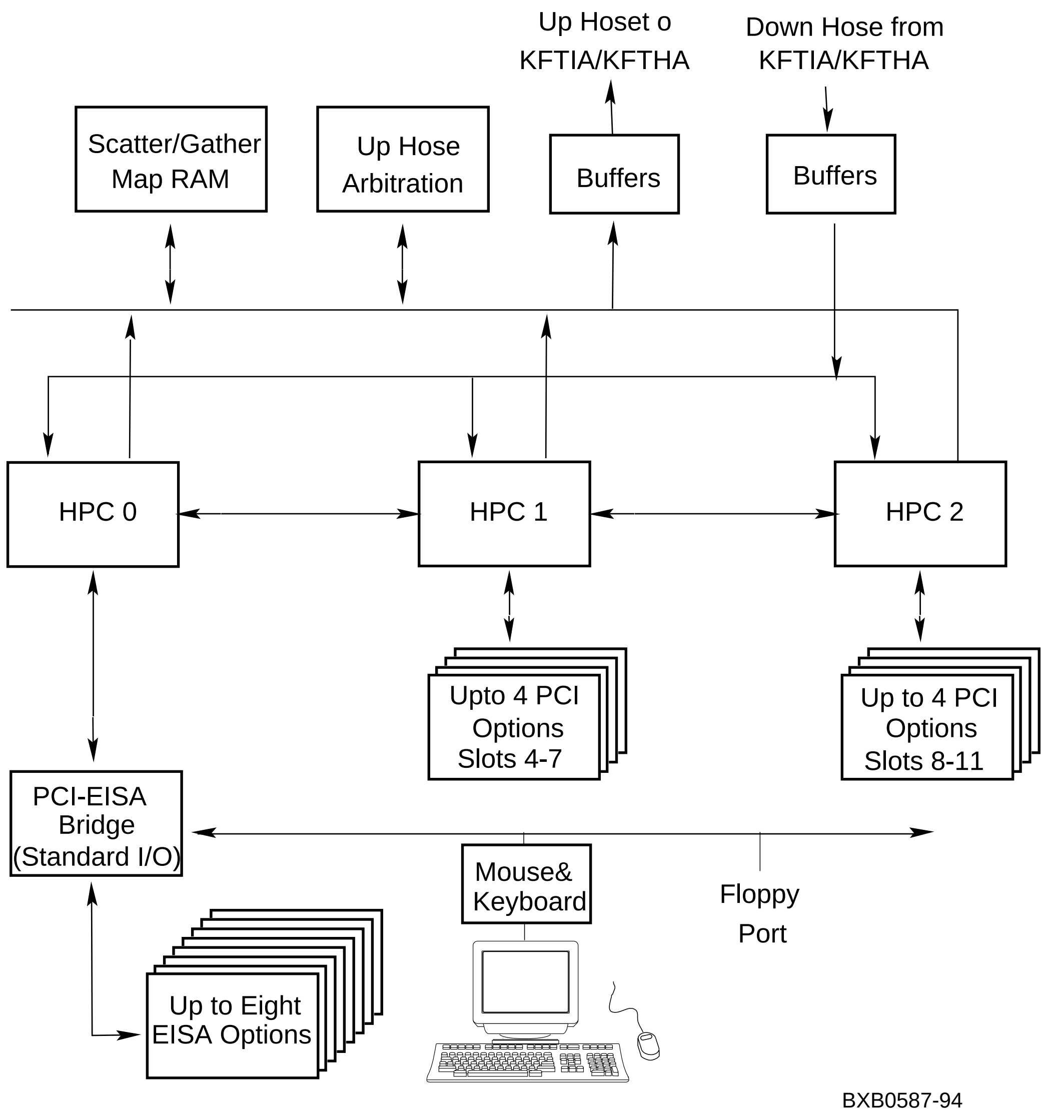
The Ethernet port is usable directly. Support for the other ports requires an internal cable and bulkhead that occupies the adjacent PCI/EISA expansion slot.

Support for the various functions of the standard I/O module is operating system dependent.

6.1 PCI and EISA Subsystem Block Diagram

The EISA eight-slot subsystem consists of the PCI-EISA bridge chipset attached to one PCI bus. Figure 6-1 shows this structure. The bridge chips are on a module installed in a dedicated slot on PCI segment 0.

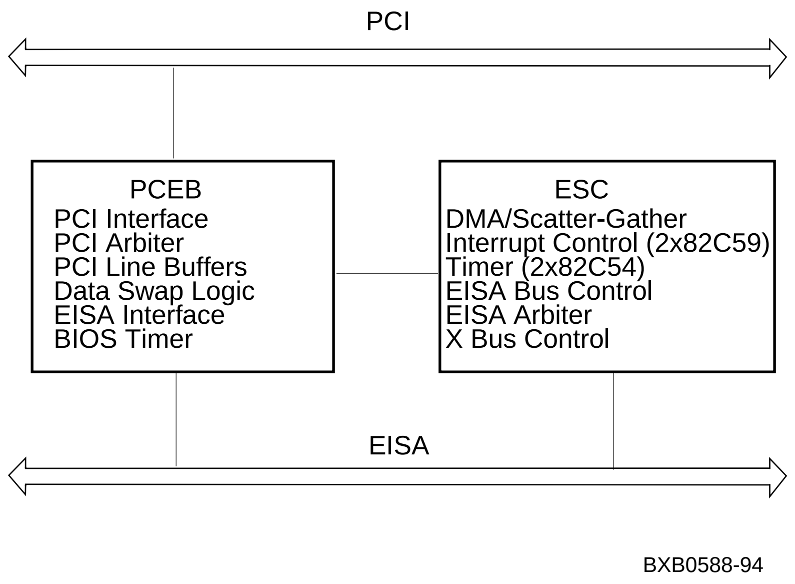
Figure 6-1 PCI and EISA Subsystem



6.2 PCI-EISA Bridge Chipset

Figure 6-2 shows the functional division between the two components of the INTEL PCI-EISA bridge chipset. The PCEB contains most of the PCI interface as well as the entire datapath connection between the two buses. The ESC manages the EISA and ISA protocol and helps to control the EISA side of the PCEB datapath.

Figure 6-2 PCI-EISA Bridge Chip Function



6.3 Address Mapping

6.3.1 PCI to EISA

The standard I/O module "resides" within the bottom 64 Kbytes of the PCI I/O space. The bridge can be configured to pass additional PCI address ranges through to the EISA bus. Any PCI transaction that addresses the bottom 64 Kbytes of PCI I/O space and does not receive a DEVSEL# is subtractively decoded by the PCEB and passed on to the EISA bus. Refer to the PCEB Specification for additional information on subtractive decoding.

Table 6-1 PCI to EISA Address Mapping

CPU Address Range	PCI Address Range	PCI Address Space
x2.0000.0000 - x2.0001.FFFF	0000.0000 - 0000.0FFF	EISA bridge registers and XBus addressing
x2.0002.0000 - x2.0003.FFFF	0000.1000 - 0000.1FFF	EISA slot 1
x2.0004.0000 - x2.0005.FFFF	0000.2000 - 0000.2FFF	EISA slot 2
x2.0006.0000 - x2.0007.FFFF	0000.3000 - 0000.3FFF	EISA slot 3
x2.0008.0000 - x2.0009.FFFF	0000.4000 - 0000.4FFF	EISA slot 4
x2.000A.0000 - x2.000B.FFFF	0000.5000 - 0000.5FFF	EISA slot 5
x2.000C.0000 - x2.000D.FFFF	0000.6000 - 0000.6FFF	EISA slot 6
x2.000E.0000 - x2.000F.FFFF	0000.7000 - 0000.7FFF	EISA slot 7
x2.0010.0000 - x2.0011.FFFF	0000.8000 - 0000.8FFF	EISA slot 8
x2.0012.0000 - x2.001F.FFFF	0000.9000 - 0000.FFFF	Available for remaining PCI options
x2.0020.0000 - x2.1FFF.FFFF	0001.0000 - 00FF.FFFF	PCI I/O space - Sparse mapping (fixed)
x2.2000.0000 - x2.FFFF.FFFF	0100.0000 - 07FF.FFFF	PCI I/O space - Sparse mapping (variable)

6.3.2 EISA to PCI

The PCI-EISA bridge passes all EISA traffic to the PCI side with the exception of a fixed "hole" between 0008 0000 and 0010 0000 (512 to 1024 Kbytes) and a programmable hole. EISA devices require the fixed hole's address range to be served locally (that is within the EISA subsystem) for backward compatibility with ISA PC systems. The programmable hole may be used if an EISA device has EISA addressable memory.

6.4 Bridge Buffers

The PCEB buffers packets in both directions. Four line buffers (LBs), each capable of holding 16 bytes of data, smooth write data flow from EISA to PCI or read data flow from PCI to EISA. Strict ordering is maintained. These buffers are flushed and bypassed for PCI interrupt acknowledge cycles. The bridge chipset automatically controls the flushing of line buffers as necessary. Note that an additional form of buffering, posted write buffers, are no longer supported by the Mercury chipset.

6.5 Interrupts

Interrupt lines from the EISA and XBus devices are connected to the 8259 interrupt controller tree on the standard I/O module. The outputs of the master 8259 are connected to the HPC for PCI segment 0. Table 6-2 shows the interrupt connections. The HPC interrupt pin determines the interrupt priority, with interrupt 0 being the highest.

Table 6-2 EISA Bridge and Standard I/O Interrupt Connections

Device	HPC INTR Pin	Shared With
EISA Bridge Nonmaskable Interrupt (NMI)	0	PCI Bus 0 Slot 0 - INTA
EISA Bridge Device and Internal Interrupts	1	PCI Bus 0 Slot 0 - INTB
Standard I/O Ethernet chip (Tulip)	4	PCI Bus 0 Slot 1 - INTA

EISA and standard I/O device interrupts are consolidated and prioritized by a tree of five 8259 interrupt controller chips. The interrupt sources are connected as shown in Table 6-3. When an EISA interrupt is pending, the master 8259 chip asserts an interrupt line on the HPC, as shown above. The interrupt vector for the HPC interrupt points to an operating system EISA interrupt dispatcher. The dispatcher must perform a CSR read of the Standard I/O IACK register to obtain the identity of the interrupt source. Reading the IACK register generates a pair of IACK pulses needed by the 8259s.

A passive release condition is indicated by an interrupt through request line 7 on the master 8359, with no corresponding interrupt indicated in the 8259's Interrupt Status Register.

Table 6-3 Standard I/O EISA Interrupt Assignments

8258	Interrupt line	Function
Master	0	ESC chip interrupt (DMA and timers)
	1	Cascade from Slave 0
	2	Reserved
	3	Cascade from Slave 1
	4	Cascade from Slave 2
	5	Cascade from Slave 3
	6	Reserved
	7	Reserved for passive release
Slave 0	0	Not used with PCIA
	1	Not used with PCIA
	2	Tulip Ethernet port
	3	Mouse
	4	Not used with PCIA
	5	Not used with PCIA
	6	Keyboard
	7	Floppy disk controller
Slave 1	0	Serial port 2
	1	Parallel port
	2	EISA IRQ 3
	3	EISA IRQ 4
	4	EISA IRQ 5
	5	EISA IRQ 6
	6	EISA IRQ 7
	7	Serial port 1

6.6 EISA Bus Lock

The EISA DSSI option requires the use of bus locks. This option can assert EISA LOCK to perform atomic operations to the VAXport command queue headers. The PCEB chip passes on the lock state by asserting PCILOCK when it arbitrates for and wins ownership of the PCI bus. The PCILOCK signal *must* be asserted in the cycle following the address phase. The HPC monitors the PCILOCK line and if it sees that the PCEB chip has asserted that signal during a read command, it converts the read to an IREAD on the Up Hose. When the quadword of read data returns, the HPC passes the data on to the PCEB which in turn forwards the data to the EISA adapter. The EISA adapter deasserts EISA LOCK at this point so that PCILOCK may be released by the PCEB. The EISA adapter then examines the lowest order bit in the quadword and decides whether it has gained access to the shared resource based on the state of the least significant bit. A one indicates that another device or a CPU currently owns the lock variable. If this is the case, the adapter retries the EISA LOCK command.

Datapath buffering is disabled by the PCEB during EISA LOCK operations.

6.7 Lockout Prevention

The PCIA and KFTIA/KFTHA process commands in the order they are presented. No buffers in the PCIA require special flushing or synchronization between the read and write transactions, so no special steps are needed to prevent lockout between the PCI and EISA subsystems.

6.8 Configuration Rules

The PCIA has 12 expansion slots (card cage positions) that can hold expansion modules. Eight of these positions are equipped with both EISA and PCI connectors, and can accommodate either type of module. Table 6-4 shows the slot usage when the standard I/O module is installed.

The standard I/O module shares PCI bus resources with some of the PCI adapter module slots. The EISA bridge occupies the configuration space addresses also used by PCI slot 0, as described in Chapter 2. The EISA bridge and standard I/O PCI devices use the interrupt lines that are also used by PCI slots 0 and 1, as described above. Therefore, PCI slots 0 and 1 must not be populated with PCI adapter modules when the standard I/O module is installed. Slot 0 is physically blocked by the standard I/O module. Console or diagnostic firmware is responsible for indicating an error condition if both the standard I/O slot and PCI slot 1 are populated. The information required to perform this check is available in the PCIA PRESENT register.

If the standard I/O module is not present, all twelve slots are available for PCI options. If the standard I/O module is installed, there is one EISA-only slot, seven PCI or EISA slots, and three PCI-only slots.

Table 6-4 Slot Usage with Standard I/O Module Installed

Bulkhead Position	PCI Slot	EISA Slot
0	Physically/electrically blocked by standard I/O module	
1	Electrically blocked by standard I/O module	0 (EISA position only)
2	2 (PCI position only)	
3	3	1
4	4	2
5	5	3
6	6 (PCI position only)	
7	7	4
8	8	5
9	9	6
10	10 (PCI position only)	
11	11	7

6.9 EISA Bridge Registers

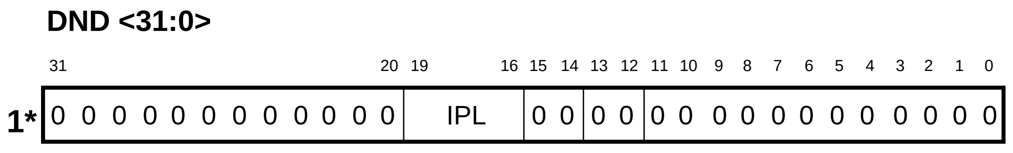
The EISA bridge registers are given in the *Intel 82420/82430 PCIset ISA and EISA Bridges Databook*, April 1993. The *PCI Adapter (PCIA) Engineering Specification*, February 1994, lists the EISA bride registers used on the PCIA.

Appendix A

PCIA Supported Hose Packets

A.1 Down Hose Packets

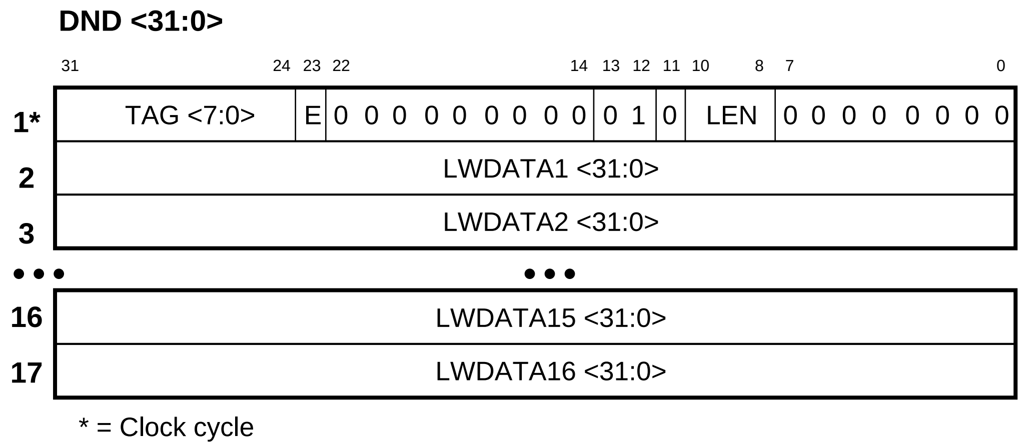
Figure A-1 Interrupt Status Return Packet



Bits <31:20> - Zero
Bits <19:16> - IPL
Bits <15:14> - Zero
Bits <13:12> - 00, Interrupt status return command.
Bits <11:0> - Zero

BXB-0643A-94

Figure A-2 DMA Read Data Return Packet



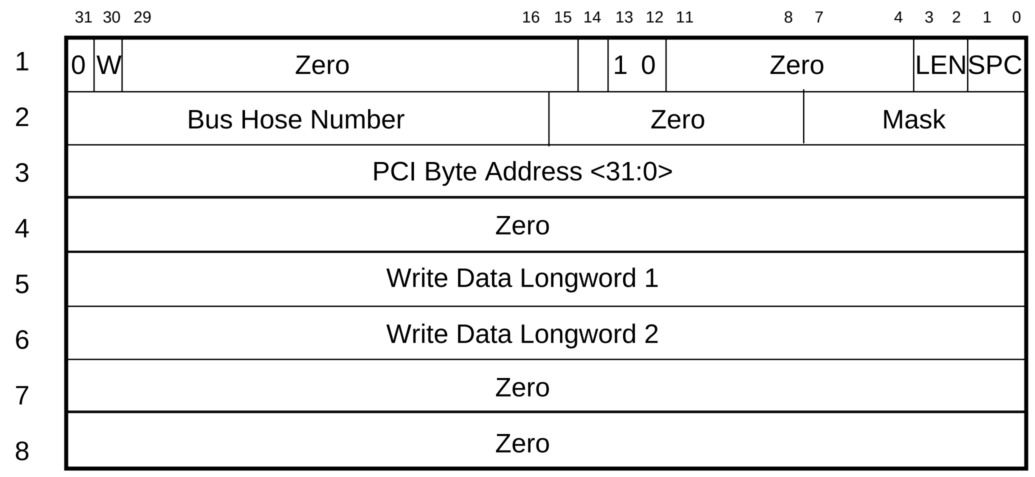
Cycle 1	Bits <31:24>	-	Tag <7:0>
Cycle 1	Bit <23>	-	Error
Cycle 1	Bits <22:14>	-	Zero
Cycle 1	Bits <13:12>	-	01, DMA read return command.
Cycle 1	Bit <11>	-	Zero
Cycle 1	Bits <10:8>	-	DMA length
Cycle 1	Bits <7:0>	-	Zero

Cycles 2 - 17 Bits <31:0> - Data longwords

BXB-0641A-94

Figure A-3 Mailbox Command Packet

DND <31:0>



* = Clock cycle

Cycle 1 Bits <31> - Zero
Cycle 1 Bit <30> - Who am I bit
Cycle 1 Bits <29:15> - Zero
Cycle 1 Bit <14> - Read/Write bit. 0 is a Read, 1 is Write.
Cycle 1 Bits <13:12> - 10, Indicates this is a mailbox command.
Cycle 1 Bits <11:4> - Zero
Cycle 1 Bits <3:2> - Byte Length Field.
00 = Byte
01 = Word
10 = Tribyte
11 = Longword, if PCI byte address <1:0> = 00 and PCI address space <1:0> = 01, 10, or 11
11 = Quadword, if PCI byte address <1:0> = 11 and PCI address space <1:0> = 01, 10, or 11
Cycle 1 Bits <1:0> - PCI Address Space Field <1:0>
00 = Dense PCI Memory Space
01 = Sparse PCI Memoy Space
10 = Sparse PCI I/O Space
11 = Sparse PCI Configuration Space

Cycle 2 Bits <31:16> - Hose number <15:0>
Cycle 2 Bits <15:8> - Zero
Cycle 2 Bits <7:0> - Byte mask. If bit is set, byte is masked.
If bit is clear, byte is written.

Cycle 3 Bits <31:0> - PCI byte address <31:0>

Cycle 4 Bits <31:0> - Zero

Cycle 5 Bits <31:0> - Data Longword 1 to be written

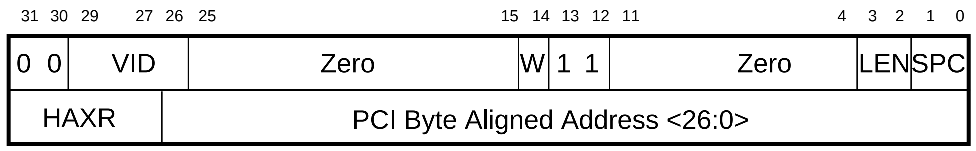
Cycle 6 Bits <31:0> - Data Longword 2 to be written

Cycle 7-8 Bits <31:0> - Zero

BXB-0599B-94

Figure A-4 CSR Sparse Read Packet

DND <31:0>



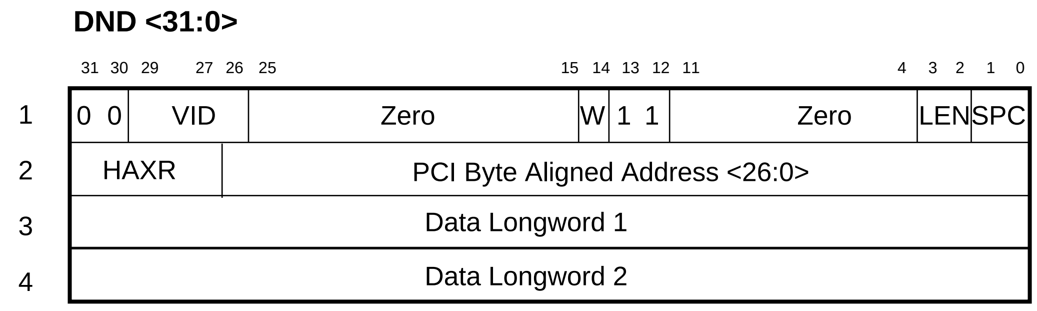
* = Clock cycle

Cycle 1 Bits <31:30> - Zero
Cycle 1 Bit <29:26> - Virtual ID of TLSB Commanding Node. Indicates the CPU that is the source of the transaction.
Cycle 1 Bits <25:15> - Zero
Cycle 1 Bit <14> - Read/Write bit. 0 is a Read.
Cycle 1 Bits <13:12> - 11, Indicates this is a CSR command.
Cycle 1 Bits <11:4> - Zero
Cycle 1 Bits <3:2> - Byte Length Field.
00 = Byte
01 = Word
10 = Tribyte
11 = Longword, if PCI byte address <1:0> = 00 and PCI address space <1:0> = 01, 10, or 11
11 = Quadword, if PCI byte address <1:0> = 11 and PCI address space <1:0> = 01, 10, or 11
Cycle 1 Bits <1:0> - PCI Address Space Field <1:0>
00 = Dense PCI Memory Space
01 = Sparse PCI Memoy Space
10 = Sparse PCI I/O Space
11 = Sparse PCI Configuration Space

Cycle 2 Bits <31:27> - Zero or the Memory Space Hardware Extension field of the PCIx CTL register
Cycle 2 Bits <26:0> - PCI Byte Aligned Address <26:0>

BXB-0599-94

Figure A-5 CSR Sparse Write Packet



* = Clock cycle

Cycle 1 Bits <31:30> - Zero
Cycle 1 Bit <29:26> - Virtual ID of TLSB Commanding Node. Indicates the CPU
that is the source of the transaction.
Cycle 1 Bits <25:15> - Zero
Cycle 1 Bit <14> - Read/Write bit. 0 is a Read.
Cycle 1 Bits <13:12> - 11, Indicates this is a CSR command.
Cycle 1 Bits <11:4> - Zero
Cycle 1 Bits <3:2> - Byte Length Field.
00 = Byte
01 = Word
10 = Tribyte
11 = Longword, if PCI byte address <1:0> = 00 and
PCI address space <1:0> = 01, 10, or 11
11 = Quadword, if PCI byte address <1:0> = 11 and
PCI address space <1:0> = 01, 10, or 11
Cycle 1 Bits <1:0> - PCI Address Space Field <1:0>
00 = Dense PCI Memory Space
01 = Sparse PCI Memoy Space
10 = Sparse PCI I/O Space
11 = Sparse PCI Configuration Space

Cycle 2 Bits <31:27> - Zero or the Memory Space Hardware Extension field of
the PCIx CTL register
Cycle 2 Bits <26:0> - PCI Byte Aligned Address <26:0>

Cycle 3 Bits <31:0> - Data Longword 1 to be written

Cycle 4 Bits <31:0> - Data Longword 2 to be written

BXB-0599A-94

Figure A-6 CSR Dense Read Packet

DND <31:0>

31 30	29 26	25 15	14	13 12	11 4	3 2	1 0
0 0	VID	Zero	W	1 1	Zero	LEN	SPC
PCI Byte Aligned Address <31:0>							

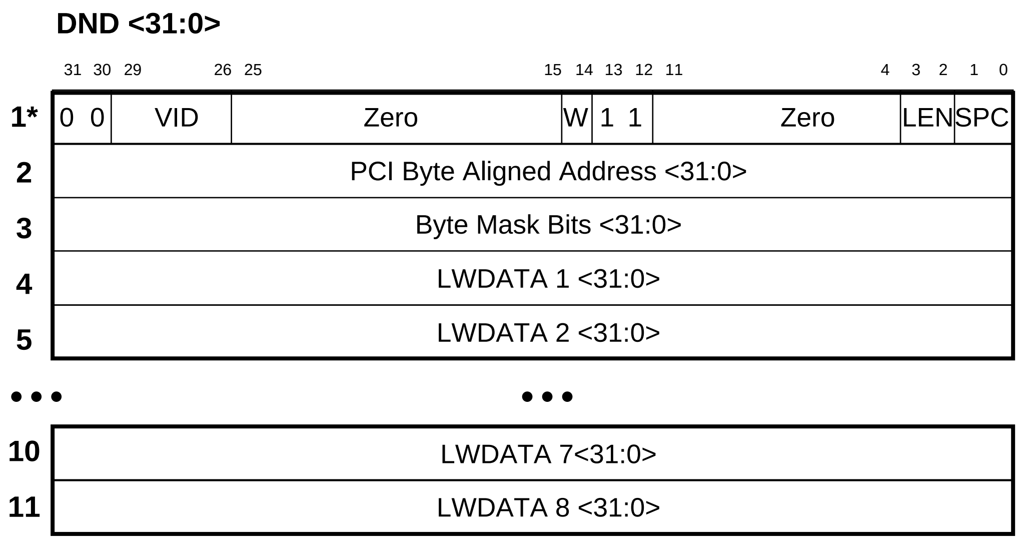
* = Clock cycle

Cycle 1 Bits <31:30> - Zero
Cycle 1 Bit <29:26> - Virtual ID of TLSB Commanding Node. Indicates the CPU that is the source of the transaction.
Cycle 1 Bits <25:15> - Zero
Cycle 1 Bit <14> - Read/Write bit. 0 is a Read.
Cycle 1 Bits <13:12> - 11, Indicates this is a CSR command.
Cycle 1 Bits <11:4> - Zero
Cycle 1 Bits <3:2> - Byte Length Field. Not used during dense reads.
Cycle 1 Bits <1:0> - PCI Address Space Field <1:0>
00 = Dense PCI Memory Space
01 = Sparse PCI Memoy Space
10 = Sparse PCI I/O Space
11 = Sparse PCI Configuration Space

Cycle 2 Bits <31:0> - PCI Byte Aligned Address <31:0>

BXB-0598-94

Figure A-7 CSR Dense Write Packet



* = Clock cycle

Cycle 1 Bits <31:30> - Zero
Cycle 1 Bit <29:26> - Virtual ID of TLSB Commanding Node. Indicates the CPU that is the source of the transaction.
Cycle 1 Bits <25:15> - Zero
Cycle 1 Bit <14> - Read/Write bit. 1 is a Write.
Cycle 1 Bits <13:12> - 11, Indicates this is a CSR command.
Cycle 1 Bits <11:4> - Zero
Cycle 1 Bits <3:2> - Byte Length Field. Not used during dense writes.
Cycle 1 Bits <1:0> - PCI Address Space Field <1:0>
00 = Dense PCI Memory Space
01 = Sparse PCI Memoy Space
10 = Sparse PCI I/O Space
11 = Sparse PCI Configuration Space

Cycle 2 Bits <31:0> - PCI Byte Aligned Address <31:0>

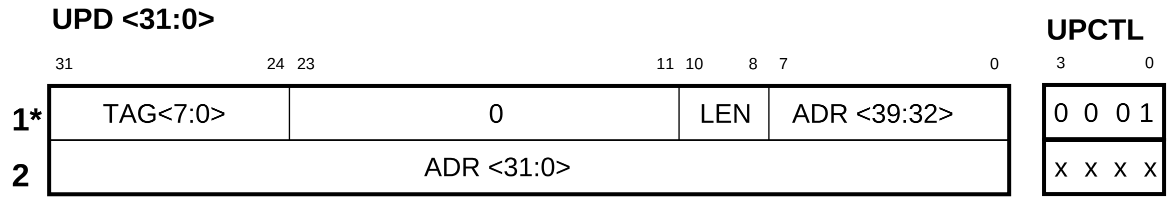
Cycle 3 Bits <31:0> - Byte mask<31:0>. If bit is set, byte is written.
If bit is clear, byte is masked.

Cycles 4 - 11 Bits <31:0> - Data longwords 1-8 to be written.

BXB-0597-94

A.2 Up Hose Packets

Figure A-8 DMA Read Packet



* = hose cycle

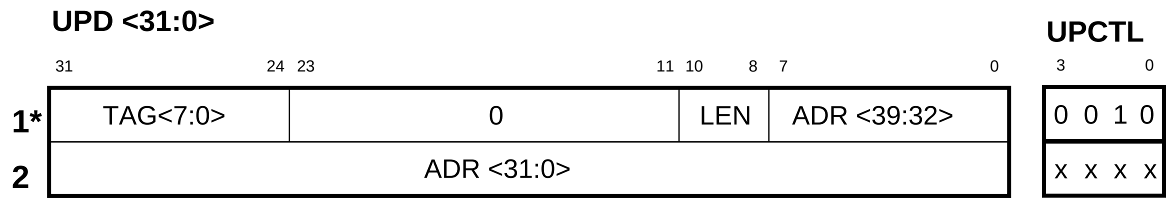
UPCTL <3:0> = 0001

Cycle 1 Bits <31:24> - Tag <7:0>. HPC ID is inserted as tag on all DMA read transactions.
Cycle 1 Bits <23:11> - Zero
Cycle 1 Bits <10:8> - DMA length code <2:0>. Hexword and Double Hexwords supported
000 = Hexword
100 = Double Hexword
Cycle 1 Bits <7:0> - DMA address bits <39:32>

Cycle 2 Bits <31:0> - DMA address bits <31:0>

BXB-0592-94

Figure A-9 IREAD Packet



* = hose cycle

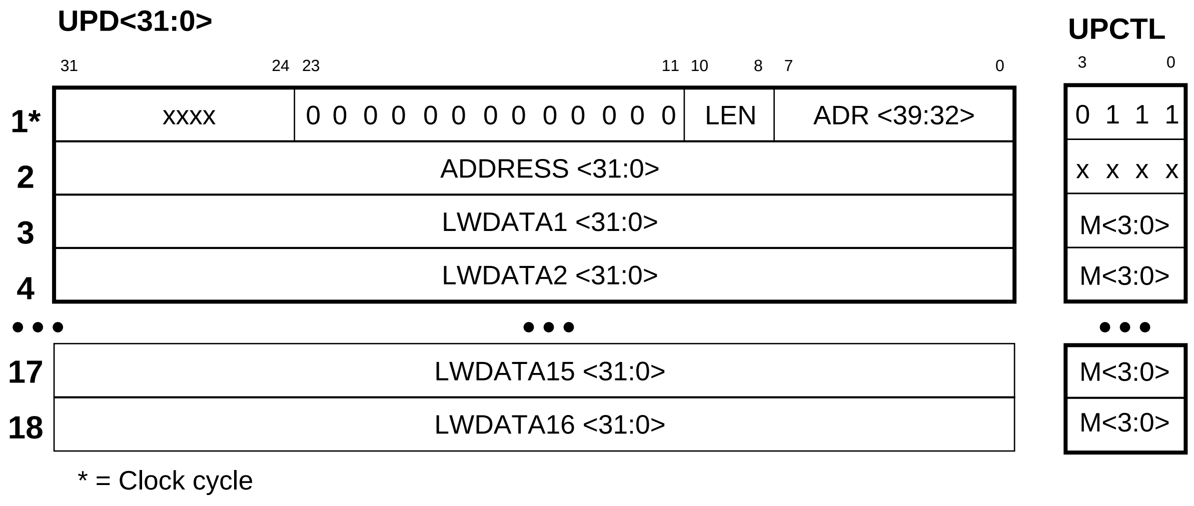
UPCTL <3:0> = 0010

Cycle 1 Bits <31:24> - Tag <7:0>. HPC ID is inserted as tag on all IRead transactions.
Cycle 1 Bits <23:11> - Zero
Cycle 1 Bits <10:8> - DMA length code <2:0>. Only Octaword supported
011 = Octaword
Cycle 1 Bits <7:0> - DMA address bits <39:32>

Cycle 2 Bits <31:0> - DMA address bits <31:0>

BXB-0593-94

Figure A-10 DMA Write Mask Packet



Cycle 1 UPCTL <3:0> = 0111

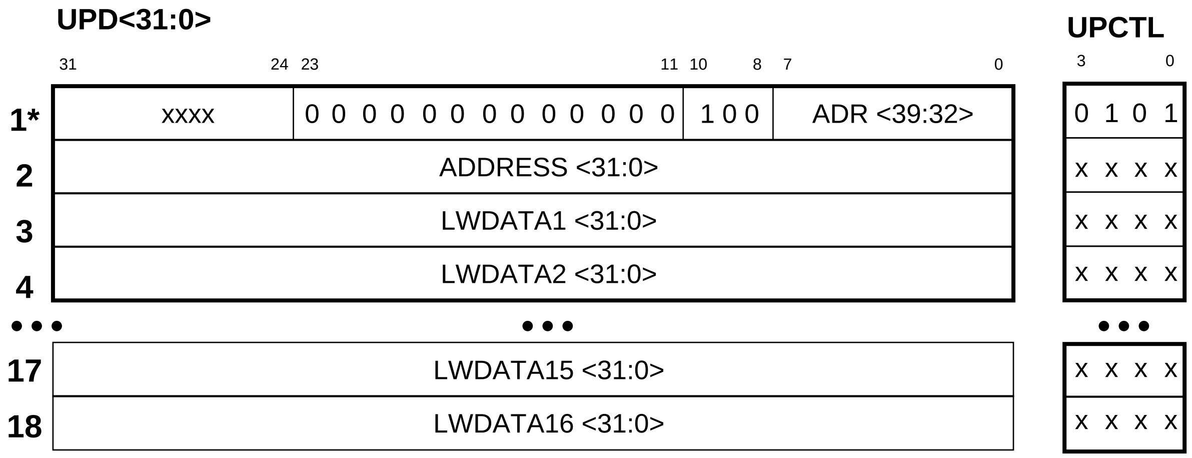
Cycle 1 Bits <31:24> - Tag <7:0>. HPC ID is inserted as tag on all DMA write transactions.
Cycle 1 Bits <23:11> - Zero
Cycle 1 Bits <10:8> - DMA length code <2:0>. Octaword, Hexword, and Double Hexwords
011 = Octaword
000 = Hexword
100 = Double Hexword
Cycle 1 Bits <7:0> - DMA address bits <39:32>

Cycle 2 UPCTL <3:0> = Don't care
Cycle 2 Bits <31:0> - DMA address bits <31:0>

Cycle 3 - 18 UPCTL <3:0> = Byte Mask
Cycle 3 - 18 Bits <31:0> - Data longwords 0 - 15 to be written.

BXB-0594-94

Figure A-11 DMA Unmasked Write Packet



* = Clock cycle

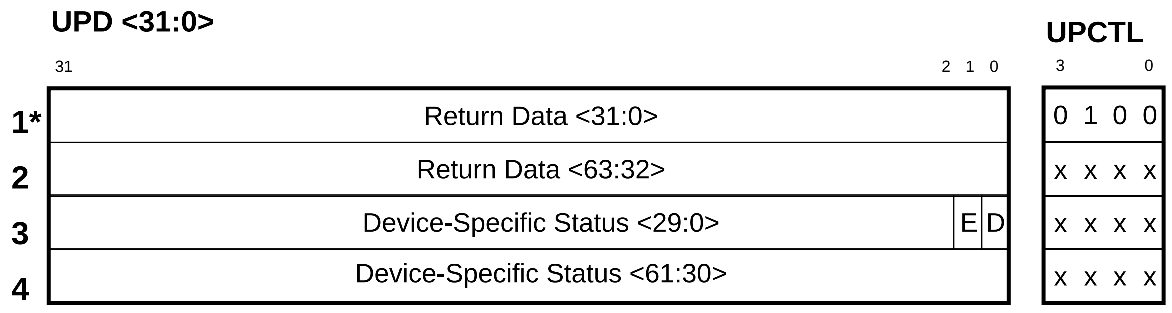
Cycle 1 UPCTL <3:0> = 0101
Cycle 1 Bits <31:24> - Tag <7:0>. HPC ID is inserted as tag on all DMA write transactions.
Cycle 1 Bits <23:11> - Zero
Cycle 1 Bits <10:8> - DMA length code <2:0>. Only Double Hexwords supported
100 = Double Hexword
Cycle 1 Bits <7:0> - DMA address bits <39:32>

Cycle 2 UPCTL <3:0> = Don't care
Cycle 2 Bits <31:0> - DMA address bits <31:0>

Cycle 3 - 18 UPCTL <3:0> =Don't care
Cycle 3 - 18 Bits <31:0> - Data longwords 0 - 15 to be written.

BXB-0595-94

Figure A-12 Mailbox Status Return Packet



* = hose cycle

Cycle 1 UPCTL <3:0> = 0100
Cycle 1 Bits <31:0> - Data Longword 0 read.

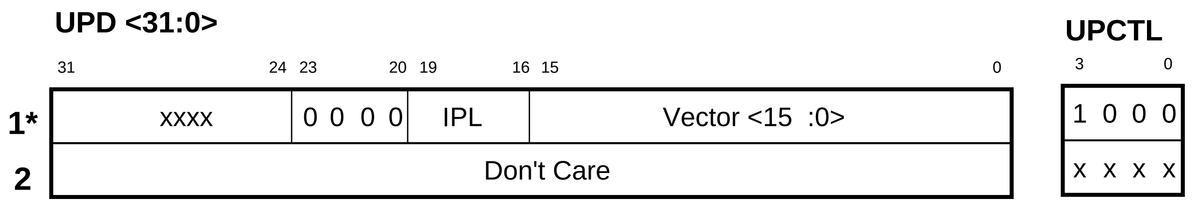
Cycle 2 UPCTL <3:0> = Don't care
Cycle 2 Bits <31:0> - Data Longword 1 read

Cycle 3 UPCTL <3:0> = Don't care
Cycle 3 Bits <31:2> - Zero
Cycle 3 Bit <1> - Error bit. Set if an error occurs during the transaction.
Cycle 3 Bit <0> - Done bit. Set when the mailbox command is completed.

Cycle 4 UPCTL <3:0> = Don't care
Cycle 4 Bits <31:0> - Zero

BXB-0644a-94

Figure A-13 Interrupt/IDENT Packet



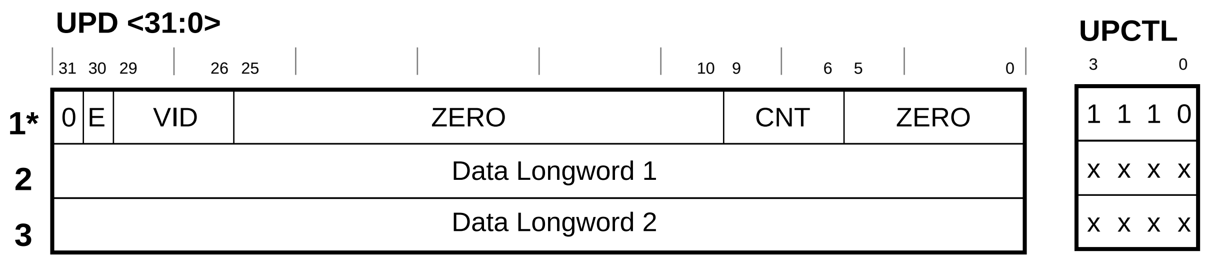
* = Clock cycle

Cycle 1 UPCTL <3:0> = 1000
Cycle 1 Bits <31> - PCIA specific bit.
0, if this is a device interrupt.
1, if this is an error interrupt.
Cycle 1 Bits <30:29> - PCIA specific field. HPC ID<1:0>.
Cycle 1 Bits <28:24> - PICA specific field. Zero.
Cycle 1 Bits <23:20> - Zero
Cycle 1 Bits <19:16> - IPL
Cycle 1 Bits <15:0> - Interrupt vector

Cycle 2 UPCTL <3:0> = Don't care
Cycle 2 Bits <31:0> - Zero

BXB-0647a-94

Figure A-14 CSR Sparse Read Return Packet



* = Clock cycle

Cycle 1 UPCTL <3:0> = 1110

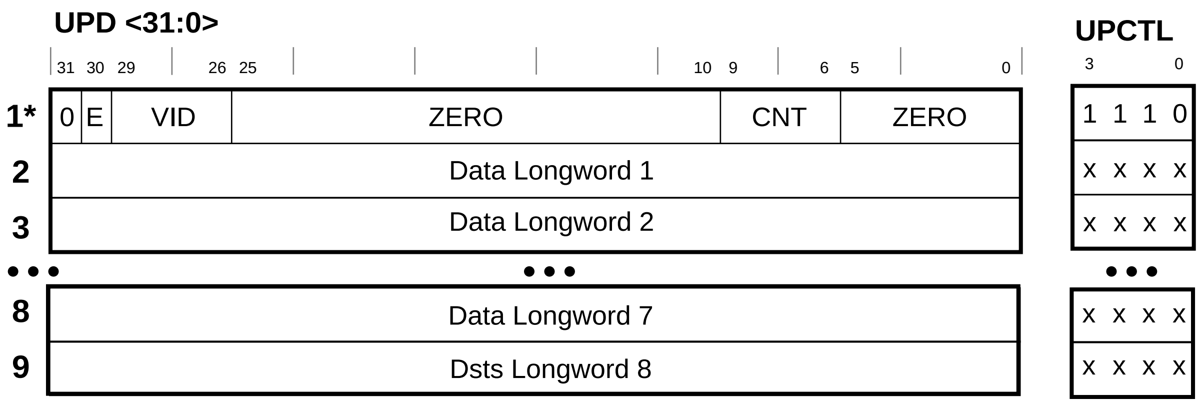
Cycle 1	Bits <31>	-	Zero
Cycle 1	Bits <30>	-	Error bit, set if there is an error during the transaction
Cycle 1	Bits <29:26>	-	Virtual ID of the TLSB commander initiating the transaction
Cycle 1	Bits <26:10>	-	Zero
Cycle 1	Bits <9:6>	-	Count <3:0> = 0100 for DWLPA and 0110 for DWLPB. Indicates the number of transactions the adapter can queue.
Cycle 1	Bits <5:0>	-	Zero

Cycle 2 UPCTL <3:0> = Don't care
Cycle 2 Bits <31:0> - Data Longword 1 read

Cycle 3 UPCTL <3:0> = Don't care
Cycle 3 Bits <31:0> - Data Longword 2 read

BXB-0647b-94

Figure A-15 CSR Dense Read Return Packet



* = Clock cycle

Cycle 1 UPCTL <3:0> = 1110

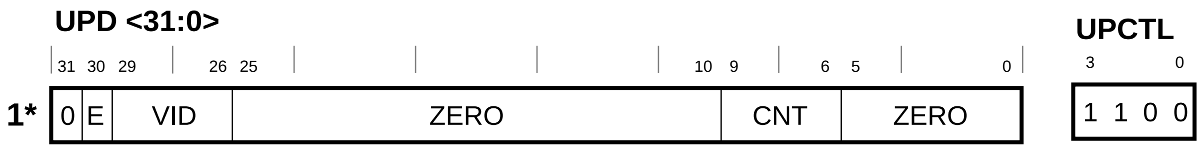
Cycle 1	Bits <31>	-	Zero
Cycle 1	Bits <30>	-	Error bit. Set if there is an error during the transaction
Cycle 1	Bits <29:26>	-	Virtual ID of the TLSB commander initiating the transaction
Cycle 1	Bits <26:10>	-	Zero
Cycle 1	Bits <9:6>	-	Count <3:0> = 0100 for DWLPA and 0110 for DWLPB. Indicates the number of transactions the adapter can queue.
Cycle 1	Bits <5:0>	-	Zero

Cycle 2 - 9 UPCTL <3:0> = Don't care

Cycle 2 - 9 Bits <31:0> - Data Longword 1 - 8 read

BXB-0647c-94

Figure A-16 CSR Write Status Return Packet



* = Clock cycle

Cycle 1 UPCTL <3:0> = 1110

Cycle 1	Bits <31>	-	Zero
Cycle 1	Bits <30>	-	Error bit. Set if there is an error during the transaction
Cycle 1	Bits <29:26>	-	Virtual ID of the TLSB commander initiating the transaction
Cycle 1	Bits <26:10>	-	Zero
Cycle 1	Bits <9:6>	-	Count <3:0> = 0100 for DWLPA and 0110 for DWLPB. Indicates the number of transactions the adapter can queue.
Cycle 1	Bits <5:0>	-	Zero

BXB-0647d-94

Index

A

Addressing
  PCI, 4-2
  PCI to system bus, 2-13
  system bus to PCI, 2-2
  TLSB system, 2-1
Address burst order, 4-11
Address conversions, 4-3
Address decode, 4-9
Address mapping, 6-3
Address mapping, CPU to PCI, 2-4
Address map, TLSB, 2-1
Address parity errors, 4-11
Address phase, 4-5, 4-10
Address queuing, 4-10
Address translation, 2-17, 4-9
  direct mapped, 4-9
  scatter/gather, 4-9

B

Bit types, 3-4
Block diagram, PCI and EISA, 6-1
Bridge buffers, 6-4
Bridge interrupts, 6-4
Bus parking, default, 4-5

C

Cache coherency, 1-2
Cache, scatter/gather, 4-9
Commands
  Down Hose, 4-14
  PCI, 4-1
  PCI target, 4-7
Command decode, 4-4
Communication overview, 1-5
Configurability, 1-1
Configuration Cycle Type bits, 3-15
Configuration rules, 6-6
Contents of PCI Slots bits, 3-8
CPU command decode, 4-4
CPU request queue, 4-4
CPU to PCI address mapping, 2-4
CPU-initiated PCI commands, 4-1
CPU-initiated PCI cycles, 4-1
CPU-initiated transactions, 4-20
CSR address space, 2-13
CSR command errors, 5-2
CSR Dense Read packet, A-6
CSR Dense Read Return packet, A-13
CSR Dense Write packet, A-7
CSR Overrun Error bit, 3-20
CSR Parity Error bit, 3-20
CSR Sparse Read packet, A-4
CSR Sparse Read Return packet, A-12
CSR Sparse Write packet, A-5
CSR Transactions, 1-6
CSR Write Status Return packet, A-13
CTLx register, 3-13

D

Data phase, 4-5
Data transfers
  read, 4-12
  write, 4-11
Default bus parking, 4-5
Dense mapping, 2-4
Device interrupts, 4-23, 4-24
Device Interrupt ID bits, 3-30
Device Interrupt in Progress bit, 3-30
Device Interrupt Priority Level bits, 3-23
Device-initiated exclusive access, 4-13
Device-initiated PCI commands, 4-7
Device-initiated PCI cycles, 4-7
Diagnostic features, 4-27
DIAGx register, 3-25
DMA Data Return packet, A-2
DMA errors, 5-2
DMA Map RAM Invalid Entry Error bit, 3-19
DMA Map RAM Parity Error bit, 3-20
DMA Read packet, A-8
DMA Read Return Data Parity/Length Error
  bit, 3-20
DMA transactions, 1-5, 4-21
DMA Transfer Direction bit, 3-21
DMA Unmasked Write packet, A-10
DMA Write Mask packet, A-9
Down Hose, 4-14

Down Hose commands, 4-14
Down hose header cycle parity error, 5-1
Down Hose packets, A-1
Down Hose signals, 4-15

E

EISA bridge registers, 6-7
EISA bus lock, 6-5
EISA interrupts, 4-26
EISA subtractive decode window, 2-13
EISA support, 1-1
Enable New Map bit, 3-9
Error
  CSR overrun, 5-2
  CSR packet parity, 5-2
  DMA map RAM invalid entry, 5-3
  DMA map RAM parity, 5-3
  DMA read data return, 5-3
  DMA read data return parity/length, 5-2
  down hose header cycle parity, 5-1
  Mailbox illegal length, 5-2
  Mailbox parity, 5-2
  PCIA command overflow, 5-5
  PCIA illegal CSR address, 5-5
  PCIA map RAM parity, 5-5
  PCI address parity, 5-4
  PCI data parity, 5-4
  PCI disconnected master abort, 5-5
  PCI incremental latency exceeded, 5-5
  PCI nonexistent address, 5-4
  PCI SERR_L asserted, 5-5
  PCI target abort parity, 5-4
  PCI target disconnect, 5-4
  PCI write parity, 5-4
  Up Hose packet count, 5-6
Error categories, 5-1
Error interrupts, 4-23, 4-25
Error Interrupt Enable bit, 3-23
Error Interrupt in Progress bit, 3-30
Error Interrupt Pending bit, 3-28
Error Interrupt Priority Level bits, 3-23
Error Summary bit, 3-20
ERRVEC register, 3-34
ERRx register, 3-18
Exclusive access, 4-13
Exclusive access support, 4-7

F

FADRx register, 3-21
Failing PCI DMA Address bits, 3-21
Features, 1-1

G

GPRx register, 3-17

H

Hardware Address Extension Disable bit, 3-15
Hardware compatibilty, 1-1
Hexword
  reads, 4-6
  writes, 4-6
Hose interface, 4-14
Hose interrupt support, 4-23
HPC Gate Array Revision bits, 3-25

I

IACK register, 3-6
IBR register, 3-12
IMASKx register, 3-22
Incremental Latency Exceeded bit, 3-19
Interrrupt status packet, 4-24
Interrupts
  bridge, 6-4
  EISA, 4-26
  PCI, 4-23
  vectored, 4-24
Interrupt A Enable bit, 3-24
Interrupt A Pending bit, 3-28
Interrupt B Enable bit, 3-23
Interrupt B Pending bit, 3-28
Interrupt C Enable bit, 3-23
Interrupt C Pending bit, 3-28
Interrupt D Enable bit, 3-23
Interrupt D Pending bit, 3-28
Interrupt in Progress registers, 4-27
Interrupt masking, 4-26
Interrupt Pending registers, 4-27
Interrupt Status Return packet, A-1
Interrupt transactions, 1-7
Interrupt vector offsets, 3-35
Interrupt/IDENT packet, A-11
Invert Down Hose Parity bit, 3-25
IPENDx register, 3-27
IPROGx register, 3-29
IREAD packet, A-8
I/O Port Up Hose Buffers bits, 3-14
I/O space assignment, 2-2
I/O Space Hardware Address Extension bits, 3-15
I/O system block diagram, 1-2

L

Latency timeout, 4-13
Lockout prevention, 6-6
Longword
  reads, 4-5
  writes, 4-5

M

Mailbox Comand packet, A-3
Mailbox errors, 5-2
Mailbox Illegal Length Error bit, 3-20
Mailbox parity error, 5-2
Mailbox Parity Error bit, 3-20
Mailbox Status Return packet, A-11
Mailbox transactions, 1-6
Mapping
  dense, 2-4
  sparse, 2-5
Map RAM address space, 2-13
Map RAM invalid page errors, 4-10
Map RAM logic command, 4-19
Map RAM parity errors, 4-10
Map RAM structure, 4-18
Masking PCI interrupts, 4-26
Master abort, 4-7
Master Retry Limit bits, 3-16
Master timeout, 4-6
MCTL register, 3-11
Memory block boundary, 4-11
Memory Block Size bit, 3-15
Memory locks, 1-2
Memory Read Multiple Arbitration bit, 3-14
Memory read multiple command, 4-8
Memory Read Multiple Enable bit, 3-14
Memory Read Multiple Prefetch Size bit, 3-14
Memory Space Hardware Address Extension bits, 3-15
Memory space, PCI, 2-4, 2-15
Module Revision bits, 3-8
MRETRYx register, 3-16

P

Packet
  CSR Dense Read, A-6
  CSR Dense Read Return, A-13
  CSR Dense Write, A-7
  CSR Sparse Read, A-4
  CSR Sparse Read Return, A-12
  CSR Sparse Write, A-5
  CSR Write Status Return, A-13
  DMA Data Return, A-2
  DMA Read, A-8
  DMA Unmasked Write, A-10
  DMA Write Mask, A-9
  Interrupt Status Return, A-1
  Interrupt/IDENT, A-11
  IREAD, A-8
  Mailbox Command, A-3
  Mailbox Status Return, A-11
PCIA block diagram, 1-5
PCIA description, 1-3
PCIA errors, 5-5
PCIA Illegal CSR Address bit, 3-19
PCIA Information Base Repair Register, 3-12
PCIA interrupts, 4-23
PCIA Interrupt Acknowledge Register, 3-6
PCIA Map RAM Parity Error bit, 3-19
PCIA Module Control Register, 3-11
PCIA physical layout, 1-3
PCIA registers, 3-2
PCIA reserved addresses, 2-13
PCIA Slot Present Register, 3-7
PCIA Special Cycle Register, 3-5
PCIA TBIT Register, 3-9
PCIx Bus Control Register, 3-13
PCIx Diagnostic Register, 3-25
PCIx Error Interrupt Vector Register, 3-34
PCIx Error Summary Register, 3-18
PCIx Failing Address Register, 3-21
PCIx General Purpose Register, 3-17
PCIx Interrupt in Progress Register, 3-29
PCIx Interrupt Mask Register, 3-22
PCIx interrupt Pending Register, 3-27
PCIx Master Retry Limit Register, 3-16
PCIx Transaction Base Register, 3-33
PCIx Window Base Register, 3-32
PCIx Window Mask Register, 3-31
PCI addressing, 1-2, 4-2
PCI address decode, 4-9
PCI Address Parity Error bit, 3-19
PCI Arbitration Control bits, 3-13
PCI bus request, 4-4
PCI commands
  CPU-initiated, 4-1
  device-initiated, 4-7
PCI configuration errors, 5-6
PCI configuration space, 2-8
PCI Cut-through Enable bit, 3-15
PCI Cut-through Threshold bits, 3-15
PCI cycles
  CPU-initiated, 4-1
  device-initiated, 4-7
PCI Data Parity Error bit, 3-19
PCI Disconnected Master Abort Error bit, 3-20
PCI errors, 5-4
PCI Force Bad Parity bits, 3-26
PCI interface, 4-1
PCI interrupts, 4-23
PCI I/O space, 2-8
PCI master abort, 4-7
PCI master timeout, 4-6
PCI memory space, 2-4, 2-15
PCI Nonexistent Address Error bit, 3-19
PCI peer-to-peer errors, 5-6
PCI Reset bit, 3-15
PCI SERR_L bit, 3-19
PCI Target Abort Error bit, 3-19
PCI target commands, 4-7

PCI target disconnect, 4-6
PCI Target Disconnect Error bit, 3-19
PCI to EISA bridge hardware, 4-20
PCI to system bus addressing, 2-13
PCI window mask values, 2-15
PCI Write Parity Error bit, 3-19
PCI-EISA chip functions, 6-3
Physical Page Number bits, 3-36
PRESENT register, 3-7

Q

Quadword reads, 4-5
Quadword writes, 4-6

R

Reads
  hexword, 4-6
  longword, 4-5
  quadward, 4-5
Read data errors, 4-13
Read data transfers, 4-12
Read multiple command, 4-8
Read return buffer, 4-13
Register
  PCIA Information Base Repair, 3-12
  PCIA Interrupt Acknowledge, 3-6
  PCIA Module Control, 3-11
  PCIA Slot Present, 3-7
  PCIA Special Cycle, 3-5
  PCIA TBIT, 3-9
  PCIx Bus Control, 3-13
  PCIx Diagnostic, 3-25
  PCIx Error Interrupt Vector, 3-34
  PCIx Error Summary, 3-18
  PCIx Failing Address, 3-21
  PCIx General Purpose, 3-17
  PCIx Interrupt in Progress, 3-29
  PCIx Interrupt Mask, 3-22
  PCIx interrupt Pending, 3-27
  PCIx Master Retry Limit, 3-16
  PCIx Transaction Base, 3-33
  PCIx Window Base, 3-32
  PCIx Window Mask, 3-31
Register list, 3-2
Request queue, 4-4
Reserved addresses, 2-13

S

Scatter/gather cache, 4-9
Scatter/gather capability, 1-1
Scatter/Gather Enable bit, 3-32
Scatter/Gather Map RAM Size bits, 3-14
Scatter/Gather RAM Map Entry Format, 3-36
SCYCLE register, 3-5
Self-test, 1-1
Self-Test Passed LED bit, 3-11
Serial EEPROM, 4-20
Serial EEPROM Clock bit, 3-12
Serial EEPROM Receive Data bit, 3-12
Serial EEPROM Transmit Data bit, 3-12
Slot usage, 6-6
Software compatibilty, 1-1
Sparse mapping, 2-5
Standard I/O Present bit, 3-8
Subsystem block diagram, 6-1
Synchronization, Down Hose to PCI, 4-14
System bus to PCI addressing, 2-2

T

Target disconnect, 4-6
Target latency timeout, 4-13
TBASE_xx register, 3-33
TBIT register, 3-9
TLSB address map, 2-1
TLSB I/O, 1-2
TLSB system addressing, 2-1
Transactions
  CPU-initiated, 4-20
  CSR, 1-6
  DMA, 1-5, 4-21
  interrupt, 1-7
  mailbox, 1-6
Transaction termination, 4-6
Transfers
  masked, 4-4
  unmasked, 4-4
Translated Base Address bits, 3-33

U

Up Hose, 4-15
Up Hose arbitration, 4-17
Up Hose commands, 4-15
Up Hose command FIFO, 4-17
Up Hose Force Bad Parity bit, 3-25
Up Hose packets, A-8
Up Hose packet count error, 5-6
Up Hose packet status, 4-16
Up Hose programming, 4-19
Up Hose signals, 4-18

V

Valid bit, 3-36
Vectored interrupts, 4-24

W

WBASE_xx register, 3-32
Window Base Address bits, 3-32
Window Enable bit, 3-32

Window Mask bits, 3-31
WMASK_xx register, 3-31
Writes
  hexword, 4-6
  longword, 4-5
  quadrword, 4-6
Write data buffers, 4-11
Write data errors, 4-12
Write data transfers, 4-11

X

XBus, 6-1, 6-3