ADVANCE INFORMATION

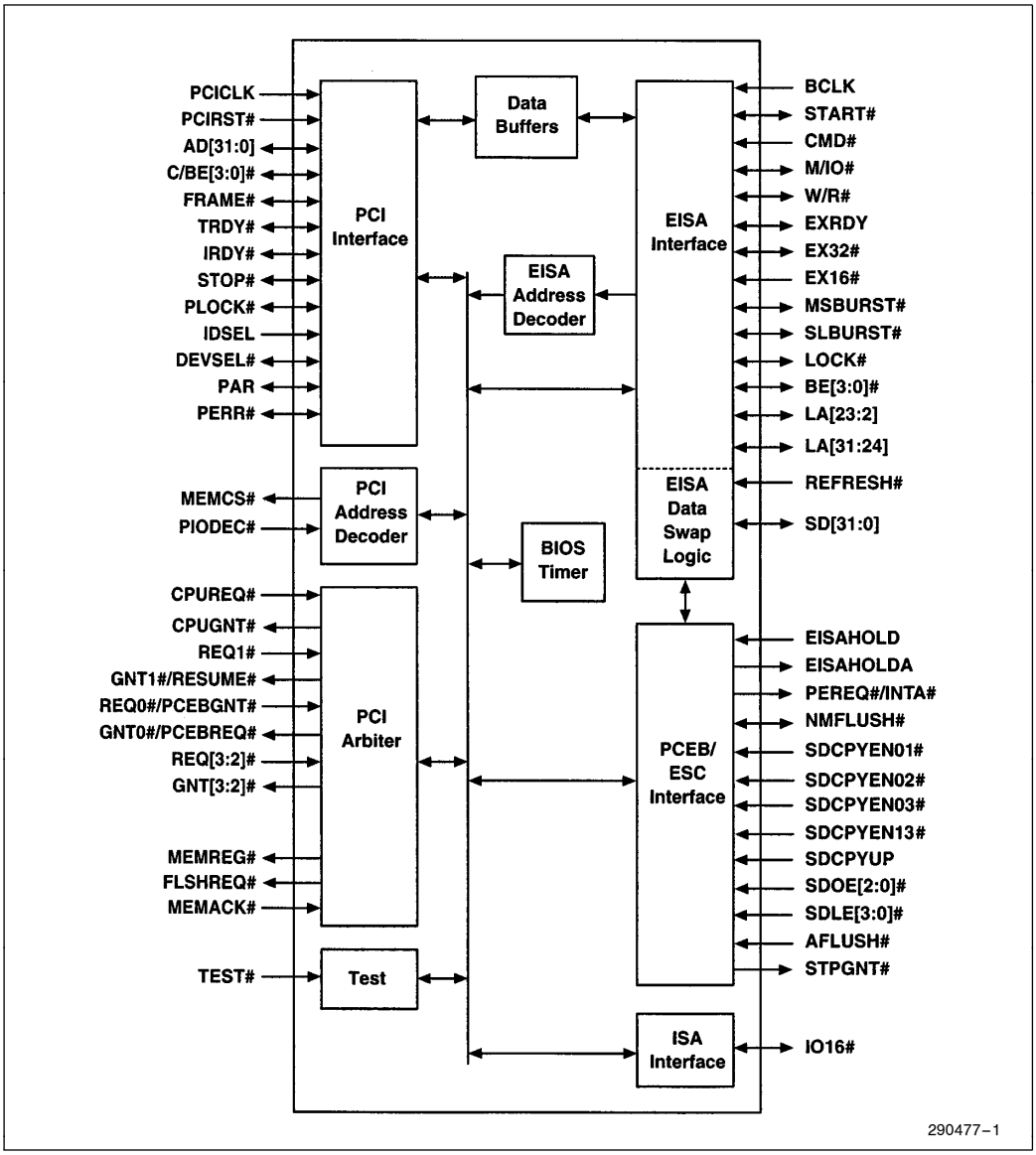
82375EB/82375SB PCI-EISA BRIDGE (PCEB)

- **Provides the Bridge Between the PCI Local Bus and EISA Bus**
- **100% PCI and EISA Compatible**
  - **— PCI and EISA Master/Slave Interface**
  - **— Directly Drives 10 PCI Loads and 8 EISA Slots**
  - **— Supports PCI from 25 to 33 MHz**
- **Data Buffers Improve Performance**
  - **— Four 32-bit PCI-to-EISA Posted Write Buffers**
  - **— Four 16-byte EISA-to-PCI Read/Write Line Buffers**
  - **— EISA-to-PCI Read Prefetch**
  - **— EISA-to-PCI and PCI-to-EISA Write Posting**
- **Data Buffer Management Ensures Data Coherency**
  - **— Flush Posted Write Buffers**
  - **— Flush or Invalidate Line Buffers**
  - **— System-Wide Data Buffer Coherency Control**
- **Burst Transfers on both the PCI and EISA Buses**
- **32-Bit Data Paths**
- **Integrated EISA Data Swap Buffers**
- **Arbitration for PCI Devices**
  - **— Supports Six PCI Masters**
  - **— Fixed, Rotating, or a Combination of the Two**
  - **— Supports External PCI Arbiter and Arbiter Cascading**
- **PCI and EISA Address Decoding and Mapping**
  - **— Positive Decode of Main Memory Areas (MEMCS# Generation)**
  - **— Four Programmable PCI Memory Space Regions**
  - **— Four Programmable PCI I/O Space Regions**
- **Programmable Main Memory Address Decoding**
  - **— Main Memory Sizes up to 512 MBytes**
  - **— Access Attributes for 15 Memory Segments in First 1 MByte of Main Memory**
  - **— Programmable Main Memory Hole**
- **Integrated 16-bit BIOS Timer**
- **Only Available as Part of a Supported Kit**

The 82375EB/SB PCI-EISA Bridge (PCEB) provides the master/slave functions on both the PCI Local Bus and the EISA Bus. Functioning as a bridge between the PCI and EISA buses, the PCEB provides the address and data paths, bus controls, and bus protocol translation for PCI-to-EISA and EISA-to-PCI transfers. Extensive data buffering in both directions increases system performance by maximizing PCI and EISA Bus efficiency and allowing concurrency on the two buses. The PCEB's buffer management mechanism ensures data coherency. The PCEB integrates central bus control functions including a programmable bus arbiter for the PCI Bus and EISA data swap buffers for the EISA Bus. Integrated system functions include PCI parity generation, system error reporting, and programmable PCI and EISA memory and I/O address space mapping and decoding. The PCEB also contains a BIOS Timer that can be used to implement timing loops. The PCEB is intended to be used with the EISA System Component (ESC) to provide an EISA I/O subsystem interface.

This document describes both the 82375EB and 82375SB components. Unshaded areas describe the 82375EB. Shaded areas, like this one, describe the 82375SB operations that differ from the 82375EB.

*Other brands and names are the property of their respective owners.
Information in this document is provided in connection with Intel products. Intel assumes no liability whatsoever, including infringement of any patent or copyright, for sale and use of Intel products except as provided in Intel's Terms and Conditions of Sale for such products. Intel retains the right to make changes to these specifications at any time, without notice. Microcomputer Products may have minor variations to this specification known as errata.
COPYRIGHT © INTEL CORPORATION, 1996
March 1996 Order Number: 290477-004



PCEB Simplified Block Diagram

82375EB/82375SB PCI-EISA BRIDGE (PCEB)

CONTENTS

PAGE

1.0 ARCHITECTURAL OVERVIEW 8
1.1 PCEB Overview 10
1.2 ESC Overview 12

2.0 SIGNAL DESCRIPTION 14
2.1 PCI Bus Interface Signals 15
2.2 PCI Arbiter Signals 18
2.3 Address Decoder Signals 19
2.4 EISA Interface Signals 20
2.5 ISA Interface Signals 23
2.6 PCEB/ESC Interface Signals 24
2.7 Test Signal 26

3.0 REGISTER DESCRIPTION 27
3.1 Configuration Registers 27
3.1.1 VID—VENDOR IDENTIFICATION REGISTER 29
3.1.2 DID—DEVICE IDENTIFICATION REGISTER 29
3.1.3 PCICMD—PCI COMMAND REGISTER 30
3.1.4 PCISTS—PCI STATUS REGISTER 31
3.1.5 RID—REVISION IDENTIFICATION REGISTER 31
3.1.6 MLT—MASTER LATENCY TIMER REGISTER 32
3.1.7 PCICON—PCI CONTROL REGISTER 32
3.1.8 ARBCON—PCI ARBITER CONTROL REGISTER 33
3.1.9 ARBPRI—PCI ARBITER PRIORITY CONTROL REGISTER 34
3.1.10 ARBPRIX—PCI ARBITER PRIORITY CONTROL EXTENSION REGISTER 35
3.1.11 MCSCON—MEMCS# CONTROL REGISTER 35
3.1.12 MCSBOH—MEMCS# BOTTOM OF HOLE REGISTER 36
3.1.13 MCSTOH—MEMCS# TOP OF HOLE REGISTER 37
3.1.14 MCSTOM—MEMCS# TOP OF MEMORY REGISTER 37
3.1.15 EADC1—EISA ADDRESS DECODE CONTROL 1 REGISTER 38
3.1.16 IORT—ISA I/O RECOVERY TIMER REGISTER 39
3.1.17 MAR1—MEMCS# ATTRIBUTE REGISTER #1 40
3.1.18 MAR2—MEMCS# ATTRIBUTE REGISTER #2 40
3.1.19 MAR3—MEMCS# ATTRIBUTE REGISTER #3 41
3.1.20 PDCON—PCI DECODE CONTROL REGISTER 42
3.1.21 EADC2—EISA ADDRESS DECODE CONTROL EXTENSION REGISTER 43

CONTENTS

PAGE

3.1.22 EPMRA—EISA-TO-PCI MEMORY REGION ATTRIBUTES REGISTER 44
3.1.23 MEMREGN[4:1]—EISA-TO-PCI MEMORY REGION ADDRESS REGISTERS 45
3.1.24 IOREGN[4:1]—EISA-TO-PCI I/O REGION ADDRESS REGISTERS 46
3.1.25 BTMR—BIOS TIMER BASE ADDRESS REGISTER 46
3.1.26 ELTCR—EISA LATENCY TIMER CONTROL REGISTER 47
3.2 I/O Registers 47
3.2.1 BIOSTM—BIOS TIMER REGISTER 47

4.0 ADDRESS DECODING 48
4.1 PCI Cycle Address Decoding 50
4.1.1 MEMORY SPACE ADDRESS DECODING 51
4.1.1.1 Main Memory Decoding (MEMCS#) 51
4.1.1.2 BIOS Memory Space 54
4.1.1.3 Subtractively And Negatively Decoded Cycles To EISA 54
4.1.2 PCEB CONFIGURATION REGISTERS 56
4.1.3 PCEB I/O REGISTERS 56
4.1.4 POSITIVELY DECODED COMPATIBILITY I/O REGISTERS 56
4.1.4.1 ESC Resident PIC Registers 57
4.1.4.2 EISA Resident IDE Registers 57
4.2. EISA Cycle Address Decoding 58
4.2.1 POSITIVELY DECODED MEMORY CYCLES TO MAIN MEMORY 58
4.2.2 PROGRAMMABLE EISA-TO-PCI MEMORY ADDRESS REGIONS 61
4.2.3 PROGRAMMABLE EISA-TO-PCI I/O ADDRESS REGIONS 61
4.2.4 EXTERNAL EISA-TO-PCI I/O ADDRESS DECODER 62
4.3 Palette DAC Snoop Mechanism 62

5.0 PCI INTERFACE 62
5.1 PCI Bus Transactions 63
5.1.1 PCI COMMAND SET 63
5.1.2 PCI CYCLE DESCRIPTIONS 64
5.1.2.1 Interrupt Acknowledge 64
5.1.2.2 Special Cycle 65
5.1.2.3 I/O Read 65
5.1.2.4 I/O Write 65
5.1.2.5 Memory Read 66
5.1.2.6 Memory Write 67
5.1.2.7 Configuration Read, Configuration Write 67
5.1.2.8 Memory Read Multiple 68

CONTENTS

PAGE

5.1.2.9 Memory Read Line 68
5.1.2.10 Memory Write And Invalidate 68
5.1.3 PCI TRANSFER BASICS 68
5.1.3.1 Turn-Around-Cycle Definition 69
5.1.3.2 Idle Cycle Definition 69
5.1.4 BASIC READ 71
5.1.5 BASIC WRITE 72
5.1.6 CONFIGURATION CYCLES 73
5.1.7 INTERRUPT ACKNOWLEDGE CYCLE 74
5.1.8 EXCLUSIVE ACCESS 75
5.1.9 DEVICE SELECTION 77
5.1.10 TRANSACTION TERMINATION 78
5.1.10.1 Master Initiated Termination 78
5.1.10.2 Target Initiated Termination 79
5.1.10.3 PCEB Target Termination Conditions 81
5.1.10.4 PCEB Master Termination Conditions 81
5.1.10.5 PCEB Responses/Results Of Termination 81
5.1.11 PCI DATA TRANSFERS WITH SPECIFIC BYTE ENABLE COMBINATIONS 82
5.2 PCI Bus Latency 82
5.2.1 MASTER LATENCY TIMER (MLT) 82
5.2.2 INCREMENTAL LATENCY MECHANISM 83
5.3 PCI Bus Parity Support And Error Reporting 83
5.3.1 PARITY GENERATION AND CHECKING 83
5.3.1.1 Address Phase 84
5.3.1.2 Data Phase 84
5.3.2 PARITY ERROR—PERR# SIGNAL 84
5.3.3 SYSTEM ERRORS 84
5.4 PCI Bus Arbitration 85
5.4.1 PCI ARBITER CONFIGURATION 85
5.4.1.1 Fixed Priority Mode 87
5.4.1.2 Rotating Priority Mode 89
5.4.1.3 Mixed Priority Mode 89
5.4.1.4 Locking Masters 89
5.4.2 ARBITRATION SIGNALING PROTOCOL 89
5.4.2.1 REQ# and GNT# Rules 90
5.4.2.2 Back-to-Back Transactions 90

CONTENTS

PAGE

5.4.3 RETRY THRASHING RESOLVE 91
5.4.3.1 Resume Function 91
5.4.3.2 Master Retry Timer 92
5.4.4 BUS LOCK MODE 92
5.4.5 MEMREQ#, FLSHREQ#, AND MEMACK# PROTOCOL 92
5.4.5.1 Flushing System Posted Write Buffers 93
5.4.5.2 Guaranteed Access Time Mode 94
5.4.5.3 Interrupt Synchronization-Buffer Flushing 95
5.4.6 BUS PARKING 95
5.4.7 PCI ARBITRATION AND PCEB/ESC EISA OWNERSHIP EXCHANGE 96
5.4.7.1 GAT Mode and PEREQ# Signaling 96
5.4.7.2 PCI Retry and EISA Latency Timer (ELT) Mechanism 96

6.0 DATA BUFFERING 97
6.1 Line Buffers 98
6.1.1 WRITE STATE 98
6.1.2 READ STATE 99
6.2 Buffer Management Summary 100

7.0 EISA INTERFACE 101
7.1 PCEB As An EISA Master 102
7.1.1 STANDARD EISA MEMORY AND I/O READ/WRITE CYCLES 102
7.1.2 EISA BACK-OFF CYCLE 103
7.2 PCEB As An EISA Slave 105
7.2.1 EISA MEMORY AND I/O READ/WRITE CYCLES 106
7.2.2 EISA MEMORY BURST CYCLES 107
7.3 I/O Recovery 108

8.0 EISA DATA SWAP BUFFERS 109
8.1 Data Assembly And Disassembly 109
8.2 The Copy Operation (Up Or Down) 110
8.3 The Re-Drive Operation 111

9.0 BIOS TIMER 114
9.1 Bios Timer Operations 114

CONTENTS

PAGE

10.0 PCEB/ESC INTERFACE 115
10.1 Arbitration Control Signals 115
10.2 System Buffer Coherency Control-APIC 117
10.3 Power Management (82375SB) 117
10.4 EISA Data Swap Buffer Control Signals 117
10.5 Interrupt Acknowledge Control 118

11.0 ELECTRICAL CHARACTERISTICS 118
11.1 Absolute Maximum Ratings 118

12.0 PINOUT AND PACKAGE INFORMATION 119
12.1 Pin Assignment 119
12.2 Package Characteristics 127

13.0 TESTABILITY 128
13.1 NAND Tree 128

1.0 ARCHITECTURAL OVERVIEW

The PCI-EISA bridge chip set provides an I/O subsystem core for the next generation of high-performance personal computers (e.g., those based on the Intel486™ or Pentium® processors). System designers can take advantage of the power of the PCI local bus while maintaining access to the large base of EISA and ISA expansion cards, and corresponding software applications. Extensive buffering and buffer management within the PCI-EISA bridge ensures maximum efficiency in both bus environments.

The chip set consists of two components—the 82375EB PCI-EISA Bridge (PCEB) and the 82374EB EISA System Component (ESC). These components work in tandem to provide an EISA I/O subsystem interface for personal computer platforms based on the PCI standard. This section provides an overview of the PCI and EISA Bus hierarchy followed by an overview of the PCEB and ESC components.

Bus Hierarchy—Concurrent Operations

Figure 1 shows a block diagram of a typical system using the PCI-EISA Bridge chip set. The system contains three levels of buses structured in the following hierarchy:

- Host Bus as the execution bus
- PCI Bus as a primary I/O bus
- EISA Bus as a secondary I/O bus

PCI Bus

The PCI Bus has been defined to address the growing industry needs for a standardized *local bus* that is not directly dependent on the speed and the size of the processor bus. New generations of personal computer system software such as Windows™ and Win-NT™ with sophisticated graphical interfaces, multi-tasking and multi-threading bring new requirements that traditional PC I/O architectures can not satisfy. In addition to the higher bandwidth, reliability and robustness of the I/O subsystem are becoming increasingly important. The PCI environment addresses these needs and provides an upgrade path for the future. PCI features include:

- Processor independent
- Multiplexed, burst mode operation
- Synchronous up to 33 MHz
- 120 MByte/sec usable throughput (132 MByte/sec peak) for 32-bit data path
- 240 MByte/sec usable throughput (264 MByte/sec peak) for 64-bit data path
- Optional 64-bit data path with operations that are transparent with the 32-bit data path
- Low latency random access (60 ns write access latency to slaves from a master parked on the bus)
- Capable of full concurrency with processor/memory subsystem
- Full multi-master capability allowing any PCI master peer-to-peer access to any PCI slave
- Hidden (overlapped) central arbitration
- Low pin count for cost effective component packaging (multiplexed address/data)
- Address and data parity
- Three physical address spaces: memory, I/O, and configuration
- Comprehensive support for autoconfiguration through a defined set of standard configuration functions

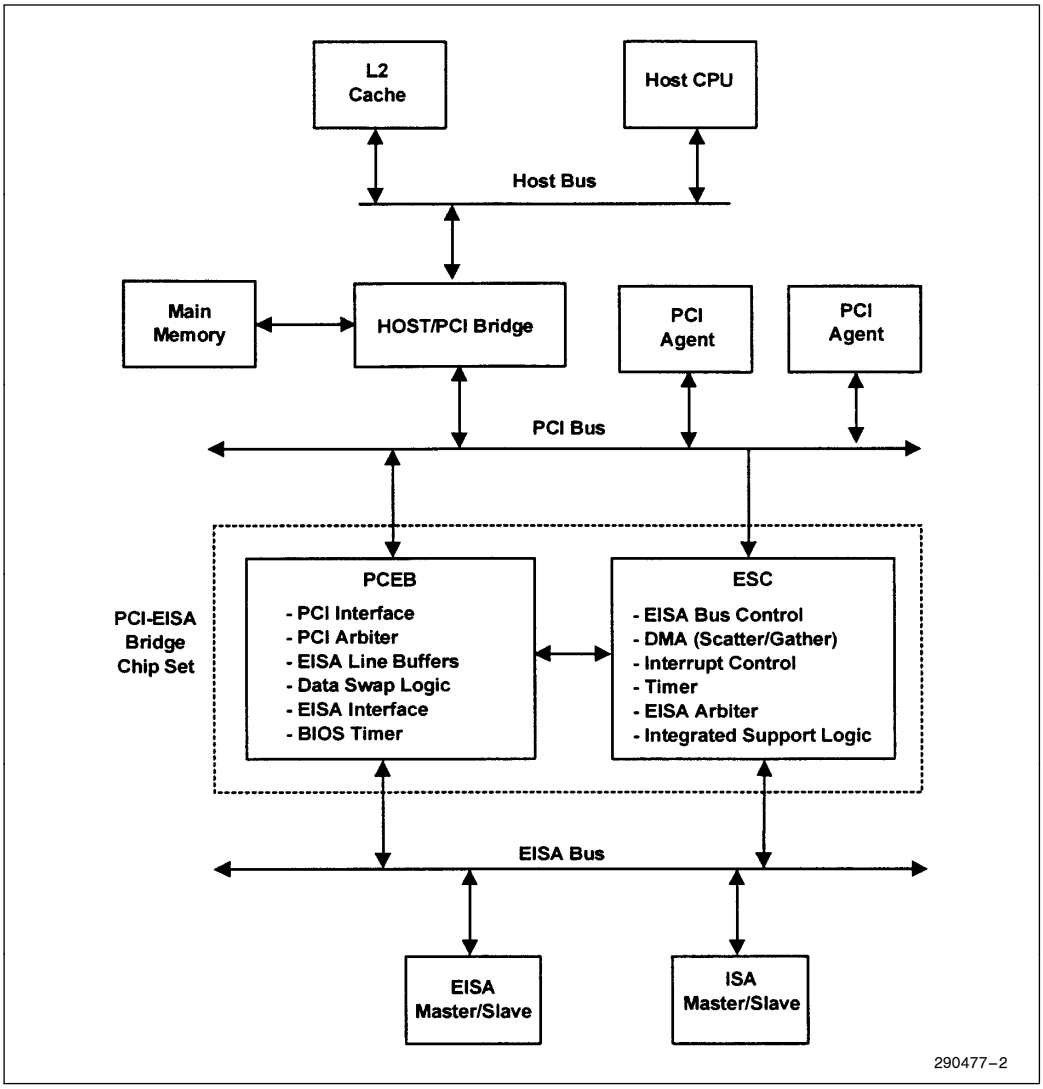


Figure 1. PCI-EISA System Diagram

System partitioning shown in Figure 1 illustrates how the PCI can be used as a common interface between different portions of a system platform that are typically supplied by the chip set vendor. These portions are the Host/PCI Bridge (including a main memory DRAM controller and an optional secondary cache controller) and the PCI-EISA Bridge. Thus, the PCI allows a system I/O core design to be decoupled from the processor/

memory treadmill, enabling the I/O core to provide maximum benefit over multiple generations of processor/memory technology. For this reason, the PCI-EISA Bridge can be used with different processors (i.e. derivatives of the Intel486 CPU or the new generation processors, such as the Pentium processor.) Regardless of the new requirements imposed on the processor side of the Host/PCI Bridge (e.g. 64-bit data path, 3.3V interface, etc.) the PCI side remains unchanged. This standard PCI environment allows reusability, not only of the rest of the platform chip set (i.e. PCI-EISA Bridge), but also of all other I/O functions interfaced at the PCI level. These functions typically include graphics, SCSI, and LAN.

EISA Bus

The EISA bus in the system shown in the Figure 1 represents a second level I/O bus. It allows personal computer platforms built around the PCI as a primary I/O bus to leverage the large EISA/ISA product base. Combinations of PCI and EISA buses, both of which can be used to provide expansion functions, will satisfy even the most demanding applications.

Along with compatibility for 16-bit and 8-bit ISA hardware and software, the EISA bus provides the following key features:

- 32-bit addressing and 32-bit data path
- 33 MByte/sec bus bandwidth
- Multiple bus master support through efficient arbitration
- Support for autoconfiguration

Integrated Bus Central Control Functions

The PCI-EISA Bridge chip set integrates central bus functions on both the PCI and EISA Buses. For PCI, the functions include PCI bus arbitration and the default bus driver. For the EISA Bus, central functions include the EISA Bus controller and EISA arbiter that are integrated in the ESC and EISA data swap buffers that are integrated in the PCEB.

Integrated System Functions

The PCI-EISA Bridge chip set integrates system functions including PCI parity and system error reporting, buffer management, PCI and EISA memory and I/O address space mapping and decoding. For maximum flexibility, all of these functions are programmable allowing for variety of optional features.

1.1 PCEB Overview

The PCEB and ESC form a PCI-EISA Bridge chip set. The PCEB/ESC interface provides the inter-chip communications between these two devices. The major functions provided by the PCEB are described in this section.

PCI Bus Interface

The PCEB can be either a master or slave on the PCI Bus and supports bus frequencies from 25-to-33 MHz. The PCEB becomes a slave when it positively decodes a PCI cycle. The PCEB also becomes a slave for unclaimed cycles on the PCI Bus. These unclaimed cycles are subtractively decoded by the PCEB and forwarded to the EISA Bus. As a slave, the PCEB supports single cycle transfers for memory, I/O, and configuration operations.

For EISA-initiated transfers to the PCI Bus, the PCEB is a PCI master. The PCEB permits EISA devices to access either PCI memory or I/O. While all PCI I/O transfers are single cycle, PCI memory cycles can be either single cycle or burst, depending on the status of the PCEB's Line Buffers. During EISA reads of PCI memory, the PCEB uses a burst read cycle of four Dwords to prefetch data into a Line Buffer. During EISA-to-PCI memory writes, the PCEB uses PCI burst cycles to flush the Line Buffers. The PCEB contains a programmable Master Latency Timer that provides the PCEB with a guaranteed time slice on the PCI Bus, after which it surrenders the bus.

As a master on the PCI Bus, the PCEB generates address and command signals (C/BE[3:0]#), address parity for read and write cycles, and data parity for write cycles. As a slave, the PCEB generates data parity for read cycles. Parity checking is not supported.

The PCEB, as a resource, can be locked by any PCI master. In the context of locked cycles, the entire PCEB subsystem (including the EISA Bus) is considered a single resource.

PCI Bus Arbitration

The PCI arbiter supports six PCI masters—the Host/PCI bridge, PCEB, and four other PCI masters. The arbiter can be programmed for twelve fixed priority schemes, a rotating scheme, or a combination of the fixed and rotating schemes. The arbiter can be programmed for bus parking that permits the Host/PCI Bridge default access to the PCI Bus when no other device is requesting service. The arbiter also contains an efficient PCI retry mechanism to minimize PCI Bus thrashing when the PCEB generates a retry.

EISA Bus Interface

The PCEB contains a fully EISA-compatible master and slave interface. The PCEB directly drives eight EISA slots without external data or address buffering. The PCEB is only a master or slave on the EISA Bus for transfers between the EISA Bus and PCI Bus. For transfers contained to the EISA Bus, the PCEB is never a master or slave. However, the data swap buffers contained in the PCEB are involved in these transfers, if data size translation is needed. The PCEB also provides support for I/O recovery.

EISA/ISA masters and DMA can access PCI memory or I/O. The PCEB only forwards EISA cycles to the PCI Bus if the address of the transfer matches one of the address ranges programmed into the PCEB for EISA-to-PCI positive decode. This includes the main memory segments used for generating MEMCS# from the EISA Bus, one of the four programmable memory regions, or one of the four programmable I/O regions. For EISA-initiated accesses to the PCI Bus, the PCEB is a slave on the EISA Bus. I/O accesses are always non-buffered and memory accesses can be either non-buffered or buffered via the Line Buffers. For buffered accesses, burst cycles are supported.

During PCI-initiated cycles to the EISA Bus, the PCEB is an EISA master. Single cycle transfers are used for I/O and memory read/write cycles from PCI to EISA.

PCI/EISA Address Decoding

The PCEB contains two address decoders—one to decode PCI-initiated cycles and the other to decode EISA-initiated cycles. The two decoders permit the PCI and EISA Buses to operate concurrently.

The PCEB can also be programmed to provide main memory address decoding on behalf of the Host/PCI bridge. When programmed, the PCEB monitors the PCI and EISA bus cycle addresses, and generates a memory chip select signal (MEMCS#) indicating that the current cycle is targeted to main memory residing behind the Host/PCI bridge. Programmable features include, read/write attributes for specific memory segments and the enabling/disabling of a memory hole. If not used, the MEMCS# feature can be disabled.

In addition to the main memory address decoding, there are four programmable memory regions and four programmable I/O regions for EISA-initiated cycles. EISA/ISA master or DMA accesses to one of these regions are forwarded to the PCI Bus.

Data Buffering

The PCEB contains four 16-byte wide Line Buffers for EISA-initiated cycles to the PCI Bus. The Line Buffers permit prefetching of read data from PCI memory and posting of data being written to PCI memory.

By using burst transactions to fill or flush these buffers, when appropriate, the PCEB maximizes bus efficiency. For example, an EISA device could fill a Line Buffer with byte, word, or Dword transfers and the PCEB would use a PCI burst cycle to flush the filled line to PCI memory.

BIOS Timer

The PCEB has a 16-bit BIOS Timer. The timer can be used by BIOS software to implement timing loops. The timer count rate is derived from the EISA clock (BCLK) and has an accuracy of $\pm$ 1 μs.

1.2 ESC Overview

The PCEB and ESC form a PCI-EISA bridge. The PCEB/ESC interface provides the inter-chip communications between these two devices. The major functions provided by the ESC are described in this section.

EISA Controller

The ESC incorporates a 32-bit master and an 8-bit slave. The ESC directly drives eight EISA slots without external data or address buffering. EISA system clock (BCLK) generation is integrated by dividing the PCI clock (divide by 3 or divide by 4) and wait state generation is provided. The AENx and MACKx signals provide a direct interface to four EISA slots and supports eight EISA slots with encoded AENx and MACKx signals.

The ESC contains an 8-bit data bus (lower 8 bits of the EISA data bus) that is used to program the ESC's internal registers. Note that for transfers between the PCI and EISA Buses, the PCEB provides the data path. Thus, the ESC does not require a full 32-bit data bus. A full 32-bit address bus is provided and is used during refresh cycles and for DMA operations.

The ESC performs cycle translation between the EISA Bus and ISA Bus. For mis-matched master/slave combinations, the ESC controls the data swap buffers that are located in the PCEB. This control is provided through the PCEB/ESC interface.

DMA Controller

The ESC incorporates the functionality of two 82C37 DMA controllers with seven independently programmable channels. Each channel can be programmed for 8- or 16-bit DMA device size, and ISA-compatible, type ''A'', type ''B'', or type ''C'' timings. Full 32-bit addressing is provided. The DMA controller also generates refresh cycles.

The DMA controller supports an enhanced feature called scatter/gather. This feature provides the capability of transferring multiple buffers between memory and I/O without CPU intervention. In scatter/gather mode, the DMA can read the memory address and word count from an array of buffer descriptors, located in main memory, called the scatter/gather descriptor (SGD) table. This allows the DMA controller to sustain DMA transfers until all of the buffers in the SGD table are handled.

Interrupt Controller

The ESC contains an EISA compatible interrupt controller that incorporates the functionality of two 82C59 Interrupt Controllers. The two interrupt controllers are cascaded providing 14 external and two internal interrupts.

Advanced Programmable Interrupt Controller (APIC)

In addition to the standard EISA compatible interrupt controller described above, the ESC incorporates the Advanced Programmable Interrupt Controller (APIC). While the standard interrupt controller is intended for use in a uni-processor system, APIC can be used in either a uni-processor or multi-processor system. APIC provides multi-processor interrupt management and incorporates both static and dynamic symmetric interrupt distribution across all processors. In systems with multiple I/O subsystems, each subsystem can have its own set of interrupts.

Timer/Counter

The ESC provides two 82C54 compatible timers (Timer 1 and Timer 2). The counters in Timer 1 support the system timer interrupt (IRQ0#), refresh request, and a speaker tone output (SPKR). The counters in Timer 2 support fail-safe time-out functions and the CPU speed control.

Integrated Support Logic

To minimize the chip count for board designs, the ESC incorporates a number of extended features. The ESC provides support for ALTA20 (Fast A20GATE) and ALTRST with I/O Port 92h. The ESC generates the control signals for SA address buffers and X-Bus buffer. The ESC also provides chip selects for BIOS, the keyboard controller, the floppy disk controller, and three general purpose devices. Support for generating chip selects with an external decoder is provided for IDE, a parallel port, and a serial port. The ESC provides support for a PC/AT compatible coprocessor interface and IRQ13 generation.

Power Management

Extensive power management capability permits a system to operate in a low power state without being powered down. Once in the low power state (called ''Fast Off'' state), the computer appears to be off. For example, the SMM code could turn off the CRT, line printer, hard disk drive's spindle motor, and fans. In addition, the CPU's clock can be governed. To the user, the machine appears to be in the off state. However, the system is actually in an extremely low power state that still permits the CPU to function and maintain communication connections normally associated with today's desktops (e.g., LAN, Modem, or FAX). Programmable options provide power management flexibility. For example, various system events can be programmed to place the system in the low power state or break events can be programmed to wake the system up.

2.0 SIGNAL DESCRIPTION

This section provides a detailed description of each signal. The signals are arranged in functional groups according to their associated interface.

The ''#'' symbol at the end of a signal name indicates that the active, or asserted state occurs when the signal is at a low voltage level. When ''#'' is not present after the signal name, the signal is asserted when at the high voltage level.

The terms assertion and negation are used extensively. This is done to avoid confusion when working with a mixture of ''active-low'' and ''active-high'' signals. The term **assert**, or **assertion** indicates that a signal is active, independent of whether that level is represented by a high or low voltage. The term **negate**, or **negation** indicates that a signal is inactive.

The following notations are used to describe the signal type.

in Input is a standard input-only signal

out Totem Pole output is a standard active driver

o/d Open Drain input/output

t/s Tri-State is a bi-directional, tri-state input/output pin

s/t/s Sustained Tri-State is an active low tri-state signal owned and driven by one and only one agent at a time. The agent that drives a s/t/s pin low must drive it high for at least one clock before letting it float. A new agent can not start driving a s/t/s signal any sooner than one clock after the previous owner tri-states it. An external pull-up is required to sustain the inactive state until another agent drives it and must be provided by the central resource.

2.1 PCI Bus Interface Signals

Pin Name	Type	Description
PCICLK	in	**PCI CLOCK:** PCICLK provides timing for all transactions on the PCI Bus. All other PCI signals are sampled on the rising edge of PCICLK and all timing parameters are defined with respect to this edge. Frequencies supported by the PCEB range from 25 to 33 MHz.
PCIRST#	in	**PCI RESET:** PCIRST# forces the PCEB into a known state. All t/s and s/t/s signals are forced to a high impedance state, and the s/o/d signals are allowed to float high. The PCEB negates all GNT# lines to the PCI Bus and the PCEB negates its internal request. The PCEB drives AD[31:0], C/BE[3:0]#, and PAR during reset to keep these signals from floating (depending on the state of CPUREQ# and REQ1#—as described in the following paragraph). As long as PCIRST# is asserted, the PCEB drives the AD[31:0] signals to keep them from floating. Note that CPUREQ# must be sampled high when PCIRST# is asserted. All PCEB registers are set to their default values. PCIRST# may be asynchronous to PCICLK when asserted or negated. Although asynchronous, the negation of PCIRST# must be a clean, bounce-free edge. PCIRST# must be asserted for a minimum 1 μs, and PCICLK must be active during the last 100 μs of the PCIRST# pulse.
AD[31:0]	t/s	**ADDRESS AND DATA:** AD[31:0] is a multiplexed address and data bus. During the first clock of a transaction, AD[31:0] contain a physical address. During subsequent clocks, AD[31:0] contain data. A PCEB bus transaction consists of an address phase followed by one or more data phases. Little-endian byte ordering is used. AD[7:0] define the least significant byte (LSB) and AD[31:24] the most significant byte (MSB). The information contained in the two low order address bits varies by address space. In the I/O address space, AD[1:0] are used to provide full byte address. In the memory and configuration address space, AD[1:0] are driven ‘‘00’’ during the address phase. The other three encodings are reserved. See Section 5.0, PCI Interface for more details. When the PCEB is a target, AD[31:0] are inputs during the address phase of a transaction. During the following data phase(s), the PCEB may be asked to supply data on AD[31:0] as for a PCI read, or accept data as for a PCI write. As an Initiator, the PCEB drives a valid address on AD[31:0] (with exceptions related to AD[1:0]) during the address phase, and drives write or latches read data on AD[31:0] during the data phase. When PCIRST# is asserted, the PCEB drives the AD[31:0] signals to keep them from floating. In addition, the PCEB acts as the central resource responsible for driving the AD[31:0] signals when no device owns the PCI Bus and the bus is idle.

Pin Name	Type	Description
C/BE[3:0]#	t/s	**BUS COMMAND AND BYTE ENABLES:** The command and byte enable signals are multiplexed on the same PCI pins. During the address phase of a transaction, C/BE[3:0]# define the bus command for bus command definitions. During the data phase, C/BE[3:0]# are used as Byte Enables. The Byte Enables determine which byte lanes carry meaningful data. C/BE[0]# applies to byte 0 and C/BE[3]# to byte 3. C/BE[3:0]# are not used for address decoding. The PCEB drives C/BE[3:0]# as an initiator of a PCI Bus cycle and monitors C/BE[3:0]# as a target. When PCIRST# is asserted, the PCEB drives C/BE[3:0]# to keep them from floating. In addition, the PCEB acts as the central resource responsible for driving the C/BE[3:0]# signals when no device owns the PCI Bus and the bus is idle
FRAME#	s/t/s	**FRAME:** FRAME# is driven by the current initiator to indicate the beginning and duration of an access. FRAME# is asserted to indicate that a bus transaction is beginning. During a transaction, data transfers continue while FRAME# is asserted. When FRAME# is negated, the transaction is in the final data phase. FRAME# is an input when the PCEB is the target. FRAME# is an output when the PCEB is the initiator. During reset, this signal is tri-stated.
TRDY#	s/t/s	**TARGET READY:** TRDY#, as an output, indicates the target's ability to complete the current data phase of the transaction. TRDY# is used in conjunction with IRDY#. A data phase is completed on any clock that both TRDY# and IRDY# are sampled asserted. When PCEB is the target during a read cycle, TRDY# indicates that the PCEB has valid data present on AD[31:0]. During a write, it indicates that the PCEB, as a target, is prepared to latch data. TRDY # is an input to the PCEB when the PCEB is the initiator. During reset, this signal is tri-stated.
IRDY#	s/t/s	**INITIATOR READY:** IRDY#, as an output, indicates the initiator's ability to complete the current data phase of the transaction. IRDY# is used in conjunction with TRDY#. A data phase is completed on any clock that both IRDY# and TRDY# are sampled asserted. When PCEB is the initiator of a write cycle, IRDY# indicates that the PCEB has valid data present on AD[31:0]. During a read, it indicates the PCEB is prepared to latch data. IRDY# is an input to the PCEB when the PCEB is the target. During reset, this signal is tri-stated.
STOP#	s/t/s	**STOP:** As a target, the PCEB asserts STOP# to request that the master stop the current transaction. When the PCEB is an initiator, STOP# is an input. As an initiator, the PCEB stops the current transaction when STOP# is asserted. Different semantics of the STOP# signal are defined in the context of other handshake signals (TRDY# and DEVSEL#). During reset, this signal is tri-stated.
PLOCK#	s/t/s	**PCI LOCK:** PLOCK# indicates an atomic operation that may require multiple transactions to complete. PLOCK# is an input when PCEB is the target and output when PCEB is the initiator. When PLOCK# is sampled negated during the address phase of a transaction, a PCI agent acting as a target will consider itself a locked resource until it samples PLOCK# and FRAME# negated. When other masters attempt accesses to the PCEB (practically to the EISA subsystem) while the PCEB is locked, the PCEB responds with a retry termination. During reset, this signal is tri-stated.

Pin Name	Type	Description
IDSEL	in	**INITIALIZATION DEVICE SELECT:** IDSEL is used as a chip select during configuration read and write transactions. The PCEB samples IDSEL during the address phase of a transaction. If the PCEB samples IDSEL asserted during a configuration read or write, the PCEB responds by asserting DEVSEL# on the next cycle.
DEVSEL#	s/t/s	**DEVICE SELECT:** The PCEB asserts DEVSEL# to claim a PCI transaction as a result of positive or subtractive decode. As an output, the PCEB asserts DEVSEL# when it samples IDSEL asserted during configuration cycles to PCEB configuration registers. As an input, DEVSEL# indicates the response to a PCEB-initiated transaction. The PCEB, when not a master, samples this signal for all PCI transactions to decide whether to subtractively decode the cycle (except for configuration and special cycles). During reset, this signal is tri-stated.
PAR	t/s	**PARITY:** PAR is even parity across AD[31:0] and C/BE[3:0]#. When acting as a master, the PCEB drives PAR during the address and write data phases. As a target, the PCEB drives PAR during read data phases. When PCIRST# is asserted, the PCEB drives the PAR signal to keep it from floating. The PCEB acts as the central resource responsible for driving the PAR signal when no other device is granted the PCI Bus and the bus is idle. Note that the driving and tri-stating of the PAR signal is always one clock delayed from the corresponding driving and tri-stating of the AD[31:0] and C/BE[3:0]# signals.
PERR#	s/t/s	**PARITY ERROR:** PERR# reports data parity errors on all transactions, except special cycles. This signal can only be asserted (by the agent receiving data) two clocks following the data (which is one clock following the PAR signal that covered the data). The duration of PERR# is one clock for each data phase that a data parity error is detected. (If multiple data errors occur during a single transaction the PERR# signal is asserted for more than a single clock.) PERR# must be driven high for one clock before being tri-stated. During reset, this signal is tri-stated.

2.2 PCI Arbiter Signals

Pin Name	Type	Description
CPUREQ#	in	**CPU REQUEST:** CPUREQ# asserted indicates that the Host CPU requests use of the PCI Bus. During PCIRST#, this signal must be sampled high by the PCEB. When PCIRST# is asserted (and CPUREQ# is sampled high), the PCEB drives the AD, C/BE#, and PAR signals to keep them from floating.
REQ[3:0]#	in	**REQUEST:** A bus master asserts the corresponding request signal to request the PCI Bus.
CPUGNT#	out	**CPU GRANT:** The PCEB asserts CPUGNT# to indicate that the CPU master (Host Bridge) has been granted the PCI Bus. During PCI reset, CPUGNT# is tri-stated.
GNT[3:0]#	out	**GRANT:** The PCEB asserts one of the GNT[3:0] signals to indicate that the corresponding PCI master has been granted the PCI Bus. During PCI reset, these signals are tri-stated.
MEMREQ#	out	**MEMORY REQUEST:** If the PCEB is configured in Guaranteed Access Time (GAT) Mode, MEMREQ# is asserted when an EISA device or DMA requests the EISA Bus. The PCEB asserts this signal (along with FLSHREQ#) to indicate that the PCEB requires ownership of main memory. The PCEB asserts FLSHREQ# concurrently with asserting MEMREQ#. This signal is synchronous to the PCI clock. During reset, this signal is driven high. **FLSHREQ# MEMREQ# Meaning** 1 1 Idle 0 1 Flush buffers pointing towards PCI to avoid ISA deadlock 1 0 GAT enabled or disabled: For buffer coherency in APIC systems, the buffers pointing to main memory must be flushed and disabled for the duration of assertion. 0 0 GAT mode: Guarantee PCI Bus immediate access to main memory (this may or may not require the PCI-to-main memory buffers to be flushed first, depending on the number of buffers).
FLSHREQ#	out	**FLUSH REQUEST:** FLSHREQ# is asserted by the PCEB to command all of the system's posted write buffers pointing towards PCI to be flushed. This is required before granting the EISA Bus to an EISA master or the DMA. Note that, for APIC related buffer flush requests, this signal is negated. This signal is synchronous to the PCI clock. During reset, this signal is driven high.

Pin Name	Type	Description
MEMACK#	in	**MEMORY ACKNOWLEDGE:** MEMACK# is the response handshake that indicates to the PCEB that the function requested over the MEMREQ# and/or FLSHREQ# signals has been completed. If the PCEB is configured for Guaranteed Access Time Mode through the Arbiter Control Register, and both MEMREQ# and FLSHREQ# are asserted, the assertion of MEMACK# indicates to the PCEB that ownership of main memory has been granted and that all system buffers have been flushed and temporarily disabled. If MEMACK# is asserted in response to assertion of MEMREQ# (GAT either enabled or disabled), it indicates that the system's buffers pointing towards the main memory are flushed and temporarily disabled so that APIC can proceed with the interrupt message sequence. If FLSHREQ# is asserted and MEMREQ# is not asserted (with GAT mode being either enabled or disabled), the assertion of MEMACK# indicates that the system's posted write buffers pointing towards PCI are flushed and temporarily disabled, and the EISA Bus can be granted to an EISA master or DMA. This signal is synchronous to the PCI clock.

2.3 Address Decoder Signals

Pin Name	Type	Description
MEMCS#	out	**MEMORY CHIP SELECT:** MEMCS# is a programmable address decode signal provided to a Host CPU bridge. A Host bridge can use MEMCS# to forward a PCI cycle to the main memory behind the bridge. MEMCS# is asserted one PCI clock after FRAME# is sampled asserted (address phase) and is valid for one clock cycle before being negated. MEMCS# is driven high during reset.
PIODEC#	in	**PCI I/O SPACE DECODER:** PIODEC# can be used to provide arbitrarily complex EISA-to-PCI I/O address space mapping. This signal can be connected to the decode select output of an external I/O address decoder. When PIODEC# is asserted during an EISA I/O cycle, that cycle is forwarded to the PCI Bus. Note that an external pull-up resistor is required if this input signal is not used (i.e., not driven by the external logic).

2.4 EISA Interface Signals

Pin Name	Type	Description
BCLK	in	**BUS CLOCK:** BCLK is the system clock used to synchronize events on the EISA Bus. The ESC device generates BCLK (BCLKOUT), which is a divided down clock from a PCICLK. BCLK runs at a frequency that is dependent on PCICLK and a selected division factor (within the ESC). For example, a 25 MHz PCICLK and a division factor of 3 results in an 8.33 MHz BCLK.
START#	t/s	**START:** START# provides timing control at the start of the cycle and remains asserted for one BCLK period. When the PCEB is an EISA master, START# is an output signal. START# is asserted after LA[31:24]#, LA[23:2] and M/IO# become valid. START# is negated on the rising edge of the BCLK, one BCLK after it was asserted. The trailing edge of START# is always delayed from the rising edge of BCLK. When the PCEB is an EISA master, for cycles to a mismatched slave (see note at the end of this section), START# becomes an input signal at the end of the first START# phase and remains an input until the negation of the last CMD#. The ESC gains the control of the transfer and generates START#. When the PCEB is an EISA slave, START# is an input signal. It is sampled on the rising edge of BCLK. Upon PCIRST#, this signal is tri-stated and placed in output mode.
CMD#	in	**COMMAND:** CMD# provides timing control within the cycle. In all cases, CMD# is an input to the PCEB from the ESC. CMD# is asserted from the rising edge of BCLK, simultaneously with the negation of START#, and remains asserted until the end of the cycle.
M/IO#	t/s	**MEMORY OR I/O:** M/IO# identifies the current cycle as a memory or an I/O cycle. M/IO# is pipelined from one cycle to the next and must be latched by the slave. M/IO# = 1 indicates a memory cycle and M/IO# = 0 indicates an I/O cycle. When the PCEB is an EISA master, the M/IO# is an output signal. When the PCEB is an EISA slave, M/IO# is an input signal. The PCEB responds as an EISA slave for both memory and I/O cycles. Upon PCIRST#, this signal is tri-stated and is placed in output mode.
W/R#	t/s	**WRITE OR READ:** W/R# identifies the cycle as a write or a read cycle. The W/R# signal is pipelined from one cycle to the next and must be latched by the slave. W/R# = 1 indicates a write cycle and W/R# = 0 indicates a read cycle. When the PCEB is an EISA master, W/R# is an output signal. When the PCEB is an EISA slave, W/R# is an input signal. Upon PCIRST#, this signal is tri-stated and placed in output mode.

Pin Name	Type	Description
EXRDY	od	**EISA READY:** EXRDY is used by EISA I/O and memory slaves to request wait states during a cycle. Each wait state is a BCLK period. The PCEB, as an EISA master or slave, samples EXRDY. As an input, the EXRDY is sampled on the falling edge of BCLK after the CMD# has been asserted, and if inactive, each falling edge thereafter. When PCEB is an EISA slave, it may drive EXRDY low to introduce wait states. During reset, this signal is not driven.
EX32#	od	**EISA 32 BIT:** EX32# is used by the EISA slaves to indicate support of 32 bit transfers. When the PCEB is an EISA master, it samples EX32# on the same rising edge of BCLK that START# is negated. During mismatched cycles (see note at the end of this section), EX32# (and EX16#) is used to transfer the control back to the PCEB. EX32# (along with EX16#) is asserted by the ESC on the falling edge of BCLK before the rising edge of the BCLK when the last CMD# is negated. This indicates that the cycle control is transferred back to the PCEB. As an EISA slave, the PCEB always drives EX32# to indicate 32 bit support for EISA cycles. During reset, this signal is not driven.
EX16#	in	**EISA 16 BIT:** EX16# is used by the EISA slaves to indicate their support of 16 bit transfers. As an EISA master, the PCEB samples EX16# on the same rising edge of BCLK that START# is negated. During mismatched cycles (see note at the end of this section), EX16# (and EX32#) is used to transfer the control back to the PCEB. EX16# (along with EX32#) is asserted by the ESC on the falling edge of the BCLK before the rising edge of the BCLK when the last CMD# is negated. This indicates that the cycle control is transferred back to the PCEB. As an EISA slave, the PCEB never asserts EX16#.
MSBURST#	t/s	**MASTER BURST:** MSBURST# is an output when the PCEB is an EISA master and an input when the PCEB is a slave. As a master, the PCEB asserts MSBURST# to indicate to the slave that the next cycle is a burst cycle. If the PCEB samples SLBURST# asserted on the rising edge of BCLK after START# is asserted, the PCEB asserts MSBURST# on the next BCLK edge and proceeds with the burst cycle. As a slave, the PCEB monitors this signal in response to the PCEB asserting SLBURST#. The EISA master asserts MSBURST# to the PCEB to indicate that the next cycle is a burst cycle. As a slave, the PCEB samples MSBURST# on the rising edge of BCLK after the rising edge of BCLK that CMD# is asserted by the ESC. MSBURST# is sampled on all subsequent rising edges of BCLK until the signal is sampled negated. The burst cycle is terminated on the rising edge of BCLK when MSBURST# is sampled negated, unless EXRDY is sampled negated on the previous falling edge of BCLK. During reset, this signal is tri-stated.

Pin Name	Type	Description
SLBURST#	t/s	**SLAVE BURST:** SLBURST# is an input when the PCEB is an EISA master and an output when the PCEB is a slave. When the PCEB is a master, the slave indicates that it supports burst cycles by asserting SLBURST# to the PCEB. The PCEB samples SLBURST# on the rising edge of BCLK at the end of START# for EISA master cycles. When the PCEB is an EISA slave, this signal is an output. As a slave, the PCEB asserts this signal to the master indicating that the PCEB supports EISA burst cycles. During reset, this signal is tri-stated.
LOCK#	t/s	**LOCK:** When asserted, LOCK# guarantees exclusive memory access. This signal is asserted by the PCEB when the PCI master is running locked cycles to EISA slaves. When asserted, this signal locks the EISA subsystem. LOCK# can also be activated by a device on the EISA Bus. This condition is propagated to the PCI Bus via the PLOCK# signal. During reset, this signal is tri-stated.
BE[3:0]#	t/s	**BYTE ENABLES:** BE[3:0]# identify the specific bytes that are valid during the current EISA Bus cycles. When the PCEB is an EISA master and the cycles are directed to a matched slave (slave supports 32-bit transfers), the BE[3:0]# are outputs from the PCEB. When the cycles are directed to a mis-matched slave (slave does not support 32-bit transfers - see note), the BE[3:0]# are floated one and half BCLKs after START# is asserted. These signals become inputs (driven by the ESC) for the rest of the cycle. BE[3:0]# are pipelined signals and must be latched by the addressed slave. When the PCEB is an EISA/ISA/DMA slave, BE[3:0]# are inputs to the PCEB. Upon PCIRST#, these signals are tri-stated and placed in output mode.
LA[31:24]#, LA[23:2]	t/s	**LATCHABLE ADDRESS:** LA[31:24]# and LA[23:2] are the EISA address signals. When the PCEB is an EISA master, these signals are outputs from the PCEB. These addresses are pipelined and must be latched by the EISA slave. LA[31:24]# and LA[23:2] are valid on the falling edge of START#. Note that the upper address bits are inverted before being driven on LA[31:24]#. The timing for LA[31:24] and LA[23:2] are the same. When the PCEB is an EISA slave, these signals are inputs and are latched by the PCEB. For I/O cycles, the PCEB, as an EISA master, floats LA[31:24]# to allow for ESC's address multiplexing (during I/O cycle to configuration RAM). LA[23:2] are actively driven by the PCEB. For memory cycles, the PCEB as an EISA master, drives the LA address lines. During reset, these signals are tri-stated.

Pin Name	Type	Description
SD[31:0]	t/s	**SYSTEM DATA:** SD[31:0] are bi-directional data lines that transfer data between the PCEB and other EISA devices. Data transfer between EISA and PCI devices use these signals. The data swapping logic in the PCEB ensures that the data is available on the correct byte lanes for any given transfer. During reset, these signals are tri-stated.
REFRESH#	in	**REFRESH:** When asserted, REFRESH# indicates to the PCEB that the current cycle on the EISA Bus is a refresh cycle. It is used by the PCEB decoder to distinguish between EISA memory read cycles and refresh cycles.

NOTE:
Mis-matched Cycles. When the PCEB is an EISA master, cycles to the slaves, other than 32 bits transfers, are considered a mis-matched cycle. For mis-matched cycles, the PCEB backs off the EISA Bus one and half BCLKs after it asserted START# by releasing (floating) START#, BE[3:0]# and the SD[31:0] lines. The ESC device then takes control of the transfer. The ESC controls the transfer until the last transfer. At the end of the last transfer, the control is transferred back to the PCEB. The ESC transfers control back to the PCEB by asserting EX32# and EX16# on the falling edge of BCLK before the rising edge of BCLK when the last CMD# is negated.

2.5 ISA Interface Signals

An ISA interface signal is included to improve the PCEB's handling of I/O cycles on the EISA side of the bridge. This signal permits ISA masters to address PCI I/O slaves using the full 16-bit bus size. The signal also allows the PCEB to identify 8-bit I/O slaves for purposes of generating the correct amount of I/O recovery.

Pin Name	Type	Description
IO16#	o/d	**16-BIT I/O CHIP SELECT:** As an EISA slave, the PCEB asserts IO16# when PIODEC# is asserted or an I/O cycle to PCI is detected. As an EISA master, the PCEB uses IO16# as an input to determine the correct amount of I/O recovery time from the I/O Recovery Time (IORT) Register. This register contains bit-fields that are used to program recovery times for 8-bit and 16-bit I/O. When IO16# is asserted, the recovery time programmed into the 16-bit I/O field (bits [1:0]) , if enabled, is used. When IO16# is negated, the recovery time programmed into the 8-bit I/O field (bits [5:3]), if enabled, is used. This signal must have an external pull-up resistor. During reset, this signal is not driven.

2.6 PCEB/ESC Interface Signals

Pin Name	Type	Description
ARBITRATION AND INTERRUPT ACKNOWLEDGE CONTROL		
EISAHOLD	in	**EISA HOLD:** EISAHOLD is used by the ESC to request control of the EISA Bus from the PCEB. This signal is synchronous to PCICLK and is driven inactive when PCIRST# is asserted.
EISAHLDA	out	**EISA HOLD ACKNOWLEDGE:** The PCEB asserts EISAHLDA to inform the ESC that it has been granted ownership of the EISA Bus. This signal is synchronous to the PCICLK.
PEREQ#/ INTA#	out	**PCI-TO-EISA REQUEST OR INTERRUPT ACKNOWLEDGE:** PEREQ#/INTA# is a dual-function signal. The signal function is determined by the state of EISAHLDA signal. When EISAHLDA is negated, this signal is an interrupt acknowledge (i.e., PEREQ#/ INTA# asserted indicates to the ESC that the current cycle on the EISA is an interrupt acknowledge). When EISAHLDA is asserted, this signal is a PCI-to-EISA request (i.e. PEREQ#/ INTA# asserted indicates to the ESC that the PCEB needs to obtain the ownership of the EISA Bus on behalf of a PCI agent). This signal is synchronous to the PCICLK and it is driven inactive when PCIRST# is asserted.
STPGNT#	out	**STOP GRANT ACKNOWLEDGE:** STPGNT# is asserted when the PCEB receives a STOP GRANT PCI special cycle for one PCICLK period. This signal is only asserted when the PCI AD[31:0] signals equal 00120002h during the first data phase of the PCI special cycle. Data of 00120002h on AD[31:0] in subsequent data phases during a PCI special cycle does not result in the assertion of STPGNT#.

PIN NAME	Type	Description
PCEB BUFFER COHERENCY CONTROL		
NMFLUSH#	t/s	**NEW MASTER FLUSH:** The bi-directional NMFLUSH# signal provides handshake between the PCEB and ESC to control flushing of PCI system buffers on behalf of EISA masters. During an EISA Bus ownership change, before the ESC can grant the bus to the EISA master (or DMA), the ESC must ensure that system buffers are flushed and the buffers pointing towards the EISA subsystem are disabled. The ESC asserts NMFLUSH# for one PCI clock to request system buffer flushing. (After asserting NMFLUSH# for 1 PCI clock, the ESC tri-states NMFLUSH#.) When the PCEB samples NMFLUSH# asserted, it starts immediately to assert NMFLUSH# and begins flushing its internal buffers, if necessary. The PCEB also requests PCI system buffer flushing via the MEMREQ#, FLSHREQ#, and MEMACK# signals. When the PCEB completes its internal buffer flushing and MEMACK# is asserted (indicating that the PCI system buffer flushing is complete), the PCEB negates NMFLUSH# for 1 PCI clock and stops driving it. When the ESC samples NMFLUSH# negated, it grants the EISA Bus to an EISA master (or DMA). The ESC resumes responsibility of the default NMFLUSH# driver and starts driving NMFLUSH# negated until the next time a new EISA master (or DMA) wins arbitration. This signal is synchronous with PCICLK and is negated by the ESC at reset.
AFLUSH#	t/s	**APIC FLUSH:** AFLUSH# is bi-directional signal between the PCEB and ESC that controls system buffer flushing on behalf of the APIC. After a reset the ESC negates AFLUSH# until the APIC is initialized and the first interrupt request is recognized.

Pin Name	Type	Description
DATA SWAP BUFFER CONTROL		
SDCPYEN01 # SDCPYEN02 # SDCPYEN03 # SDCPYEN13 #	in	**COPY ENABLE:** These active Low signals perform byte copy operation on the EISA data bus (SD[31:0]). The Copy Enable signals are asserted during mis-matched cycles and are used by the PCEB to enable byte copy operations between the SD data byte lanes 0, 1, 2, and 3 as follows: SDCPYEN01 #: Copy between Byte Lane 0 (SD[7:0]) and Byte Lane 1 (SD[15:8]) SDCPYEN02 #: Copy between Byte Lane 0 (SD[7:0]) and Byte Lane 2 (SD[23:16]) SDCPYEN03 #: Copy between Byte Lane 0 (SD[7:0]) and Byte Lane 3 (SD[31:24]) SDCPYEN13 #: Copy between Byte Lane 1 (SD[15:8]) and Byte Lane 3 (SD[31:24]) Note that the direction of the copy is controlled by SDCPYUP.
SDCPYUP	in	**SYSTEM DATA COPY UP:** SDCPYUP controls the direction of the byte copy operation. A high on SDCPYUP indicates a COPY UP operation where the lower byte(s) of the SD data bus are copied onto the higher byte(s) of the bus. A low on the signal indicates a COPY DOWN operation where the higher byte(s) of the data bus are copied on to the lower byte(s) of the bus. The PCEB uses this signal to perform the actual data byte copy operation during mis-matched cycles.
SDOE[2:0] #	in	**SYSTEM DATA OUTPUT ENABLE:** These active Low signals enable the SD data output onto the EISA Bus. The ESC only activates these signals during mis-matched cycles. The PCEB uses these signal to enable the SD data buffers as follows: SDOE0 # Enables byte lane 0 SD[7:0] SDOE1 # Enables byte lane 1 SD[15:8] SDOE2 # Enables byte lane 3 SD[31:24] and byte lane 2 SD[23:16]
SDLE[3:0] #	in	**SYSTEM DATA LATCH ENABLE:** SDLE[3:0] # enable the latching of data on the EISA Bus. These signals are activated only during mis-matched cycles, except PCEB-initiated write cycles. The PCEB uses these signals to latch the SD data bus as follows: SDLE0 # Latch byte lane 0 SD[7:0] SDLE1 # Latch byte lane 1 SD[15:8] SDLE2 # Latch byte lane 2 SD[23:16] SDLE3 # Latch byte lane 3 SD[31:24]

2.7 Test Signal

Pin Name	Type	Description
TEST #	in	**TEST:** This pin is used to tri-state all PCEB outputs. During normal operations, this pin must be tied high.

3.0 REGISTER DESCRIPTION

The PCEB contains both PCI configuration registers and I/O registers. The configuration registers (Table 1) are located in PCI configuration space and are only accessible from the PCI Bus. The addresses shown in the table for each register are offset values that appear on AD[7:2] and C/BE[3:0]#. The configuration registers can be accessed as Byte, Word (16-bit), or Dword (32-bit) quantities. All multi-byte numeric fields use ''little-endian'' ordering (i.e., lower addresses contain the least significant parts of the fields).

The BIOS Timer is the only non-configuration register (Section 3.2, I/O Registers). This register, like the configuration registers, is only accessible from the PCI Bus. The BIOS Timer Register can be accessed as byte, word, or Dword quantities.

Some of the PCEB registers contain reserved bits. These bits are labeled ''Reserved''. Software must take care to deal correctly with bit-encoded fields that are reserved. On reads, software must use appropriate masks to extract the defined bits and not rely on reserved bits being any particular value. On writes, software must ensure that the values of reserved bits are preserved. That is, the values of reserved bit positions must first be read, merged with the new values for other bit positions and the data then written back.

In addition to reserved bits within a register, the PCEB contains address locations in the PCI configuration space that are marked ''Reserved'' (Table 1). The PCEB responds to accesses to these address locations by completing the PCI cycle. When a reserved register location is read, 0000h is returned. Writes have no affect on the PCEB.

During a hard reset (PCIRST# asserted), the PCEB registers are set to pre-determined **default** states. The default values are indicated in the individual register descriptions.

During the address phase of a configuration cycle, Bits [10:8] encode one of eight possible functions on a device. The PCEB only supports one function; that of a bridge between the PCI and EISA/ISA Busses. This function has the code of 000. Thus, for accessing PCEB configuration registers, Bits[10:8] = 000 of the address. If the PCEB IDSEL is asserted and any of the above three bits is 1, the PCEB returns all zeros for a read and does not respond to a write.

3.1 Configuration Registers

Table 1 summarizes the PCEB configuration space registers. Following the table, is a detailed description of each register and register bit. The register descriptions are arranged in the order that they appear in Table 1. The following nomenclature is used for access attributes.

RO **Read Only**. If a register is read only, writes to this register have no effect.

R/W **Read/Write**. A register with this attribute can be read and written.

R/WC **Read/Write Clear**. A register bit with this attribute can be read and written. However, a write of a 1 clears (sets to 0) the corresponding bit and a write of a 0 has no effect.

NOTE:

Some register fields are used to program address ranges for various PCEB functions. The register contents represent the address bit value and not the signal level on the bus. For example, the upper address lines on the EISA Bus have inverted signals (LA[31:24]#). However, this inversion is automatically handled by the PCEB hardware and is transparent to the programmer.

Table 1. Configuration Registers

Address Offset	Abbreviation	Register Name	Access
00–01h	VID	Vendor Identification	RO
02–03h	DID	Device Identification	RO
04–05h	PCICMD	Command Register	R/W
06–07h	PCISTS	Status Register	RO, R/WC
08h	RID	Revision Identification	RO
09–0Ch	—	Reserved	—
0Dh	MLTIM	Master Latency Timer	R/W
0E–3Fh	—	Reserved	—
40h	PCICON	PCI Control	R/W
41h	ARBCON	PCI Arbiter Control	R/W
42h	ARBPRI	PCI Arbiter Priority Control	R/W
43h	ARBPRIX	PCI Arbiter Priority Control Extension	R/W
44h	MCSCON	MEMCS# Control	R/W
45h	MCSBOH	MEMCS# Bottom of Hole	R/W
46h	MCSTOH	MEMCS# Top of Hole	R/W
47h	MCSTOM	MEMCS# Top of Memory	R/W
48–49h	EADC1	EISA Address Decode Control 1	R/W
4A–4Bh	—	Reserved	—
4Ch	IORTC	ISA I/O Recovery Time Control	R/W
4Dh–53h	—	Reserved	—
54h	MAR1	MEMCS# Attribute Register #1	R/W
55h	MAR2	MEMCS# Attribute Register #2	R/W
56h	MAR3	MEMCS# Attribute Register #3	R/W
57h	—	Reserved	—
58h	PDCON	PCI Decode Control	R/W
59h	—	Reserved	—
5Ah	EADC2	EISA Address Decode Control 2	R/W
5Bh	—	Reserved	—

Table 1. Configuration Registers (Continued)

Address Offset	Abbreviation	Register Name	Access
5Ch	EPMRA	EISA-to-PCI Memory Region Attributes	R/W
5D–5Fh	—	Reserved	—
60–6Fh	MEMREGN[4:1]	EISA-to-PCI Memory Region Address (4 Registers)	R/W
70–7Fh	IOREGN[4:1]	EISA-to-PCI I/O Region Address (4 Registers)	R/W
80–81h	BTMR	BIOS Timer Base Address	R/W
84h	ELTCR	EISA Latency Timer Control Register	R/W
85–87h	—	Reserved	—
88–8Bh	PTCR	PCEB Test Control Register—**DO NOT WRITE**	—
8C–FFh	—	Reserved	—

3.1.1 VID—VENDOR IDENTIFICATION REGISTER

Address Offset: 00–01h
Default Value: 8086h
Attribute: Read Only
Size: 16 bits

The VID Register contains the vendor identification number. This register, along with the Device Identification Register, uniquely identify any PCI device. Writes to this register have no effect.

Bit	Description
15:0	**Vendor Identification Number:** This is a 16-bit value assigned to Intel.

3.1.2 DID—DEVICE IDENTIFICATION REGISTER

Address Offset: 02–03h
Default Value: 0482h
Attribute: Read Only
Size: 16 bits

The DID Register contains the device identification number. This register, along with the VID Register, define the PCEB. Writes to this register have no effect.

Bit	Description
15:0	**Device Identification Number:** This is a 16-bit value assigned to the PCEB.

3.1.3 PCICMD—PCI COMMAND REGISTER

Address Offset: 04–05h
Default Value: 0007h
Attribute: Read/Write, Read Only
Size: 16 bits

This 16-bit register contains PCI interface control information. This register enables/disables PCI parity error checking, enables/disables PCEB bus master capability, and enables/disables the PCEB to respond to PCI-originated memory and I/O cycles. Note that, for certain PCI functions that are not implemented within the PCEB, the control bits are still shown (labeled ''not supported'').

Bit	Description
15:9	**Reserved**
8	**SERR# Enable (SERRE)—Not Supported—RO:** Function of this bit is to control the SERR# signal. Since the PCEB does not implement the SERR# signal, this bit always reads as 0 (disabled).
7	**Wait State Control (WSC)—Not Supported—RO:** This bit controls insertion of wait-states for devices that do not meet the 33-10 PCI specification. Since PCEB meets the 33-10 specification, this control function is not implemented. WSC is always read as 0.
6	**Parity Error Enable (PERRE)—R/W:** PERRE controls the PCEB's response to PCI parity errors. When PERRE = 1, the PCEB asserts the PERR# signal when a parity error is detected. When PERRE = 0, the PCEB ignores any parity errors that it detects. After PCIRST#, PERRE = 0 (parity checking disabled).
5	**VGA Palette Snoop (VGPS)—Not Supported—RO:** This bit is intended only for specific control of PCI-based VGA devices and it is not applicable to the PCEB. This bit is not implemented and always reads as 0.
4	**Memory Write and Invalidate Enable (MWIE)—Not Supported—RO:** This is an enable bit for using the Memory Write and Invalidate command. The PCEB doesn't support this command as a master. As a slave the PCEB aliases this command to a memory write. This bit always reads as 0 (disabled).
3	**Special Cycle Enable (SCE)—Not Supported—RO:** Since this capability is not implemented, the PCEB does not respond to any type of special cycle. This bit always reads as 0.
2	**Bus Master Enable (BME)—R/W:** ME enables/disables the PCEB's PCI Bus master capability. When BME = 0, the PCEB bus master capability is disabled. This prevents the PCEB from requesting the PCI Bus on behalf of EISA/ISA masters, the DMA, or the Line Buffers. When BME = 1, the bus master capability is enabled. This bit is set to 1 after PCIRST#.
1	**Memory Space Enable (MSE)—R/W:** This bit enables the PCEB to accept PCI-originated memory cycles. When MSE = 1, the PCEB responds to PCI-originated memory cycles to the EISA Bus. When MSE = 0, the PCEB does not respond to PCI-originated memory cycles to the EISA Bus (DEVSEL# is inhibited). This bit is set to 1 (enabled for BIOS access) after PCIRST#.
0	**I/O Space Enable (IOSE)—R/W:** This bit enables the PCEB to accept PCI-originated I/O cycles. When IOSE = 1, the PCEB responds to PCI-originated I/O cycles. When IOSE = 0, the PCEB does not respond to a PCI I/O cycle (DEVSEL# is inhibited), including I/O cycles bound for the EISA Bus. This bit is set to 1 (I/O space enabled) after PCIRST#.

3.1.4 PCISTS—PCI STATUS REGISTER

Address Offset: 06–07h
Default Value: 0200h
Attribute: Read Only, Read/Write Clear
Size: 16 bits

This 16-bit register provides status information for PCI Bus-related events. Some bits are read/write clear. These bits are set to 0 whenever the register is written, and the data in the corresponding bit location is 1 (R/WC). For example, to clear bit 12 and not affect any other bits, write the value 0001__0000__0000__0000b to this register. Note that for certain PCI functions that are not implemented in the PCEB, the control bits are still shown (labeled ''not supported'').

Bit	Description
15	**Parity Error Status (PERRS)—R/WC:** This bit is set to 1 whenever the PCEB detects a parity error, even if parity error handling is disabled (as controlled by bit 6 in the PCI Command Register). Software sets PERRS to 0 by writing a 1 to this bit location.
14	**SERR# Status (SERRS)—Not Supported:** This bit is used to indicate that a PCI device asserted the SERR# signal. The PCEB does not implement this signal. SERRS is always read as 0.
13	**Master Abort Status (MA)—R/WC:** When the PCEB, as a master, generates a master abort, this bit is set to 1. Software sets MA to 0 by writing a 1 to this bit location.
12	**Received Target Abort Status (RTAS)—R/WC:** When the PCEB, as a master, receives a target abort condition, this bit is set to 1. Software sets RTAS to 0 by writing a 1 to this bit location.
11	**Signaled Target Abort Status (STAS)—Not Supported:** This bit is set to 1 by a PCI target device when they generate a Target Abort. Since the PCEB never generates a target abort, this bit is not implemented and will always be read as a 0.
10:9	**DEVSEL Timing Status (DEVT)—RO:** This read only field indicates the timing of the DEVSEL# signal when PCEB responds as a target. The PCI Specification defines three allowable timings for assertion of DEVSEL#: 00b = fast, 01b = medium, and 10b = slow (11b is reserved). DEVT indicates the slowest time that a device asserts DEVSEL# for any bus command, except configuration read and configuration write cycles. The PCEB implements medium speed DEVSEL# timing and, therefore, DEVT[10:9] = 01 when read.
8:0	**Reserved**

3.1.5 RID—REVISION IDENTIFICATION REGISTER

Address Offset: 08h
Default Value: 03h (82375EB, A-2 stepping)
04h (82375SB, B-0 stepping)
Attribute: Read Only
Size: 8 bits

This 8-bit register contains the device revision number of the PCEB. Writes to this register have no effect.

Bit	Description
7:0	**Revision Identification Number:** This 8-bit value is the revision number of the PCEB.

3.1.6 MLT—MASTER LATENCY TIMER REGISTER

Address Offset: 0Dh
Default Value: 00h
Attribute: Read/Write
Size: 8 bits

This 8-bit register contains the programmable value of the Master Latency Timer for use when the PCEB is a master on the PCI Bus. The granularity of the timer is 8 PCI clocks. Thus, bits[2:0] are not used and always read as 0s.

Bit	Description
7:3	**Count Value:** This 5-bit field contains the count value of the Master Latency Timer, with a granularity of 8 PCI clocks. For example, value 00101b provides a time-out period of 5x8 = 40 PCI clocks. Maximum count value is 11111b, which corresponds to 248 PCI clocks.
2:0	**Reserved**

3.1.7 PCICON—PCI CONTROL REGISTER

Address Offset: 40h
Default Value: 20h
Attribute: Read/Write
Size: 8 bits

This 8-bit register enables/disables the PCEB's data buffers, defines the subtractive decoding sample point, and enables/disables response to the PCI interrupt acknowledge cycle.

NOTE:
The Line Buffers are typically enabled or disabled during system initialization. These buffers should not be dynamically enabled/disabled during runtime. Otherwise, data coherency can be affected, if a buffer containing valid write data is disabled and then, later, re-enabled.

Bit	Description
7	**Reserved**
6	**EISA-To-PCI Line Buffer Enable (ELBE):** When ELBE = 0, the EISA-to-PCI Line Buffers are disabled and when ELBE = 1, the EISA-to-PCI Line Buffers are enabled. After PCIRST#, the Line Buffers are disabled (ELBE = 0). Note that when ELBE is set to 1, the line buffers are utilized for transfers to or from the regions defined by the REG[4:1] bits in the EPMRA register (offset 5Ch).
5	**Interrupt Acknowledge Enable (IAE):** When IAE = 0, the PCEB decodes PCI interrupt acknowledge cycles in a semi-subtractive manner. When there is data posted in the Line Buffers, the PCEB intervenes in the PCI interrupt acknowledge cycle by generating a retry. The PCEB also initiates a buffer flush operation and will keep generating retries until the buffers are flushed. The PCEB then subtractively decodes the PCI interrupt acknowledge cycle in order to allow an external PCI-based interrupt controller to respond with the vector. If no external PCI-based interrupt controller has responded to the PCI Interrupt Acknowledge cycle at the DEVSEL# sampling point, the cycle is handled by the PCEB in a subtractive decode manner. When IAE = 1, the PCEB positively decodes the interrupt acknowledge cycles and responds to the cycles in the normal fashion (i.e., uses the PEREQ#/INTA# signal to fetch the vector from the ESC, after the internal buffers are flushed).

Bit	Description
4:3	**Subtractive Decoding Sample Point (SDSP):** The SDSP field determines the DEVSEL# sample point, after which an inactive DEVSEL# results in the PCEB forwarding the unclaimed PCI cycle to the EISA Bus (subtractive decoding). This setting should match the slowest device in the system. When the MEMCS# function is enabled, MEMCS# is sampled as well as an early indication of an eventual DEVSEL#. **Bits[4:3]** — **Operation** 00 — Slow sample point (default value) 01 — Typical sample point 10 — Reserved 11 — Reserved
2	**Reserved.** This bit must be 0 when programming this register.
1:0	**Reserved**

3.1.8 ARBCON—PCI ARBITER CONTROL REGISTER

Address Offset: 41h
Default Value: 80h
Attribute: Read/Write
Size: 8 bits

This register controls the operation of the PCEB's internal PCI arbiter. The register enables/disables auto-PEREQ#, controls the master retry timer, enables/disables CPU bus parking, controls bus lock, and enables/disables the guaranteed access time (GAT) mode for EISA/ISA accesses.

NOTE:

1. For proper system operation, the master retry timer (bits[4:3]) must not be disabled. This field defaults to 00 (disabled) and must be program to either 01, 10, or 11.
2. The PCMC Host bridge device requires that bit 7 be set to 1 (default). However, other chip sets might need to have this function disabled to provide more optimum performance for EISA subsystems. This functionality is built-in to prevent starvation of PCI agents (in particular, the host bridge, i.e., CPU) when EISA masters are performing transactions in the GAT mode. If this function is disabled, the host bridge must be capable of generating the PCI Bus request, even when the Host Bus is not controlled by the CPU (CPU tri-stated all Host Bus signals, or even only address bus, in response to HOLD/AHOLD). The CPU pin that provides an indication of a request for the external bus (e.g. after cache miss) can be used by the host bridge to generate the request for the PCI Bus during GAT mode operations, even when no address lines are driven by the CPU.

Bit	Description
7	**Auto-PEREQ# Control (APC):** APC Enables/Disables control of the auto-PEREQ# function when GAT mode is enabled via bit 0 (GAT = 1). When APC = 1 (and GAT = 1), the PEREQ# signal is asserted whenever the EISAHLDA signal is asserted. When APC = 0, the PEREQ# signal is not automatically asserted but it will be activated upon PCI Bus request from any PCI agent. After PCIRST#, APC = 1 (enabled). See note.
6:5	**Reserved**
4:3	**Master Retry Timer (MRT):** This 2-bit field determines the number of PCICLKs after the first retry that a PCI initiator's bus request will be masked. Note that for proper system operation, this register must be programmed with either 01, 10, 11. **Bits[4:3]** — **Operation** 00 — Timer disabled, Retries never masked. (Default) 01 — Retries unmasked after 16 PCICLK's. 10 — Retries unmasked after 32 PCICLK's. 11 — Retries unmasked after 64 PCICLK's.
2	**Bus Park (BP):** When BP = 1, the PCEB will park CPUREQ# on the PCI Bus when it detects the PCI Bus idle. If BP = 0, the PCEB takes responsibility for driving AD, C/BE# and PAR signals upon detection of bus idle state. After PCIRST#, BP = 0 (disabled).
1	**Bus Lock (BL):** When BL = 1, Bus Lock is enabled. The arbiter considers the entire PCI Bus locked upon initiation of any LOCKed transaction. When BL = 0, Resource Lock is enabled. A LOCKed agent is considered a LOCKed resource and other agents may continue normal PCI transactions. After PCIRST#, BL = 0 (disabled).
0	**Guaranteed Access Time (GAT):** When GAT = 1, the PCEB is configured for Guaranteed Access Time mode. This mode guarantees the 2.1 μs CHRDY time-out specification for the EISA/ISA Bus. When the PCEB is a PCI initiator on behalf of an EISA/ISA master, the PCI and main memory bus (host) are arbitrated for in serial and must be owned before the EISA/ISA master is given ownership of the EISA Bus. If the PCEB is not programmed for Guaranteed Access Time (GAT = 0), the EISA/ISA master is first granted the EISA Bus, before the PCI Bus is arbitrated. After a PCIRST#, GAT = 0 (disabled).

3.1.9 ARBPRI—PCI ARBITER PRIORITY CONTROL REGISTER

Address Offset: 42h
Default Value: 04h
Attribute: Read/Write
Size: 8 bits

This register controls the operating modes of the PCEB's internal PCI arbiter. The arbiter consists of four arbitration banks that support up to six masters and three arbitration priority modesfixed priority, rotating priority and mixed priority modes. See Section 5.4, PCI Bus Arbitration for details on programming and using different arbitration modes.

Bit	Description
7	**Bank 3 Rotate Control:** 1 = Enable; 0 = Disable
6	**Bank 2 Rotate Control:** 1 = Enable; 0 = Disable
5	**Bank 1 Rotate Control:** 1 = Enable; 0 = Disable
4 3:2	**Bank 0 Rotate Control:** 1 = Enable; 0 = Disable **Bank 2 Fixed Priority Mode Select—b,a:** ba 00 = Bank0 > Bank3 > Bank1 10 = Bank3 > Bank1 > Bank0 01 = Bank1 > Bank0 > Bank3 11 = Reserved
1	**Bank 1 Fixed Priority Mode Select:** 1 = REQ3# > CPUREQ#; 0 = CPUREQ# > REQ3
0	**Bank 0 Fixed Priority Mode Select:** 1 = REQ0# > PCEBREQ#; 0 = PCEBREQ# > REQ0#. Note that PCEBREQ# is a PCEB internal signal.

3.1.10 ARBPRIX—PCI ARBITER PRIORITY CONTROL EXTENSION REGISTER

Address Offset: 43h
Default Value: 00h
Attribute: Read/Write
Size: 8 bits

This register controls the fixed priority mode for bank 3 of the PCEB's internal arbiter. The ARBPRIX Register is used in conjunction with the PCI Arbiter Priority Control (ARBPRI) Register.

Bit	Description
7:1	**Reserved**
0	**Bank 3 Fixed Priority Mode Select:** 1 = REQ2# > REQ1#; 0 = REQ1# > REQ2#.

3.1.11 MCSCON—MEMCS# CONTROL REGISTER

Address Offset: 44h
Default value: 00h
Attribute: Read/Write
Size: 8 bits

The MCSCON Register provides the master enable for generating MEMCS#. This register also provides read enable (RE) and write enable (WE) attributes for two main memory regions (the 512 KByte - 640 KByte region and an upper BIOS region). PCI accesses within the enabled regions result in the generation of MEMCS#. Note that the 0-512 KByte region does not have RE and WE attribute bits. The 0-512 KByte region can only be disabled with the MEMCS# Master Enable bit (bit 4). Note also, that when the RE and WE bits are both 0 for a particular region, the PCI master can not access the corresponding region in main memory (MEMCS# is not generated for either reads or writes).

Bit	Description
7:5	**Reserved**
4	**MEMCS# Master Enable:** When bit 4 = 1, the PCEB asserts MEMCS# for all accesses to the defined MEMCS# region (as defined by the MCSTOM Register and excluding the memory hole defined by the MCSBOH and MCSTOH Registers), if the accessed location is in a region enabled by bits [3:0] of this register or in the regions defined by the MAR1, MAR2, and MAR3 registers. When bit 4 = 0, the entire MEMCS# function is disabled and MEMCS# is never asserted.
3	**Write Enable For 0F0000–0FFFFFh (Upper 64 KByte BIOS):** When bit 3 = 1, the PCEB generates MEMCS# for PCI master memory write accesses to the address range 0F0000–0FFFFFh. When bit 3 = 0, the PCEB does not generate MEMCS# for PCI master memory write accesses to the address range 0F0000–0FFFFFh.
2	**Read Enable For 0F0000–0FFFFFh (Upper 64 KByte BIOS):** When bit 2 = 1, the PCEB generates MEMCS# for PCI master memory read accesses to the address range 0F0000–0FFFFFh. When bit 2 = 0, the PCEB does not generate MEMCS# for PCI master memory read accesses to the address range 0F0000–0FFFFFh.
1	**Write Enable For 080000–09FFFFh (512–640 KByte):** When bit 1 = 1, the PCEB generates MEMCS# for PCI master memory write accesses to the address range 080000–09FFFFh. When bit 1 = 0, the PCEB does not generate MEMCS# for PCI master memory write accesses to the address range 080000–09FFFFh.
0	**Read Enable For 080000–09FFFFh (512–640 KByte):** When bit 0 = 1, the PCEB generates MEMCS# for PCI master memory read accesses to the address range 080000–09FFFFh. When bit 0 = 0, the PCEB does not generate MEMCS# for PCI master memory read accesses to the address range 080000–09FFFFh.

3.1.12 MCSBOH—MEMCS# BOTTOM OF HOLE REGISTER

Address Offset: 45h
Default value: 10h
Attribute: Read/Write
Size: 8 bits

This register defines the bottom of the MEMCS# hole. MEMCS# is not generated for accesses to addresses within the hole defined by this register and the MCSTOH Register. The hole is defined by the following equation:

TOH ≥ address ≥ BOH. TOH is the top of the MEMCS# hole defined by the MCSTOH Register and BOH is the bottom of the MEMCS# hole defined by this register.

For example, to program the BOH at 1 MByte, the value of 10h should be written to this register. To program the BOH at 2 MByte + 64 KByte this register should be programmed to 21h. To program the BOH at 8 MByte this register should be programmed to 80h.

When the TOH < BOH the hole is disabled. If TOH = BOH, the hole size is 64 KBytes. It is the responsibility of the programmer to guarantee that the BOH is at or above 1 MB. AD[31:24] must be 0's for the hole, meaning the hole is restricted to be under the 16 MByte boundary. The default value for the BOH and TOH disables the hole.

Bit	Description
7:0	**Bottom of MEMCS# Hole:** Bits[7:0] correspond to address lines AD[23:16], respectively.

3.1.13 MCSTOH—MEMCS# TOP OF HOLE REGISTER

Address Offset: 46h
Default value: 0Fh
Attribute: Read/Write
Size: 8 bits

This register defines the top of the MEMCS# hole. MEMCS# is not generated for accesses to addresses within the hole defined by this register and the MCSBOH Register. The hole is defined by the following equation:

TOH ≥ address ≥ BOH. TOH is the top of the MEMCS# hole defined by this register and BOH is the bottom of the MEMCS# hole defined by the MCSBOH Register.

For example, to program the TOH at 1 MByte + 64 KByte, this register should be programmed to 10h. To program the TOH at 2 MByte + 128 KByte this register should be programmed to 21h. To program the TOH at 12 MByte this register should be programmed to BFh.

When the TOH < BOH the hole is disabled. If TOH = BOH, the hole size is 64 KBytes. It is the responsibility of the programmer to guarantee that the TOH is above 1 MByte. AD[31:24] must be 0's for the hole, meaning the hole is restricted to be under the 16 MByte boundary. The default value for the BOH and TOH disables the hole.

Bit	Description
7:0	**Top of MEMCS# Hole:** Bits[7:0] correspond to address lines AD[23:16], respectively.

3.1.14 MCSTOM—MEMCS# TOP OF MEMORY REGISTER

Address Offset: 47h
Default value: 00h
Attribute: Read/Write
Size: 8 bits

This register determines MEMCS# top of memory boundary. The top of memory boundary ranges from 2 MBytes-1 to 512 MBytes-1, in 2 MByte increments. This register is typically set to the top of main memory. Accesses ≥ 1 MByte and ≤ top of memory boundary results in the assertion of the MEMCS# signal (unless the address resides in the hole programmed via the MCSBOH and MCSTOH Registers). A value of 00h sets top of memory at 2 MBytes-1 (including the 2 MByte-1 address). A value of FFh sets the top of memory at 512 MByte-1 (including the 512 MByte-1 address).

Bit	Description
7:0	**Top of MEMCS# Memory Boundary:** Bits[7:0] correspond to address lines AD[28:21], respectively.

3.1.15 EADC1—EISA ADDRESS DECODE CONTROL 1 REGISTER

Address Offset 48–49h
Default value: 0001h
Attribute: Read/Write
Size: 16 bits

This 16-bit register specifies EISA-to-PCI mapping of the 0-1 MByte memory address range. For each bit position, the memory block is enabled if the corresponding bit = 1 and is disabled if the bit = 0. EISA or DMA memory cycles to the enabled blocks result in the EISA cycle being forwarded to the PCI Bus. For disabled memory blocks, the EISA memory cycle is not forwarded to the PCI Bus.

Bit	Description
15	**880–896 KBytes Memory Enable:** EISA-to-PCI mapping for this memory space is enabled when this bit is 1 and disabled when this bit is 0.
14	**864–880 KBytes Memory Enable:** EISA-to-PCI mapping for this memory space is enabled when this bit is 1 and disabled when this bit is 0.
13	**848–864 KBytes Memory Enable:** EISA-to-PCI mapping for this memory space is enabled when this bit is 1 and disabled when this bit is 0.
12	**832–848 KBytes Memory Enable:** EISA-to-PCI mapping for this memory space is enabled when this bit is 1 and disabled when this bit is 0.
11	**816–832 KBytes Memory Enable:** EISA-to-PCI mapping for this memory space is enabled when this bit is 1 and disabled when this bit is 0.
10	**800–816 KBytes Memory Enable:** EISA-to-PCI mapping for this memory space is enabled when this bit is 1 and disabled when this bit is 0.
9	**784–800 KBytes Memory Enable:** EISA-to-PCI mapping for this memory space is enabled when this bit is 1 and disabled when this bit is 0.
8	**768–784 KBytes Memory Enable:** EISA-to-PCI mapping for this memory space is enabled when this bit is 1 and disabled when this bit is 0.
7:3	**Reserved**
2	**640–768 KBytes VGA Memory Enable:** EISA-to-PCI mapping for this memory space is enabled when this bit is 1 and disabled when this bit is 0.
1	**512–640 KBytes Memory Enable:** EISA-to-PCI mapping for this memory space is enabled when this bit is 1 and disabled when this bit is 0.
0	**0–512 KBytes Memory Enable:** EISA-to-PCI mapping for this memory space is enabled when this bit is 1 and disabled when this bit is 0.

3.1.16 IORT—ISA I/O RECOVERY TIMER REGISTER

Address Offset: 4Ch
Default Value: 56
Attribute: Read/Write
Size: 8 bits

The I/O recovery logic is used to guarantee a minimum amount of time between back-to-back 8-bit and 16-bit PCI-to-ISA I/O slave accesses. These minimum times are programmable.

The I/O recovery mechanism in the PCEB is used to add recovery delay between PCI-originated 8-bit and 16-bit I/O cycles to ISA devices. The delay is measured from the rising edge of the EISA command signal (CMD#) to the falling edge of the next EISA command. The delay is equal to the number of EISA Bus clocks (BCLKs) that correspond to the value contained in bits [1:0] for 16-bit I/O devices and in bits[5:3] for 8-bit I/O devices. Note that no additional delay is inserted for back-to-back I/O ''sub-cycles'' generated as a result of byte assembly or disassembly. This register defaults to 8- and 16-bit recovery enabled with two clocks of I/O recovery.

Bit	Description
7	**Reserved**
6	**Bit I/O Recovery Enable:** This bit enables the recovery times programmed into bits 0 and 1 of this register. When this bit is set to 1, the recovery times shown for bits 5-3 are enabled. When this bit is set to 0, recovery times are disabled.
5:3	**8-Bit I/O Recovery times:** This 3-bit field defines the recovery times for 8-bit I/O. Programmable delays between back-to-back 8-bit PCI cycles to ISA I/O slaves is shown in terms of EISA clock cycles (BCLK). The selected delay programmed into this field is enabled/disabled via bit 6 of this register. **Bits [5:3]** **BCLK** 001 1 010 2 011 3 100 4 101 5 110 6 111 7 000 8
2	**16-Bit I/O Recovery Enable:** This bit enables the recovery times programmed into bits 0 and 1 of this register. When this bit is set to 1, the recovery times shown for bits 0 and 1 are enabled. When this bit is set to 0, recovery times are disabled.
1:0	**16-Bit I/O Recovery Times:** This 2-bit field defines the Recovery time for 16-bit I/O. Programmable delays between back-to-back 16-bit PCI cycles to ISA I/O slaves is shown in terms of EISA clock cycles (BCLK). The selected delay programmed into this field is enabled/disabled via bit 2 of this register. **Bits [1:0]B** **CLK** 01 1 10 2 11 3 00 4

3.1.17 MAR1—MEMCS# ATTRIBUTE REGISTER #1

Address Offset: 54h
Default Value: 00h
Attribute: Read/Write
Size: 8 bits

RE—Read Enable. When the RE bit (bit 6, 4, 2, 0) is set to a 1, the PCEB generates MEMCS# for PCI master, DMA, or EISA master memory read accesses to the corresponding segment in main memory. When the RE bit is set to a 0, the PCEB does not generate MEMCS# for PCI master, DMA, or EISA master memory read accesses to the corresponding segment. When the RE and WE bits are both 0 (or bit 4 in the MEMCS# Control Register is set to a 0-disabled), the PCI master, DMA, or EISA master can not access the corresponding segment in main memory.

WE—Write Enable. When the WE bit (bit 7, 5, 3, 1) is set to a 1, the PCEB generates MEMCS# for PCI master, DMA, or EISA master memory write accesses to the corresponding segment in main memory. When this bit is set to a 0, the PCEB does not generate MEMCS# for PCI master, DMA, or EISA master memory write accesses to the corresponding segment. When the RE and WE bits are both 0 (or bit 4 in the MEMCS# Control Register is set to a 0-disabled), the PCI master, DMA, or EISA master can not access the corresponding segment in main memory.

Bit	Description
7	**0CC000–0CFFFFh Add-on BIOS: WE:** 1 = Enable; 0 = Disable
6	**0CC000–0CFFFFh Add-on BIOS: RE:** 1 = Enable; 0 = Disable
5	**0C8000–0CBFFFh Add-on BIOS: WE:** 1 = Enable; 0 = Disable
4	**0C8000–0CBFFFh Add-on BIOS: RE:** 1 = Enable; 0 = Disable
3	**0C4000–0C7FFFh Add-on BIOS: WE:** 1 = Enable; 0 = Disable
2	**0C4000–0C7FFFh Add-on BIOS: RE:** 1 = Enable; 0 = Disable
1	**0C0000–0C3FFFh Add-on BIOS: WE:** 1 = Enable; 0 = Disable
0	**0C0000–0C3FFFh Add-on BIOS: RE:** 1 = Enable; 0 = Disable

3.1.18 MAR2—MEMCS# ATTRIBUTE REGISTER #2

Address Offset: 55h
Default Value: 00h
Attribute: Read/Write
Size: 8 bits

RE—Read Enable. When the RE bit (bit 6, 4, 2, 0) is set to a 1, the PCEB generates MEMCS# for PCI master, DMA, or EISA master memory read accesses to the corresponding segment in main memory. When this bit is set to a 0, the PCEB does not generate MEMCS# for PCI master, DMA, or EISA master memory read accesses to the corresponding segment. When the RE and WE bits are both 0 (or bit 4 in the MEMCS# Control Register is set to a 0-disabled), the PCI master, DMA, or EISA master can not access the corresponding segment in main memory.

WE—Write Enable. When the WE bit (bit 7, 5, 3, 1) is set to a 1, the PCEB generates MEMCS# for PCI master, DMA, or EISA master memory write accesses to the corresponding segment in main memory. When this bit is set to a 0, the PCEB does not generate MEMCS# for PCI master, DMA, or EISA master memory write accesses to the corresponding segment. When the RE and WE bits are both 0 (or bit 4 in the MEMCS# Control Register is set to a 0-disabled), the PCI master, DMA, or EISA master can not access the corresponding segment in main memory.

Bit	Description
7	**0DC000–0DFFFFh Add-on BIOS: WE:** 1 = Enable; 0 = Disable
6	**0DC000–0DFFFFh Add-on BIOS: RE:** 1 = Enable; 0 = Disable
5	**0D8000–0DBFFFh Add-on BIOS: WE:** 1 = Enable; 0 = Disable
4	**0D8000–0DBFFFh Add-on BIOS: RE:** 1 = Enable; 0 = Disable
3	**0D4000–0D7FFFh Add-on BIOS: WE:** 1 = Enable; 0 = Disable
2	**0D4000–0D7FFFh Add-on BIOS: RE:** 1 = Enable; 0 = Disable
1	**0D0000–0D3FFFh Add-on BIOS: WE:** 1 = Enable; 0 = Disable
0	**0D0000–0D3FFFh Add-on BIOS: RE:** 1 = Enable; 0 = Disable

3.1.19 MAR3—MEMCS# ATTRIBUTE REGISTER #3

Address Offset: 56h
Default Value: 00h
Attribute: Read/Write
Size: 8 bits

RE—Read Enable. When the RE bit (bit 6, 4, 2, 0) is set to a 1, the PCEB generates MEMCS# for PCI master, DMA, EISA master memory read accesses to the corresponding segment in main memory. When this bit is set to a 0, the PCEB does not generate MEMCS# for PCI master, DMA, or EISA master memory read accesses to the corresponding segment. When the RE and WE bits are both 0 (or bit 4 in the MEMCS# Control Register is set to a 0-disabled), the PCI master can not access the corresponding segment in main memory.

WE—Write Enable. When the WE bit (bit 7, 5, 3, 1) is set to a 1, the PCEB generates MEMCS# for PCI master , DMA, EISA master memory write accesses to the corresponding segment in main memory. When this bit is set to a 0, the PCEB does not generate MEMCS# for PCI master, DMA, or EISA master memory write accesses to the corresponding segment. When the RE and WE bits are both 0 (or bit 4 in the MEMCS# Control Register is set to a 0-disabled), the PCI master can not access the corresponding segment in main memory.

Bit	Description
7	**0EC000–0EFFFFh BIOS Extension: WE:** 1 = Enable; 0 = Disable
6	**0EC000–0EFFFFh BIOS Extension: RE:** 1 = Enable; 0 = Disable
5	**0E8000–0EBFFFh BIOS Extension: WE:** 1 = Enable; 0 = Disable
4	**0E8000–0EBFFFh BIOS Extension: RE:** 1 = Enable; 0 = Disable
3	**0E4000–0E7FFFh BIOS Extension: WE:** 1 = Enable; 0 = Disable
2	**0E4000–0E7FFFh BIOS Extension: RE:** 1 = Enable; 0 = Disable
1	**0E0000–0E3FFFh BIOS Extension: WE:** 1 = Enable; 0 = Disable
0	**0E0000–0E3FFFh BIOS Extension: RE:** 1 = Enable; 0 = Disable

3.1.20 PDCON—PCI DECODE CONTROL REGISTER

Address Offset: 58h
Default Value: 00h
Attribute: Read/Write
Size: 8 bits

This register enables/disables positive decode of PCI accesses to the IDE and 8259 locations residing in the expansion bus subsystem. For the 82374SB, this register controls the mode of address decode (subtractive or negative) for memory cycles on the PCI Bus.

Subtractive decoding:

PCI memory cycles that are not claimed on the PCI Bus (i.e., DEVSEL# inactive) are forwarded to the EISA Bus. This is the default on power up.

Negative decoding (82374SB Only):

PCI memory cycles that are not mapped to one of the regions defined by A, B, or C below, are immediately forwarded to the EISA Bus (i.e. without waiting for DEVSEL# time-out). PCI memory cycles that are decoded to one of the four programmable PCI memory regions, but are not claimed (DEVSEL# negated), are forwarded to the EISA Bus by subtractive decode.

A. Main memory locations defined by the MEMCS# mapping (MCSCON, MCSBOH, MCSTOH, MCSTOM, MAR1, MAR2, and MAR3 Registers).

B. The enabled Video Frame Buffer region, 0A0000–0BFFFFh (as indicated by bit 2 of the EADC1 Register).

C. The four programmable PCI memory regions (defined by the MEMREGN[4:1] registers).

NOTE:
If there are devices on the PCI that are not mapped into any of the regions defined by A, B, or C, then negative decoding can not be used.

Bit	Description
7:6	**Reserved**
5	**8259 Decode Control (8259DC):** This bit enables/disables positive decode of 8259 locations 0020h, 0021h, 00A0h and 00A1h. When this bit is 1, positive decode for these locations are enabled. When this bit is 0, positive decode for these locations is disabled. After reset, this bit is 0. Note that if positive decode is disabled, these 8259 locations can still be accessed via subtractive decode.
4	**IDE Decode Control (IDEDC):** This bit enables/disables positive decode of IDE locations 1F0–1F7h (primary) or 170–177h (secondary) and 3F6h,3F7h (primary) or 376h,377h (secondary). When IDEDC = 0, positive decode is disabled. When IDECD = 1, positive decode is enabled. After reset, this bit is 0. Note that if positive decode is disabled, these IDE locations can still be accessed via subtractive decode.
3:1	**Reserved**
0	**82375EB: Reserved.** Must be 0 when programming this register.
	82375SB: PCI Memory Address Decoding Mode (PMAD): This bit selects between subtractive and negative decoding. When PMAD = 1, negative decoding is selected. When PMAD = 0, subtractive decoding is selected. After reset, this bit is 0.

3.1.21 EADC2—EISA ADDRESS DECODE CONTROL EXTENSION REGISTER

Address Offset: 5Ah
Default value: 00h
Attribute: Read/Write
Size: 8 bits

This register specifies EISA-to-PCI mapping for the 896 KByte to 1 MByte memory address range (BIOS). If this memory block is enabled, EISA memory accesses in this range will result in the EISA cycles being forwarded to the PCI Bus. (Note that enabling this block is necessary if BIOS resides within the PCI and not within the EISA subsystem.)

This register also defines mapping for the 16 MByte - 64 KByte to 16 MByte memory address range. This mapping is important if the BIOS is aliased at the top 64 KBytes of 16 MBytes. If the region is enabled and this address range is within the hole defined by the MCSBOH and MCSTOH Registers or above the top of main memory defined by the MCSTOM Register, the EISA cycle is forwarded to the PCI.

Bit	Description
7:6	**Reserved**
5	**Top 64 KByte of 16 MByte Memory Space Enable (FF0000–FFFFFFh):** This memory block is enabled when this bit is 1 and disabled when this bit is 0.
4	**960 KBytes—1 MByte Memory Space Enable (0F0000–0FFFFFh):** This memory block is enabled when this bit is 1 and disabled when this bit is 0.
3	**944–960 KByte Memory Space Enable (0EC000–0EFFFFh):** This memory block is enabled when this bit is 1 and disabled when this bit is 0.
2	**928–944 KByte Memory Space Enable (0E8000–0EBFFFh):** This memory block is enabled when this bit is 1 and disabled when this bit is 0.
1	**912–928 KByte Memory Space Enable (0E4000–0E7FFFh):** This memory block is enabled when this bit is 1 and disabled when this bit is 0.
0	**896–912 KByte Memory Space Enable (0E0000–0E3FFFh):** This memory block is enabled when this bit is 1 and disabled when this bit is 0.

3.1.22 EPMRA—EISA-TO-PCI MEMORY REGION ATTRIBUTES REGISTER

Address Offset: 5Ch
Default value: 00h
Attribute: Read/Write
Size: 8 bits

This register defines buffering attributes for EISA accesses to PCI memory regions specified by MEMREGN[4:1] Registers. When an EPMRA bit is 1 (and the Line Buffers are enabled via the PCICON Register), EISA accesses to the corresponding PCI memory region are performed in buffered mode. In buffered mode, read prefetching and write posting/assembly are enabled. When an EPMRA bit is 0, EISA accesses to the corresponding PCI memory region are performed in non-buffered mode. In non-buffered mode, a buffer bypass path is used to complete the transaction.

NOTE:

1. Using buffered mode for EISA accesses to PCI memory regions that contain memory-mapped I/O devices can cause unintended side effects. In buffered mode, strong ordering is not preserved within a Dword. If the order of the writes to an I/O device is important, non-buffered mode should be used. Also, read-prefetch can cause unintended changes of status registers in the memory-mapped I/O device.
2. The Line Buffers are typically enabled or disabled during system initialization. These buffers should not be dynamically enabled/disabled during runtime. Otherwise, data coherency can be affected, if a buffer containing valid write data is disabled and then, later, re-enabled.

Bit	Description
7:4	**Reserved**
3	**Region 4 Attribute (REG-4):** EISA accesses to this PCI memory region are buffered when this bit is 1 and non-buffered when this bit is 0. If the Line Buffers are disabled via the PCICON Register (bit 6), buffering is disabled, regardless of the value of this bit.
2	**Region 3 Attribute (REG-3):** EISA accesses to this PCI memory region are buffered when this bit is 1 and non-buffered when this bit is 0. If the Line Buffers are disabled via the PCICON Register (bit 6), buffering is disabled, regardless of the value of this bit.
1	**Region 2 Attribute (REG-2):** EISA accesses to this PCI memory region are buffered when this bit is 1 and non-buffered when this bit is 0. If the Line Buffers are disabled via the PCICON Register (bit 6), buffering is disabled, regardless of the value of this bit.
0	**Region 1 Attribute (REG-1):** EISA accesses to this PCI memory region are buffered when this bit is 1 and non-buffered when this bit is 0. If the Line Buffers are disabled via the PCICON Register (bit 6), buffering is disabled, regardless of the value of this bit.

3.1.23 MEMREGN[4:1]—EISA-TO-PCI MEMORY REGION ADDRESS REGISTERS

Address Offset: 60-63h (Memory Region 1)
64-67h (Memory Region 2)
68-6Bh (Memory Region 3)
6C-6Fh (Memory Region 4)
Default Value: 0000FFFFh
Attribute: Read/Write
Size: 32 bits

These 32-bit registers provide four windows for EISA-to-PCI memory accesses. Each window defines a positively decoded programmable address region for mapping EISA memory space to the corresponding PCI memory space. This base and limit address fields define the size and location of the region within the 4 GByte PCI memory space. The base and limit addresses can be aligned on any 64 KByte boundary and each region can be sized in 64 KByte increments, up to the theoretical maximum size of 4 GByte. The default values of this register ensure that the regions are initially disabled.

A region is selected based on the following formula: Base Address $\leq$ address $\leq$ Limit Address.

Bit	Description
31:16	**Memory Region Limit Address:** For EISA-to-PCI accesses, bits[31:16] correspond to address lines LA[31:16] on the EISA Bus and AD[31:16] on the PCI Bus. This field determines the limit address of the memory region within the 4 GByte PCI memory space.
15:0	**Memory Region Base Address:** For EISA-to-PCI accesses, bits[15:0] correspond to address lines LA[31:16] on the EISA Bus and AD[31:16] on the PCI Bus. This field determines the starting address of the memory region within the 4 GByte PCI memory space.

3.1.24 IOREGN[4:1]—EISA-TO-PCI I/O REGION ADDRESS REGISTERS

Address Offset: 70-73h (I/O Region 1)
74-77h (I/O Region 2)
78-7Bh (I/O Region 3)
7C-7Fh (I/O Region 4)
Default value: 0000FFFCh
Attribute: Read/Write
Size: 32 bits

These 32-bit registers provide four windows for EISA-to-PCI I/O accesses. The windows define positively decoded programmable address regions for mapping EISA I/O space to the corresponding PCI I/O space. Each register determines the starting and limit addresses of the particular region within the 64 KByte PCI I/O space. The base and limit addresses can be aligned on any Dword boundary and each region can be sized in Dword increments (32-bits) up to the theoretical maximum size of 64 KByte. Default values for the base and limit fields ensure that the regions are initially disabled.

The I/O regions are selected based on the following formula: Base Address ≤ address ≤ Limit Address.

Bit	Description
31:18	**I/O Region Limit Address:** For EISA-to-PCI I/O accesses, bits[31:18] correspond to address lines LA[15:2] on the EISA Bus and AD[15:2] on the PCI Bus. This field determines the limit address of the region within the 64 KByte PCI I/O space.
17:16	**Reserved**
15:2	**I/O Region Base Address:** For EISA-to-PCI I/O accesses, bits[15:2] correspond to address lines LA[15:2] on the EISA Bus and AD[15:2] on the PCI Bus. This field determines the starting address of the region within the 65 KByte PCI I/O space.
1:0	**Reserved**

3.1.25 BTMR—BIOS TIMER BASE ADDRESS REGISTER

Address Offset: 80–81h
Default value: 0078h
Attribute: Read/Write
Size: 16 bits

This 16-bit register determines the base address for the BIOS Timer Register located in PCI I/O space. The BIOS Timer resides in the PCEB and is the only internal resource mapped to PCI I/O space. The base address can be set at Dword boundaries anywhere in the 64 KByte PCI I/O space. This register also provides the BIOS Timer access enable/disable control bit.

Bit	Description
15:2	**BIOS Timer Base Address:** Bits[15:2] correspond to PCI address lines AD[15:2].
1	**Reserved**
0	**BIOS Timer Enable (BTE):** When BTE = 1, the BIOS Timer is enabled. When BTE = 0, the BIOS Timer is disabled. The default is 0 (disabled).

3.1.26 ELTCR—EISA LATENCY TIMER CONTROL REGISTER

Address Offset: 84h
Default value: 7Fh
Attribute: Read/Write
Size: 8 bits

This register provides the control for the EISA Latency Timer (ELT). The register holds the initial count value used by the ELT. The ELT uses the PCI clock for counting . The ELT time-out period is equal to:

$$ELT_{timeout} = Value\{ELTCR(7:0)\} \times T_{pciclck} \text{ [ns]}$$

where:

Tpciclk = 30 ns at 33 MHz (40 ns at 25 MHz).

Therefore, a maximum ELT time-out period at 33 MHz is 256 x 30 ns = 7.68 μs. The value written into this register is system dependent. It should be based on PCI latency characteristics controlled by the PCI Master Latency Timer mechanism and on EISA Bus arbitration/latency parametrics. A typical value corresponds to the ELT time-out period of 1–3 μs. When the value in the ELTCR Register is 0, the ELT mechanism is disabled. The ELTCR Register must be initialized before EISA masters or DMA are enabled.

Bit	Description
7:0	**EISA Latency Timer Count Value:** Bits[7:0] contain the initial count value for the EISA Latency Timer. When this field contains 00h, the EISA Latency Timer is disabled.

3.2 I/O Registers

The only PCEB internal resource mapped to the PCI I/O space is the BIOS Timer Register.

3.2.1 BIOSTM—BIOS TIMER REGISTER

Register Location: Programmable I/O address location (Dword aligned)
Default Value: 00 00 xx xxh
Attribute: Read/Write
Size: 32 bits

This 32-bit register is mapped to the PCI I/O space location determined by the value in the BTMR Register. Bit 0 of BTMR must be 1 to enable access to the BIOS Timer. The BIOS timer clock is derived from the EISA Bus clock (BCLK); either 8.25 or 8.33 MHz depending on the PCI clock. BCLK is divided by 8 to obtain the timer clock of 1.03 or 1.04 MHz. If a frequency other than 33 MHz or 25 MHz is used for PCI clock, the BIOS Timer clock will be affected. (It will always keep the same relation to the BCLK, i.e. 1:4 or 1:3, depending on the clock divisor.) The BIOS Timer is only accessible from the PCI Bus and is not accessible from the EISA Bus.

After data is written into BIOS Timer Register (BE1# and/or BE0# must be asserted), the BIOS timer starts decrementing until it reaches zero. It ''freezes'' at zero until the new count value is written.

Bit	Description
31:16	**Reserved**
15:0	**BIOS Timer Count Value:** The initial count value is written to bits[15:0] to start the timer. The value read is the current value of the BIOS Timer.

4.0 ADDRESS DECODING

Conceptually, the PCEB contains two programmable address decoders: one to decode PCI Bus cycles that need to be forwarded to the EISA Bus or serviced internally and the other to decode EISA Bus cycles that need to be forwarded to the PCI Bus. Two decoders permit the PCI and EISA Buses to operate concurrently (Figure 2). The PCEB can be programmed to respond to certain PCI memory or I/O region accesses as well as configuration space accesses to the PCEB's internal configuration registers. PCEB address decoding is discussed in Section 4.1.

The EISA address decoder decodes EISA Bus cycles generated by the bus master (DMA controller, ISA compatible master, or EISA compatible master) that need to be forwarded to the PCI Bus. The EISA decode logic can be programmed to respond to certain memory or I/O region accesses.

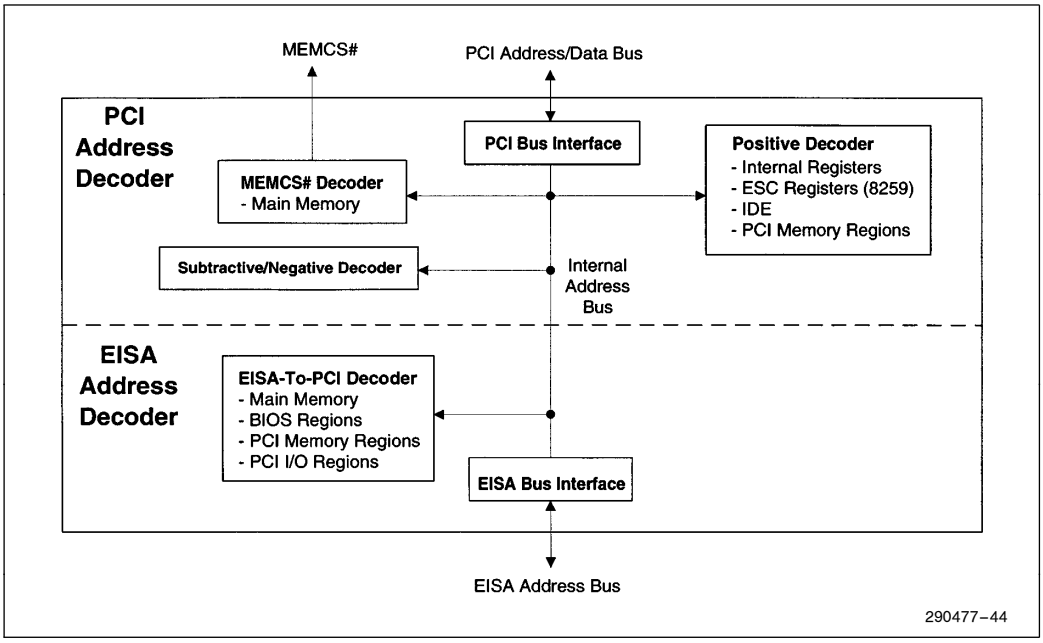


Figure 2. Block Diagram of Address Decoder

The PCEB provides three methods for decoding the current PCI Bus cycle. The PCEB can use positive, subtractive, or negative (82374SB only) decoding for these cycles, depending on the type of cycle, actions on the PCI Bus, and programming of the PCEB registers. For EISA Bus cycles, only positive decoding is used.

1. **Positive decoding**. With positive decoding, the PCI/EISA Bus cycle address is compared to the corresponding address ranges set up in the PCEB for positive decode. A match causes the PCEB decode logic to immediately service the cycle. The PCEB can be programmed (via the configuration registers) to positively decode selected memory or I/O accesses on both the PCI Bus and EISA Bus. Depending on the programming of the internal registers, the PCEB provides positive decoding for PCI accesses to selected address ranges in memory and I/O spaces and for EISA accesses to selected address ranges in memory and I/O spaces. Note that the decoding method for PCI accesses to the PCEB internal registers (configuration and I/O space registers) is not programmable and these accesses are always positively decoded.

2. **Subtractive decoding**. For PCI memory or I/O cycles, the PCEB uses subtractive decoding (or negative decoding, described in #3 of this list for the 82375SB) to respond to addresses that are not positively decoded. With subtractive decoding, if a memory or I/O cycle is not claimed on the PCI Bus (via DEVSEL#), the PCEB forwards the cycle to the EISA Bus. The PCEB waits a programmable number of PCICLKs (1 to 3 PCICLKs, as selected via the PCICON Register) for a PCI agent to claim the cycle. If the cycle is not claimed within the programmed number of PCICLKs (DEVSEL# time-out), the PCEB claims the cycle (asserts DEVSEL#) and forwards it to the EISA Bus. Note that the number of PCICLKs for a DEVSEL# time-out should be programmed to accommodate the slowest PCI Bus device.
3. **Negative decoding**. For the 82375SB, negative decoding is a programmable option (via the PDCON Register) that is only used for PCI memory cycles. With negative decoding, a PCI memory cycle that is not positively decoded by the PCEB as a main memory area (one of the MEMCS# generation areas) and is not in one of the four programmable EISA-to-PCI memory regions (defined by MEMREGN[4:1]) is immediately forwarded to the EISA Bus. This occurs without waiting for a DEVSEL# time-out to see if the cycle is going to be claimed on the PCI Bus. Thus, negative decoding can reduce the latency incurred by waiting for a DEVSEL# time-out that is associated with subtractive decoding. This increases throughput to the EISA Bus for unclaimed PCI memory cycles. If the DEVSEL# time-out is set to a 2 PCICLKtime-out, the latency is reduced by 1 PCICLK and for a 3 PCICLK time-out, the latency is reduced by 2 PCICLKs. For more information on negative (and subtractive) decoding, see Section 4.1.1.3, Subtractively and Negatively Decoded Cycles to EISA.

   Note that negative decoding imposes a restriction on the PCI system memory address map. PCI memory-mapped devices are restricted to one of the four programmable EISA-to-PCI regions (MEMREGN[4:1]). These regions always use subtractive decoding to forward an unclaimed cycle to the EISA Bus, even if negative decoding is enabled. Locating devices in these regions ensures that the PCI device has the allotted number of programmed PCICLKs (DEVSEL# time-out) to respond with DEVSEL#. Further, since the PCEB does not negatively decode I/O space addresses, enabling this feature does not impose restrictions on devices that are mapped to PCI I/O space.

4.1 PCI Cycle Address Decoding

The PCEB decodes addresses presented on the multiplexed PCI address/data bus during the address bus phase. AD[31:0] and the byte enables (C/BE[3:0]# during the data phase) are used for address decoding. C/BE[3:0]# are used during the data phase to indicate which byte lanes contain valid data. For memory cycles, the PCI address decoding is always a function of AD[31:2]. In the case of I/O cycles, all 32 address bits (AD[31:0]) are used to provide addressing with byte granularity. For configuration cycles, only a subset of the address lines carry address information.

The PCEB decodes the following PCI cycle addresses based on the contents of the relevant programmable registers:

1. Positively decodes PCEB configuration registers.
2. Positively decodes I/O addresses contained within the PCEB (BIOS Timer).
3. Positively decodes the following compatibility I/O registers to improve performance: a) Interrupt controller (8259) I/O registers contained within the ESC to optimize interrupt processing, if enabled through the PDCON Register, b) IDE registers, if enabled through the PDCON Register.
4. Positively decodes four programmable memory address regions contained within the PCI memory space.

5. Positively decodes memory addresses for selected regions of main memory (located behind the Host/PCI Bridge). When a main memory address is positively decoded, the PCEB asserts the MEMCS# signal to the Host/PCI Bridge. The PCEB does not assert DEVSEL#.
6. Subtractively or negatively (82375SB only) decodes cycles to the EISA Bus (see Section 4.1.1, Memory Space Address Decoding) .

NOTE:

A PCI requirement is that, upon power-up, PCI agents do not respond to any address. Typically, the only access to a PCI agent is through the IDSEL configuration mechanism until the agent is enabled during initialization. The PCEB/ESC subsystem is an exception to this since it controls access to the BIOS boot code. The PCEB subtractively decodes BIOS accesses and passes the accesses to the EISA Bus where the ESC generates BIOS chip select. This allows BIOS memory to be located in the PCI memory space.

4.1.1 MEMORY SPACE ADDRESS DECODING

The MCSCON, MCSTOP, MCSBOH, MCSTOM, and PDCON Registers are used to program the decoding for PCI Bus memory cycles.

4.1.1.1 Main Memory Decoding (MEMCS#)

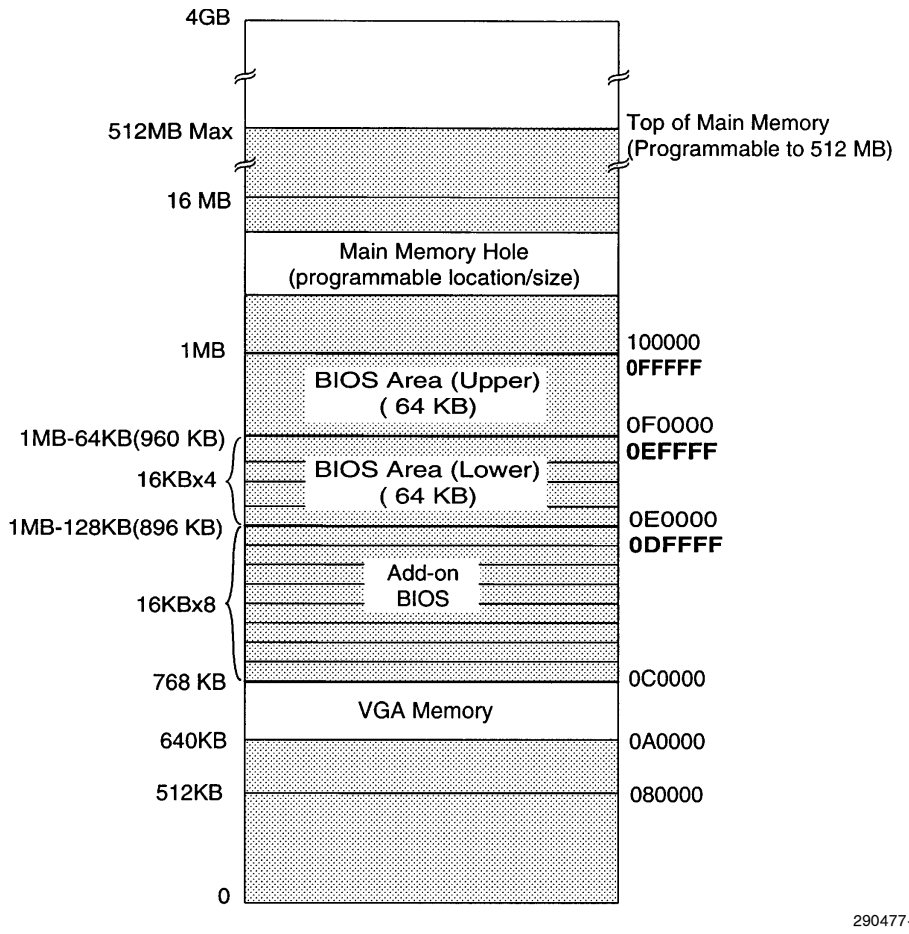
The PCEB supports positive decode of main memory areas by generating a memory chip select signal (MEMCS#) to the Host/PCI Bridge that contains the main memory interface control. The PCEB supports memory sizes up to 512 MBytes (i.e., the PCEB can be programmed to generate MEMCS# for this memory range). For PCI memory accesses above 512 MByte (512 MBytes to 4 GBytes), the PCEB does not generate MEMCS# and unclaimed cycles are forwarded to the EISA Bus using either subtractive or negative (82374SB only) decoding.

If a memory region is enabled, accesses to that region are positively decoded and result in the PCEB asserting MEMCS#. If a memory region is disabled, accesses do not generate MEMCS# and the cycle is either subtractively or negatively (82374SB only) decoded and forwarded to the EISA Bus.

Within the 512 MByte main memory range, the PCEB supports the enabling/disabling of sixteen individual memory ranges (Figure 3). Fourteen of the ranges are within the 640 KByte - 1 MByte area and have Read Enable (RE) and Write Enable (WE) attributes. These attributes permit positive address decoding for reads and writes to be independently enabled/disabled. This permits, for example, an address range to be positively decoded for a memory read and subtractively or negatively (82374SB only) decoded to the EISA Bus for a memory write.

The fifteenth range (0–512 KByte) and sixteenth range (programmable limit address from 2 MByte up to 512 MByte on 2 MByte increments) can be enabled or disabled but do not have RE/WE attributes. A seventeenth range is available that identifies a memory hole. Addresses within this hole will not generate a MEMCS#. These memory address ranges are:

- 0–512 KByte
- 512–640 KByte
- 640–768 KBytes (VGA memory page)
- 960 KByte to 1 MByte (BIOS Area)
- 768–896 KByte in 16 KByte segments (total of 8 segments)
- 896–960 KByte in 16 KByte segments (total of 4 segments)
- 960 KByte to 1 MByte (Upper BIOS area)
- 1–512 MByte in 2 MByte increments.
- Programmable memory hole in 64 KByte increments between 1 MByte and 16 MByte.



NOTE:
Shaded areas represent main memory decode regions for MEMCS#.

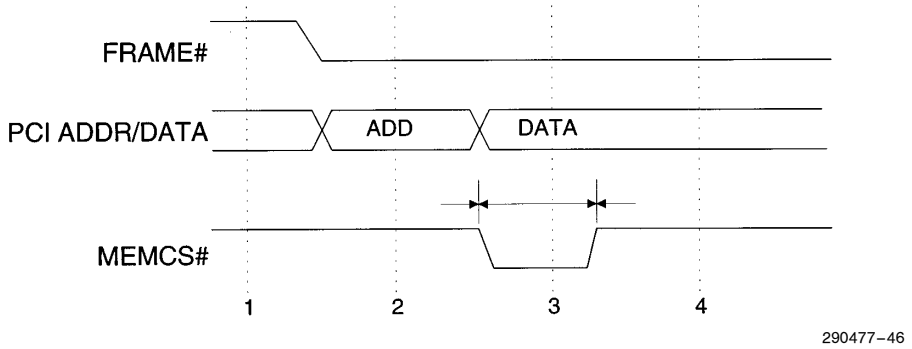
Figure 3. MEMCS# Decode Areas

Table 2 summarizes the attribute registers used in MEMCS# decoding. The MCSCON, MAR1, MAR2, and MAR3 Registers are used to assign RE/WE attributes to a particular memory range. The MEMCS# hole is programmed using the MCSTOH and MCSBOH Registers. The region above 1 MByte is programmed using the MCSTOM Register. The region from 0–512 KByte is enabled/disabled using bit 4 of the MCSCON Register. MCSCON bit 4 is also used to enable and disable the entire MEMCS# function.

Table 2. Read Enable/Write Enable Attributes For MEMCS# Decoding

Memory Attribute Registers (Register Bits are Shown in Brackets)	Attribute	Memory Segments	Comments
MCSCON[1:0]	WE RE	080000–09FFFFh	512K to 640K
MCSCON[3:2]	WE RE	0F0000–0FFFFFh	BIOS Area
MAR1[1:0]	WE RE	0C0000–0C3FFFh	Add-on BIOS
MAR1[3:2]	WE RE	0C4000–0C7FFFh	Add-on BIOS
MAR1[5:4]	WE RE	0C8000–0CBFFFh	Add-on BIOS
MAR1[7:6]	WE RE	0CC000–0CFFFFh	Add-on BIOS
MAR2[1:0]	WE RE	0D0000–0D3FFFh	Add-on BIOS
MAR2[3:2]	WE RE	0D4000–0D7FFFh	Add-on BIOS
MAR2[5:4]	WE RE	0D8000–0DBFFFh	Add-on BIOS
MAR2[7:6]	WE RE	0DC000–0DFFFFh	Add-on BIOS
MAR3[1:0]	WE RE	0E0000–0E3FFFh	BIOS Extension
MAR3[3:2]	WE RE	0E4000–0E7FFFh	BIOS Extension
MAR3[5:4]	WE RE	0E8000–0EBFFFh	BIOS Extension
MAR3[7:6]	WE RE	0EC000–0EFFFFh	BIOS Extension

The PCEB generates MEMCS# from the decode of the PCI address. MEMCS# is asserted during the first data phase as indicated in the Figure 4. MEMCS# is only asserted for one PCI clock period. The PCEB does not take any other action as a result of this decode, except to generate MEMCS#. It is the responsibility of the device using the MEMCS# signal to generate DEVSEL#, TRDY# and any other cycle response. The device using the MEMCS# will always generate DEVSEL# on the next clock. This fact can be used to avoid an extra clock delay in the subtractive decoder described in the next section.



NOTE:
Since MEMCS# is point-to-point (a sideband signal), the signal meets the guaranteed setup time to clock edge 3 (and clock edge 4). This fast generation of MEMCS# prevents the penalty caused by the decoding delay.

Figure 4. MEMCS# Generation

4.1.1.2 BIOS Memory Space

The BIOS memory space is subtractively decoded. BIOS is typically ''shadowed'' after configuration and initialization is complete. Thus, negative decoding is not implemented for accesses to the BIOS EPROM residing on the expansion bus.

The ESC decoder supports BIOS space up to 512 KBytes. The standard 128 KByte BIOS memory space is 000E 0000h to 000F FFFFh (top of 1 MByte), and aliased at FFFE 0000h to FFFF FFFFh (top of 4 GByte) and FFEE 0000h to FFEF FFFFh (top of 4 GByte -1 MByte). These aliased regions account for the CPU reset vector and the uncertainty of the state of the A20Gate when a software reset occurs.

Note that the ESC component contains the BIOS space decoder that provides address aliasing for BIOS at 4 GByte or 4 GByte - 1 MByte by ignoring the LA20 address line.

The additional 384 KByte BIOS memory space at FFF8 0000h to FFFD FFFFh is known as the enlarged BIOS memory space. Note that EISA memory (other than BIOS) must not reside within the address range from 4 GByte -1.5 MByte to 4 GByte - 1 MByte and from 4 GByte - 512 KByte to 4 GByte to avoid conflict with BIOS space.

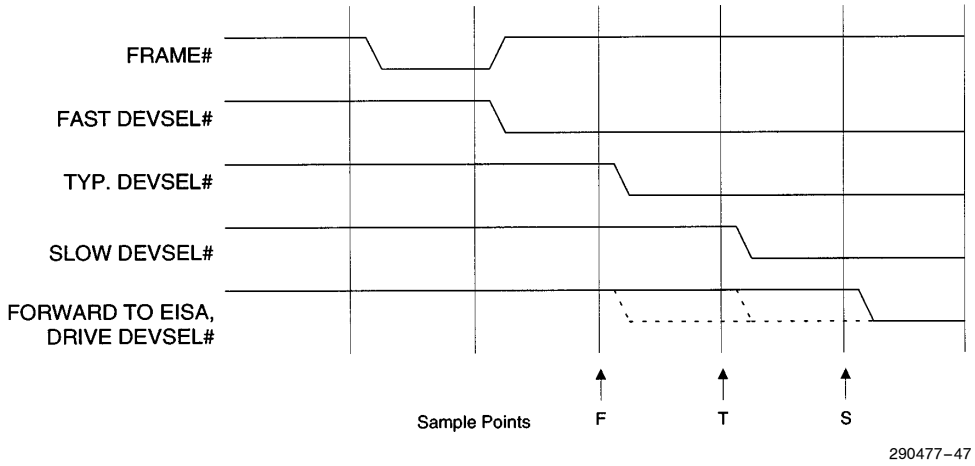
Since the BIOS device is 8 or 16 bits wide and typically has very long access times, PCI burst reads from BIOS space invoke a disconnect target termination (using the STOP# signal) after the first data transaction in order to meet the PCI incremental latency guidelines.

4.1.1.3 Subtractively And Negatively Decoded Cycles To EISA

The PCEB uses subtractive and negative (82375SB only) decoding to forward PCI Bus cycles to the EISA Bus. These modes are defined at the beginning of section 4.0. Bit 0 of the PDCON Register selects between negative and subtractive decoding.

For subtractive decoding on the 82375EB, the DEVSEL# sample point (Figure 5) can be configured to two different settings by programming the PCICON Register. If the ''typical'' point is selected, DEVSEL# is sampled at T, and, if inactive, the cycle is forwarded to EISA. If the ''slow'' point is selected, DEVSEL# is sampled at F, T, and S. The sample point should be configured to match the slowest PCI device in the system. This programmable capability (T or S) permits systems to optimize the DEVSEL# time-out latency to the response capabilities of the PCI devices in the system. The sample point selected must accommodate the slowest device on the PCI Bus. Note that when these unclaimed cycles are forwarded to the EISA Bus, the PCEB drives the DEVSEL# active. An active MEMCS# always results in an active DEVSEL# on the ''Typical'' sample point.

For subtractive decoding on the 82375SB, the DEVSEL# sample point can be configured to two different settings by programming the PCICON Register (slow and typical).



NOTE:
Fast DEVSEL# is not supported on the 82378EB or 82375SB.

Figure 5. DEVSEL# Sample Points

Only unclaimed PCI cycles within the memory address range from 0 to 4 GByte and I/O address range from 0 to 64 KByte are forwarded to EISA. Unclaimed PCI I/O cycles to address locations above 64 KBytes are not forwarded to the EISA Bus and the PCEB does not respond with DEVSEL#. In this case, these unclaimed cycles cause the master to terminate the PCI cycle with a master abort.

For the 82374SB, if negative decoding is used, the PCEB begins the PCI-to-EISA cycle forwarding process at the ''fast'' sample point. Compared to the system that uses subtractive decode at the ''slow'' sample point, negative decoding reduces the decoding overhead by 2 PCI clock cycles. In the case of subtractive decode at the ''typical'' sampling point, negative decoding reduces the overhead by 1 PCI clock.

The PCEB contains programmable configuration registers that define address ranges for PCI resident devices. There is a set of registers associated with MEMCS# decoding of main memory areas and set of registers for defining address mapping of up to four memory regions that are mapped to PCI for EISA Bus initiated cycles. Note that on the 82375EB, there is no equivalent mechanism for mapping the PCI memory regions to EISA and, therefore, all PCI memory cycles that need to be forwarded to the EISA Bus use subtractive decoding.

For the 82374SB, when negative decoding is selected, memory cycles with addresses other than those specified by the MEMCS# mapping for positive decode (via the MCSCON, MCSBOH, MCSTOH, MCSTOM, MAR1, MAR2, and MAR3 Registers) or the four programmable EISA-to-PCI memory regions (via MEMREGN[4:1]) are immediately forwarded to the EISA Bus without waiting for a DEVSEL# time-out.

Negative decoding has the following properties.

- All addresses above the top of main memory or within the MEMCS# hole (as defined by the MEMCS# map) are negatively decoded to EISA, except for the four programmable EISA-to-PCI memory regions. These regions (MEMREGN[4:1]) can overlap with active main memory ranges, the main memory hole, or with the memory space above the top of main memory. PCI accesses to MEMREGN[4:1] are always subtractively decoded to EISA.
- All addresses within MEMCS# defined ranges 640 KByte to 1 MByte can be either mapped to PCI or EISA using positive decoding. Some of these regions allow more detailed mapping based on programmable access attributes (read enable and write enable). This permits a region to be positively decoded for the enabled attribute and negatively decoded, if enabled, to the EISA Bus for the disabled attribute. For example, if a region is enabled for reads and disabled for writes, accesses to the region are positively decoded to the PCI for reads and negatively decoded, if enabled, to EISA for writes. If negative decoding is disabled (i.e., subtractive decoding enabled), the write is subtractively decoded to EISA.
- When negative decoding is enabled, MEMREGN[4:1] can still be set up for subtractive decoding. A PCI device that requires subtractive decoding must reside within Region [4:1]. As a result, the subtractive decoding penalty is only associated with some address ranges (i.e. some devices) and not with all non-PCI ranges. This feature can be used with PCI devices that dynamically change response on PCI cycles based on cycle type or an internal device state (e.g. intervention cycle).

  If a PCI device can not be located in one of the regions (Region [4:1]), then negative decoding can not be used. This could occur for systems with very specific address mapping requirements or systems where the device addresses that reside on the PCI Bus are highly fragmented and could not be accommodated with four regions.

  Note that the four regions do not limit mapping to only four devices. More than one device can be mapped into the same programmable region. These devices will reside within their own sub-regions, which are not necessarily contiguous.

4.1.2 PCEB CONFIGURATION REGISTERS

PCI accesses to the PCEB configuration registers are positively decoded. For a detailed address map of the PCEB configuration registers, see Section 3.1, Configuration Registers.

4.1.3 PCEB I/O REGISTERS

The only I/O-mapped register in the PCEB is the BIOS Timer Register. Section 3.2 provides details on the address mapping of this register. Note that the internal decode of the BIOS Timer Register is disabled after reset and all I/O accesses that are not contained within the PCI are subtractively decoded and passed to EISA Bus. To enable I/O access to the PCEB's BIOS Timer Register, The BTMR Register must be programmed.

4.1.4 POSITIVELY DECODED COMPATIBILITY I/O REGISTERS

The 8259 interrupt controller and IDE register locations are positively decoded. Access to the corresponding I/O address ranges must first be enabled through the PDCON Register.

PCI accesses to these registers are broadcast to the EISA Bus. These PCI accesses require the ownership of the EISA Bus, and will be retried if the EISA Bus is owned by an EISA/ISA master or the DMA.

4.1.4.1 ESC Resident PIC Registers

Access to the 8259 registers are positively decoded, if enabled through PDCON Register, to minimize access time to the system interrupt controller during interrupt processing (in particular during the EOI command sequence). Table 3 shows the 8259 I/O address map. After PCIRST#, positively decoded access to these address ranges is disabled.

Table 3. ESC Resident Programmable Interrupt Controller (PIC) Registers

Address (hex)	Address Bits FEDC	Address Bits BA98	Address Bits 7654	Address Bits 3210	Access Type	Register Name
0020h	0000	0000	001x	xx00	R/W	INT 1 Control
0021h	0000	0000	001x	xx01	R/W	INT 1 Mask
00A0h	0000	0000	101x	xx00	R/W	INT 2 Control
00A1h	0000	0000	101x	xx01	R/W	INT 2 Mask

4.1.4.2 EISA Resident IDE Registers

The PCI address decoder positively decodes IDE I/O addresses (Primary and Secondary IDE) that exist within the EISA subsystem (typically on the X-bus or as an ISA slave). This feature is implemented to minimize the decoding penalty for the systems that use IDE as a mass-storage controller. Table 4 shows IDE's I/O address map. Note that the PDCON Register controls the enable/disable function for IDE decoding. After PCIRST#, positive decode of the IDE address range is disabled.

Table 4. EISA Resident IDE Registers

Address (hex)	Addess (Bits)				Access Type	Register Name
	FEDC	BA98	7654	3210		
0170h	0000	0001	0111	0000	R/W	Secondary Data Register
0171h	0000	0001	0111	0001	R/W	Secondary Error Register
0172h	0000	0001	0111	0010	R/W	Secondary Sector Count Register
0173h	0000	0001	0111	0011	R/W	Secondary Sector Number Register
0174h	0000	0001	0111	0100	R/W	Secondary Cylinder Low Register
0175h	0000	0001	0111	0101	R/W	Secondary Cylinder High Register
0176h	0000	0001	0111	0110	R/W	Secondary Drive/head Register
0177h	0000	0001	0111	0111	R/W	Secondary Status Register
01F0h	0000	0001	1111	0000	R/W	Primary IDE Data Register
01F1h	0000	0001	1111	0001	R/W	Primary Error Register
01F2h	0000	0001	1111	0010	R/W	Primary Sector Count Register
01F3h	0000	0001	1111	0011	R/W	Primary Sector Number Register

Table 4. EISA Resident IDE Registers (Continued)

Address (hex)	Addess (Bits)				Access Type	Register Name
	FEDC	BA98	7654	3210		
01F4h	0000	0001	1111	0100	R/W	Primary Cylinder Low Register
01F5h	0000	0001	1111	0101	R/W	Primary Cylinder High Register
01F6h	0000	0001	1111	0110	R/W	Primary Drive/head Register
01F7h	0000	0001	1111	0111	R/W	Primary Status Register
0376h	0000	0011	0111	0110	R/W	Secondary Alternate Status Register
0377h	0000	0011	0111	0111	R	Secondary Drive Address Register
03F6h	0000	0011	1111	0110	R/W	Primary Alternate Status Register
03F7h	0000	0011	1111	0111	R	Primary Drive Address Register

4.2 EISA Cycle Address Decoding

For EISA Bus cycles, the PCEB address decoder determines the destination of EISA/ISA master and DMA cycles. This decoder provides the following functions:

- Positively decodes memory and I/O addresses that have been programmed into the PCEB for forwarding to the PCI Bus. This includes accesses to devices that reside directly on the PCI (memory Regions [4:1] and I/O Regions [4:1]) and segments of main memory that resides behind the Host/PCI Bridge.
- Provides access attributes for memory Regions [4:1]. These attributes are used to select the most optimum access mode (buffered or non-buffered).
- All cycles that are not positively decoded to be forwarded to PCI are contained within EISA.

NOTE:
The registers that reside in the PCEB (configuration registers and BIOS Timer) are not accessible from the EISA Bus.

4.2.1 POSITIVELY DECODED MEMORY CYCLES TO MAIN MEMORY

The EISA/ISA master or DMA addresses that are positively decoded by the PCEB are forwarded to the PCI Bus. If the address is not positively decoded by the PCEB, the cycle is not forwarded to the PCI Bus. Subtractive and negative (82374SB only) decoding are not used on the EISA Bus.

The PCEB permits several EISA memory address ranges (items a-i) to be positively decoded. EISA Bus cycles to these regions are forwarded to the PCI Bus. Regions described by a-f and h are fixed and can be enabled or disabled independently. These regions are controlled by the EADC1 and EADC2 Registers.

The region described by g defines a space starting at 1 MByte with a programmable upper boundary of 4 GByte - 2 MByte. Within this region a hole can be opened. Its size and location are programmable to allow a hole to be opened in memory space (for a frame buffer on the EISA Bus, for example). The size of this region and the hole are controlled by the MCSTOM, MCSBOH and MCSTOH Registers. If a hole in main memory is defined, then accesses to that address range are contained within EISA, unless defined by the EISA-to-PCI memory regions as a PCI destined access. (See next section.)

a. 0–512 KByte

b. 512–640 KByte

c. 640–768 KByte (VGA memory)

d. 768–896 KByte in eight 16 KByte sections (Expansion ROM)

e. 896–960 KByte in four 16 KByte sections (lower BIOS area)

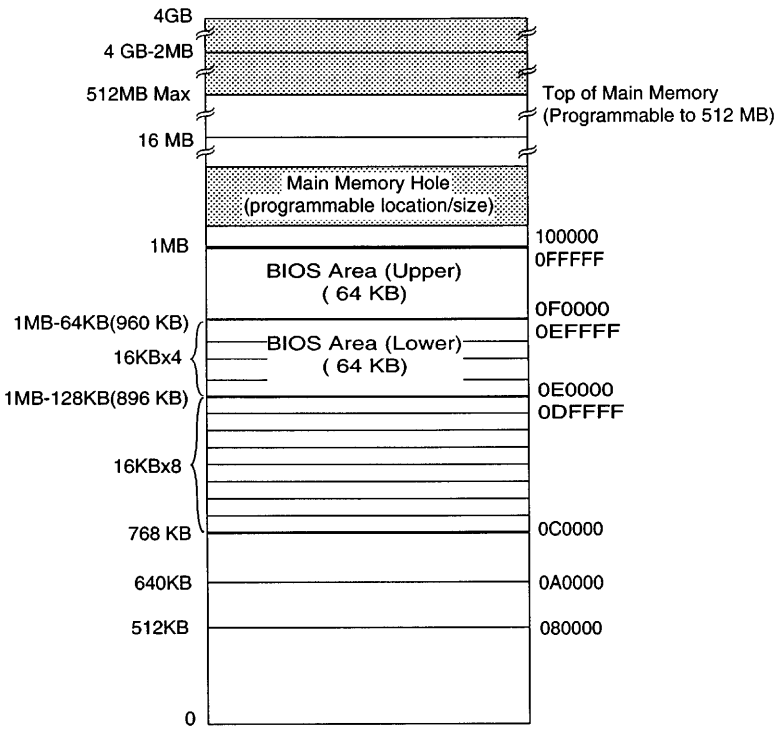
f. 960 KByte to 1 MByte (upper BIOS area)

g. 1 MByte to the top of memory (up to 4 GByte–2 MByte) within which a hole can be opened. Accesses to the hole are not forwarded to PCI. The top of the region can be programmed on 2 MByte boundaries up to 4 GByte–2 MByte. The hole can be between 64 KByte and 4 GByte–2 MByte in 64 KByte increments and located on any 64 KByte boundary.

h. 16 MByte–64 KByte to 16 MByte (FF0000–FFFFFFh). EISA memory cycles in this range are always forwarded to the PCI Bus, if this range exists in main memory as defined by the MEMCS# registers. In this case, the enable/disable control bit in EADC2 Register is a don't care. If this range is not defined in main memory (i.e., above the top of memory or defined as a hole in the main memory), EISA cycles to this address range are forwarded to the PCI Bus, based on the enable/disable bit in the EADC2 Register. (This capability is used to support access of BIOS at 16 MBytes.)

i. 4 GByte–2 MByte to 4 GByte. The address map must be programmed in a such way that this address range is always contained within EISA. This is to avoid conflict with local BIOS memory response in this address range. If this region must be mapped to PCI, then programming of the BIOS decoder Registers contained within the ESC must ensure that there is no conflict. To map this region to PCI, one of the four programmable EISA-to-PCI memory regions must be used. Mapping of this region to the PCI might be required in the case when BIOS resides on the PCI and the PCI/EISA system must have consistent address maps for both PCI and EISA.

For detailed information on the PCEB registers used to control these address regions, refer to Section 3.1, PCEB Configuration Registers.



290477–48

NOTES:

1. The four programmable EISA-to-PCI regions (Regions[4:1]) are not shown. These regions can be located anywhere within the 4 GByte memory space. Accesses to these regions are forwarded to the PCI Bus.
2. EISA cycles that access shaded regions are contained to the EISA Bus, unless the access hits one of the four programmable EISA-to-PCI regions described in note 1.
3. Memory accesses to non-shaded areas are forwarded to the PCI Bus, if the regions is enabled. If the region is disabled, the access is contained to the EISA Bus, unless the access hits one of the four programmable EISA-to-PCI regions described in note 1.

Figure 6. EISA Address Decoder Map

EISA memory cycles positively decoded for forwarding to PCI are allowed to be handled by the PCEB's Line Buffer management logic, if the line buffering is enabled through the PCICON Register.

For EISA-to-PCI transactions there are 2 modes of operation of the PCEB's Line Buffers:

- Buffered: Read-prefetch, write posting with data assembly.
- Non-buffered: Bypass path used.

Accesses within the main memory address range are normally performed in buffered mode. If there are programmable memory regions defined within the main memory hole or above the top of the main memory MEMREGN[4:1], then the mode of access depends on configuration bits of the EPMRA Register. Access attribute bits associated with these regions override the default buffered mode for a particular address range in the case of programmable regions overlapping with active main memory regions.

Access to the 64 KByte area at the top of 16 MBytes (FF0000–FFFFFFh) on the PCI, if this region is within main memory or within the main memory hole and enabled via the EADC2 Register, are always forwarded in a non-buffered mode, unless overlapped with a programmable region that defines buffered access mode.

4.2.2 PROGRAMMABLE EISA-TO-PCI MEMORY ADDRESS REGIONS

The PCEB supports four programmable memory regions for EISA-to-PCI transfers. The PCEB positively decodes EISA memory accesses to these regions and forwards the cycle to the PCI Bus. This feature permits EISA master accesses to PCI devices that reside within these address ranges.

Regions can be enabled or disabled. After reset, all regions are disabled. Each region has an associated Base and Limit Address fields MEMREGN[4:1] that determine the size and location of each region. These registers are programmed with the starting address of the region (Base) and ending address of the region (Limit). The address range for a particular region is defined by the following equation:

$$\text{Base_Address} \leq \text{Address} \leq \text{Limit_Address}$$

These regions can be defined anywhere in the 4 GByte address space at 64 KByte boundaries and with 64 KByte granularity. In practical applications, the regions will be mapped within the main memory hole or above the top of the memory defined by the MEMCS# map.

Access to the memory locations within a region can be performed in one of two modes:

- **Non-Buffered Mode**: PCEB's EISA-to-PCI Line Buffers can be disabled for all EISA-to-PCI memory read/write accesses through the PCICON Register or for selected accesses through EPMRA Register.
- **Buffered Mode**: Line Buffers enabled. Read-prefetch and write-assembly/posting allowed (without strong ordering).

Since buffered mode provides maximum performance (and concurrency in non-GAT mode), it should be selected, unless the particular region is used for memory-mapped I/O devices. I/O devices can not be accessed in read-prefetch or write-assembly/posted fashion because of potential side-effects (see Section 6.0, Data Buffering).

4.2.3 PROGRAMMABLE EISA-TO-PCI I/O ADDRESS REGIONS

The PCEB provides four programmable I/O address regions. These regions are defined by Base and Limit addresses fields contained in the associated IOREGN[4:1] Registers. These regions can be defined anywhere within the 64 KByte I/O space on Dword boundaries (and with Dword granularity). See Section 4.1, PCEB Configuration Registers.

4.2.4 EXTERNAL EISA-TO-PCI I/O ADDRESS DECODER

Since the I/O address map may be highly fragmented, it is impractical to provide enough programmable regions to completely define mapping of registers for I/O devices on the PCI. The PCEB's input signal pin PIODEC# can be used, if a more complex I/O decode scheme is needed. PIODEC# complements the functions of the four PCEB programmable I/O regions with external decode logic. If PIODEC# is asserted during an EISA I/O cycle, the cycle is forwarded to the PCI Bus.

If the PIODEC# signal is not used, a pull-up resistor is required to provide an inactive signal level.

4.3 Palette DAC Snoop Mechanism

Some advanced graphics EISA/ISA expansion boards use the pre-DAC VGA pixel data from the VGA Special Feature Connector and merge it with advanced graphics data (multi-media for example). The merged data is then run through a replicated palette DAC on the advanced graphics expansion board to create the video monitor signal. The replicated palette DAC is kept coherent by snooping VGA palette DAC writes. Snooping becomes an issue in a system where the VGA controller is placed on the PCI Bus and the snooping graphics board is on the EISA expansion bus. Normally, the PCI VGA controller will respond to the palette DAC writes with DEVSEL#, so the PCEB will not propagate the cycle to the EISA Bus using subtractive decoding.

The burden for solving this problem is placed on the VGA subsystem residing on the PCI. The VGA subsystem on PCI must have an enable/disable bit associated with palette DAC accesses. When this bit is enabled the PCI VGA device responds in handshake fashion (generates DEVSEL#, TRDY#, etc.) to I/O reads and writes to the palette DAC space.

When this bit is disabled, the PCI VGA device responds in handshake fashion only to I/O reads to palette DAC space. I/O writes to the palette DAC space will be snooped (data latched) by the PCI VGA device, but the PCI VGA subsystem will not generate a DEVSEL#. In this case, the I/O write will be forwarded to the EISA Bus by the PCEB as a result of subtractive decode. The PCI VGA device must be able to snoop these cycles in the minimum EISA cycle time.

The state of palette-DAC snooping control bit does not affect I/O reads from the palette DAC space. Regardless of whether this bit is enabled or disabled, the PCI VGA device will service the I/O reads from the palette DAC space.

5.0 PCI INTERFACE

The PCEB provides the PCI Interface for the PCI-EISA Bridge. The PCEB can be an initiator (master) or target (slave) on the PCI Bus and supports the basic PCI Bus commands as described in Section 5.1.1, PCI Command Set. For EISA-to-PCI transfers, the PCEB is a master on the PCI Bus on behalf of the requesting EISA device. An EISA device can read and write either PCI memory or I/O space.

The PCEB forwards unclaimed PCI Bus cycles to EISA. For PCI Bus cycles that are not claimed, the PCEB becomes a slave on the PCI Bus (claiming the cycle via subtractive or negative decoding) and forwards the cycle to the EISA Bus. Note that negative decoding is only used on the 82374SB.

This section describes the PCI Bus transactions supported by the PCEB. The section also covers the PCI Bus latency mechanisms in the PCEB that limit a master's time on the bus and the PCEB support of parity. In addition, the PCEB contains PCI Bus arbitration circuitry that supports up to six masters. PCI Bus arbitration is described in Section 5.4.

NOTE:
1. All signals are sampled on the rising edge of the PCI clock. Each signal has a setup and hold window with respect to the rising clock edge, in which transitions are not allowed. Outside of this range, signal values or transitions have no significance.
2. The terms initiator and master are synonymous. Likewise, the terms target and slave are synonymous.
3. Readers should be familiar with the PCI Bus specification.

5.1 PCI Bus Transactions

This section presents the PCI Bus transactions supported by the PCEB.

5.1.1 PCI COMMAND SET

PCI Bus commands indicate to the target the type of transaction requested by the master. These commands are encoded on the C/BE[3:0]# lines during the address phase of a transfer. Table 5 summarizes the PCEB's support of the PCI Bus commands.

Table 5. PCEB-Supported PCI Bus Commands

C/BE[3:0]#	Command Type	Supported As Target	Supported As Initiator
0000	Interrupt Acknowledge	Yes	No
0001	Special Cycle	No	No
0010	I/O Read	Yes	Yes
0011	I/O Write	Yes	Yes
0100	Reserved	N/A(3)	N/A(3)
0101	Reserved	N/A(3)	N/A(3)
0110	Memory Read	Yes	Yes
0111	Memory Write	Yes	Yes
1000	Reserved	N/A(3)	N/A(3)
1001	Reserved	N/A(3)	N/A(3)
1010	Configuration Read	Yes	No
1011	Configuration Write	Yes	No
1100	Memory Read Multiple	No(2)	No
1101	Reserved	N/A(3)	N/A(3)
1110	Memory Read Line	No(2)	No
1111	Memory Write and Invalidate	No(1)	No

NOTES:
1. As a target, the PCEB treats this command as a memory write command.
2. As a target, the PCEB treats this command as a a memory read command.
3. The PCEB considers a reserved command invalid and, as a target, completely ignores the transaction. All internal address decoding is ignored and the PCEB never asserts DEVSEL#. As a PCI master, the PCEB never generates a bus cycle with a reserved command type.

5.1.2 PCI CYCLE DESCRIPTIONS

Each PCI Command is listed below with the following format of information:

Command Type
  PCEB target support
    -Decode method
    -Data path
    -PCEB response
    -Result of no response on EISA
  PCEB initiator support
    -Data path
    -Conditions for generating command
    -Result of no response on PCI

5.1.2.1 Interrupt Acknowledge

Target support:
Decode: Positive
Data Path: Flow through
Response:
The interrupt acknowledge cycle is subject to retry. If the PCEB is locked, or if the interrupt acknowledge cycle triggers buffer management activity, or if the EISA Bus is occupied by an EISA/ISA master or the DMA, the interrupt acknowledge cycle is retried.

The interrupt acknowledge command is a single byte read that is implicitly addressed to the interrupt controller in the ESC component. The address bits are logical ''don't cares'' during the address phase and the byte enables indicate to the PCEB that an 8-bit interrupt vector is to be returned on byte 0. After performing the necessary buffer management operations and obtaining ownership of the EISA Bus, the PCEB generates a single pulse on the PEREQ#/INTA# inter-chip signal and performs an I/O read cycle (on the EISA Bus) to the ESC internal registers residing at I/O address 04h. The ESC decode logic uses the PEREQ#/INTA# signal to distinguish between standard accesses to I/O address 04h (DMA controller) and special accesses that result in a vector being read by the PCEB. The PCEB holds the PCI Bus, in wait states, until the interrupt vector is returned. PEREQ#/INTA# remains asserted until the end of the read cycle.

Result of no response on EISA:
The PCEB runs a standard length EISA I/O read cycle and terminates normally. The value of the data returned as an interrupt vector is meaningless.

Initiator support: None.

NOTE:
The PCEB only responds to PCI interrupt acknowledge cycles if this operation is enabled via bit 5 of the PCICON Register).

5.1.2.2 Special Cycle

Target support: None.

Initiator support: None.

5.1.2.3 I/O Read

Target support:
Decode: Positive (PCEB and some ESC registers) & Subtractive
Data Path: Flow through
Response:
The PCEB claims I/O read cycles via positive or subtractive decoding and generates DEVSEL#. The internal PCEB registers (BIOS Timer) and the IDE and the 8259 registers are positively decoded. Any unclaimed cycle below 64 KByte is subtractively decoded and forwarded to the EISA Bus. The I/O read cycle is subject to retry. If the PCEB is locked, if the cycle triggers buffer management activity, or if the EISA Bus is occupied by an EISA/ISA master or the DMA, the I/O read cycle is retried. If the cycle gets retried due to an occupied EISA Bus, the EISA Bus is requested.

Once an I/O read cycle is accepted (not retried) by the PCEB, the PCI Bus is held in wait states using TRDY# until the cycle is completed internally or on the EISA Bus.

Burst I/O reads to the EISA Bus or to the PCEB are not supported. Therefore, any burst I/O read cycles decoded by the PCEB are target terminated after the first data transaction using the disconnect semantics of the STOP# signal (Disconnect A).

Result of no response on EISA: The PCEB runs a standard length EISA I/O cycle and terminates normally.

Initiator support:
The PCEB generates PCI Bus I/O read cycles on behalf of an EISA master. EISA cycles are forwarded to the PCI Bus if the I/O address is within one of four programmable I/O address regions as defined in Section 4.0 Address Decoding.

Result of no response on PCI: Master abort due to DEVSEL# time-out. PCEB returns data value FFFFFFFFh.

5.1.2.4 I/O Write

Target support:
Decode: Positive (PCEB/ESC registers) & Subtractive
Data Path: Flow through
Response:
I/O write cycles can be claimed by the PCEB via positive or subtractive decoding. In either case, the PCEB generates DEVSEL#. The internal PCEB registers (BIOS Timer), IDE registers and 8259 registers are positively decoded, if enabled. Any unclaimed cycle below 64 KByte is subtractively decoded and forwarded to the EISA Bus. The I/O write cycle is subject to retry. If the PCEB is locked, if the cycle triggers buffer management activity, or if the EISA Bus is occupied by an EISA/ISA master or the DMA, the I/O write cycle is retried. If the cycle is retried due to an occupied EISA Bus, the EISA Bus is requested.

Once an I/O write cycle is accepted (not retried) by the PCEB, the PCI Bus is held in wait states using TRDY# until the cycle is completed within the PCEB or on the EISA Bus.

Burst I/O writes to the EISA Bus or to the PCEB are not supported. Therefore, any burst I/O write cycles decoded by the PCEB are target terminated after the first data transaction using the disconnect semantics of the STOP# signal (Figure 5-12, Disconnect A).

Result of no response on EISA:
The PCEB runs a standard length EISA I/O cycle and terminates normally.

Initiator support:
The PCEB generates PCI I/O write cycles on behalf of an EISA master. EISA cycles are forwarded to the PCI Bus if the I/O address is within one of the four programmable I/O address regions defined in Section 4.0, Address Decoding.

Result of no response on PCI:
Master abort due to DEVSEL# time-out.

5.1.2.5 Memory Read

Target support:
Decode: Negative (82374SB only) and Subtractive
Data Path: Flow through
PCEB Response:
Memory read cycles may be claimed by the PCEB via negative or subtractive decoding. The PCEB claims the cycle by asserting DEVSEL#. Unclaimed PCI cycles (DEVSEL# time-out) are claimed by the PCEB via subtractively decoding and forwarded to the EISA Bus. The memory read cycle is subject to retry. If the PCEB is locked, if the cycle triggers buffer management activity, or if the EISA Bus is occupied by an EISA/ISA master or the DMA, the memory read cycle is retried. If the cycle is retried due to an occupied EISA Bus, the EISA Bus is requested.

Once a memory read cycle is accepted (not retried) by the PCEB, the PCI Bus is held in wait states, using TRDY#, until the cycle is completed to the EISA Bus.

Incremental burst memory reads destined for the EISA Bus take longer than the allowed 8 PCICLKs. Therefore, any burst memory read cycle decoded by the PCEB causes the PCEB to target terminate the cycle after the first data transaction using the disconnect semantics of the STOP# signal (Figure 5-8, Disconnect A).

Result of no response on EISA:
The PCEB runs a standard length EISA memory read cycle and terminates normally.

Initiator support:
Data Path: Line Buffer when enabled. Flow through when Line Buffer is disabled or it is a bypass cycle.

Cycle Generation Conditions:
As an initiator, the PCEB generates a PCI memory read cycle when it decodes an EISA memory read cycle destined to the PCI that can not be serviced by the Line Buffer. This condition occurs for EISA/ISA master and DMA cycles that can not be serviced by the Line Buffer because the Line Buffer is empty, there is a Line Buffer miss, or Line Buffering is disabled.

As an initiator, the PCEB only generates linear incrementing burst ordering that is signaled by AD[1:0] = 00 during the address phase. Other types of burst transfers (i.e. cache line toggle mode) are never initiated by the PCEB.

The PCEB generates a burst memory read when it is fetching 16 bytes into one of the four Line Buffers.

Result of no response on PCI:
Master abort due to DEVSEL# time-out. PCEB returns data value FFFFFFFFh.

5.1.2.6 Memory Write

Target support:
Decode: Negative (82374SB only) and Subtractive
Data Path: Flow through
PCEB Response:
Memory write cycles may be claimed by the PCEB via negative or subtractive decoding. The PCEB asserts DEVSEL# to claim the cycle. Unclaimed PCI cycles (DEVSEL# time-out) within the 4 GByte memory space are claimed by the PCEB via subtractively decoding and forwarded to the EISA Bus. The memory write cycle is subject to retry. If the PCEB is locked, the cycle triggers buffer management activity. If the EISA Bus is occupied by an EISA/ISA master or the DMA, the memory write cycle is retried. If the cycle is retried due to a disabled buffer because the EISA Bus is occupied, the EISA Bus is requested.

Once a memory write cycle is accepted (not retried) by the PCEB, the PCEB holds the PCI Bus in wait states (using TRDY#) until the cycle is completed on the EISA Bus.

Result of no response on EISA: The PCEB initiates a standard length EISA memory write cycle and terminates normally.

Initiator support:
Data Path: Line Buffer when enabled, flow through when Line Buffer is disabled.
Cycle Generation Conditions:
As an initiator, the PCEB generates a PCI memory write cycle when it decodes an EISA memory write cycle destined to PCI, that can not be serviced by the Line Buffer because it is disabled. This occurs for EISA/ISA masters and DMA cycles when the Line Buffer is disabled. The PCEB also generates a memory write cycle when the Line Buffer needs to be flushed. The Line Buffer is flushed under several conditions, including when the 16 byte line is full, when there is a ''miss'' to the current 16 byte line, or when it is required by the buffer management logic. (See Section 6.0, Data Buffering).

As an initiator, the PCEB generates only linear incrementing burst ordering that is signaled by AD[1:0] = ''00'' during address phase. Other types of burst transfers (i.e. cache line toggle mode) are never initiated by the PCEB.

Result of no response on PCI: Master abort due to DEVSEL# time-out.

5.1.2.7 Configuration Read, Configuration Write

Target support:
Decode: via IDSEL pin
Data Path: Flow through
Response:
The PCEB responds to configuration cycles by generating DEVSEL# when its IDSEL signal is asserted, regardless of the address. During configuration cycles, AD[7:2] are used to address the PCEB's configuration space. AD[31:8] are not used and are logical ''don't cares''. AD[1:0] must be zero.

Result of no response on EISA: N/A

Initiator support:
Configuration cycles are never generated by the PCEB.

5.1.2.8 Memory Read Multiple

Target support:
The PCEB aliases this command to a normal memory read cycle. See the Memory Read command description.

Initiator support:
Memory read multiple cycles are never generated by the PCEB.

5.1.2.9 Memory Read Line

Target support:
The PCEB aliases this command to a normal memory read. See the Memory Read command description.

Initiator support:
Memory read line cycles are never generated by the PCEB.

5.1.2.10 Memory Write And Invalidate

Target support:
Response:
The PCEB treats this command like a memory write. See the Memory Write command description.

Initiator support:
Cycle Generation Conditions:
The PCEB does not generate this command cycle.

5.1.3 PCI TRANSFER BASICS

The basic bus transfer mechanism on the PCI Bus is a burst. A burst is comprised of an address phase and one or more data phases. The PCI protocol specifies the following types of burst ordering (signaled via A[1:0] during the address phase):

A[1:0]	Burst Order
0 0	Linear Incrementing
0 1	Cache line toggle mode
1 X	Reserved

The PCEB only supports linear incrementing burst ordering as an initiator. Data transfers for ordering other than linear incrementing are disconnected by the PCEB (burst split into multiple single data transfers).

The fundamentals of all PCI data transfers are controlled with the following three signals:

- FRAME# is driven by the PCI master to indicate the beginning and end of a transaction.
- IRDY# is driven by the PCI master, allowing it to force wait states.
- TRDY# is driven by the PCI target, allowing it to force wait states.

The PCI Bus is idle when both FRAME# and IRDY# are negated. The first clock edge that FRAME# is sampled asserted is the address phase, and the address and bus command code are transferred on that clock edge. The next clock edge begins the first of one or more data phases. During the data phases, data is transferred between master and slave on each clock edge that both IRDY# and TRDY# are sampled asserted. Wait states may be inserted by either the master (by negating IRDY#) or the target (by negating TRDY#). When a PCI master has one more data transfer to complete the cycle (which could be immediately after the address phase), it negates FRAME#. IRDY# must be asserted at this time, indicating that the master is ready for the final data transfer. After the target indicates the final data transfer (TRDY# asserted), the master negates IRDY#, causing the target's PCI interface to return to the idle state (FRAME# and IRDY# negated), on the next clock edge.

For I/O cycles, PCI addressing is on byte boundaries and all 32 AD lines are decoded to provide the byte address. For memory cycles, AD[1:0] are used to define the type of burst ordering. For configuration cycles, DEVSEL# is strictly a function of IDSEL#. Configuration registers are selected as Dwords using AD[7:2]. The AD[1:0] must be 00 for the target to directly respond to the configuration cycle. The byte enables determine which byte lanes contain valid data.

Each PCI agent is responsible for its own positive address decode. Only one agent (the PCEB) on the PCI Bus may use subtractive decoding. The little endian addressing model is used.

The byte enables are used to determine which bytes carry meaningful data. These signals are permitted to change between data phases. The byte enables must be driven valid from the edge of the clock that starts each data phase and must stay valid for the entire data phase. Figure 7, the data phases begin on clocks 3 and 4. (Changing byte enables during a read burst transaction is generally not useful, but is supported on the bus.) The master is permitted to change the byte enables on each new data phase, although the read diagram does not show this. The timing for changing byte enables is the same for read and write transactions. If byte enables are important for the target on a read transaction, the target must wait for the byte enables to be driven on each data phase before completing the transfer.

5.1.3.1 Turn-Around-Cycle Definition

A turn-around-cycle is required on all signals that may be driven by more than one agent. The turn-around-cycle is required to avoid contention when one agent stops driving a signal and another agent begins, and must last at least one clock. The symbol that represents a turn-around-cycle in the timing relationship figures is a circular set of two lines, each with an arrow that points to the other's tail. This turn-around-cycle occurs at different times for different signals. For example, the turn-around-cycle for IRDY#, TRDY# and DEVSEL# occurs during the address phase and for FRAME#, C/BE# and AD, it occurs during the idle cycle.

5.1.3.2 Idle Cycle Definition

The cycle between clocks 7 and 8 in Figure 8 is called an idle cycle. Idle cycles appear on the PCI Bus between the end of one transaction and the beginning of the next. An idle cycle occurs when both FRAME# and IRDY# are negated.

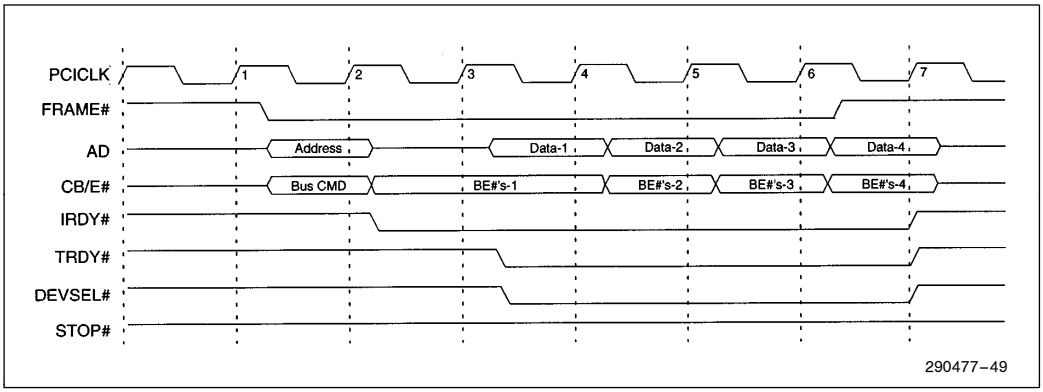


Figure 7. PCEB Burst Read from PCI Memory

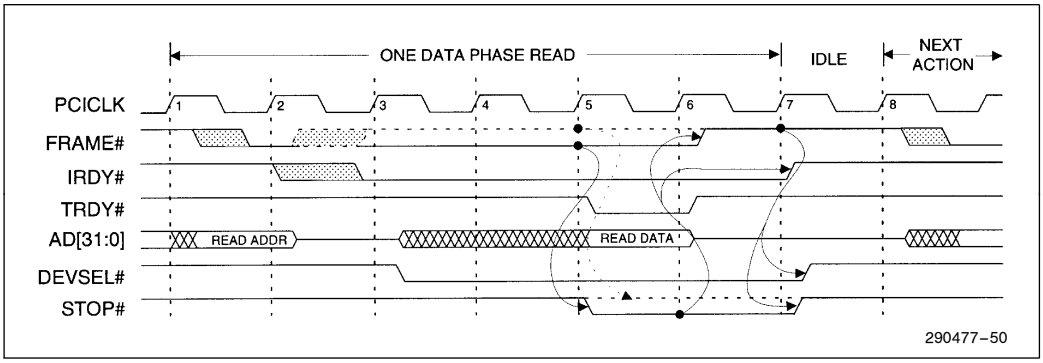


Figure 8. PCI Master Read from the PCEB (Burst with Target Termination)

5.1.4 BASIC READ

As a PCI master, the PCEB performs memory and I/O read transfers. Figure 7 shows a PCEB zero wait state burst read from PCI memory (PCEB is a master). If buffering of memory accesses is enabled, read transfers use prefetching. When reading data from PCI memory, the PCEB requests a minimum of 16 bytes (one data line of the Line Buffer), via a four data phase burst read cycle, to fill one of its internal Line Buffers. The PCEB does not buffer PCI I/O reads and only required data is transferred during these cycles. Read Cycles to PCI are generated on behalf of EISA/ISA masters and DMA devices.

The PCEB asserts FRAME# on clock 1 and places the address on AD[31:2]. CB/E[3:0]# contain a valid bus command. AD[1:0] contain the byte address for I/O cycles, burst order indication for memory cycles, and are 00 for configuration cycles.

The clock following the address phase is the beginning of the data phase. During the data phase, C/BE[3:0]# indicate which byte lanes are involved in the transaction. If the byte lanes involved in the transaction are different for data 1 and data 2, the PCEB drives new C/BE[3:0] values on clock 4. C/BE[3:0]# remain active until the end of the burst transfer.

The first data phase of a read transaction requires a turn-around-cycle, which is enforced by the target preventing the assertion of TRDY# until at least clock 3. The PCEB stops driving the address at clock 2. The target can not drive the AD bus until clock 3. This allows enough time for the PCEB to float its AD outputs. The target is required to drive the AD lines as soon as possible after clock 3, even though valid data may not be ready and the target may want to stretch the initial data phase by delaying TRDY#. This insures that the AD lines are not left floating for long intervals. The target must continue to drive these lines until the end of the burst transaction.

A single data phase is completed when the initiator of the cycle samples TRDY# asserted on the same clock that IRDY# is asserted. To add wait states, the target must negate TRDY# for one or more clock cycles. As a master, the PCEB does not add wait states. In Figure 7, data is transferred on clocks 4 and 5. The PCEB knows, at clock 6, that the next data phase is the last and negates FRAME#. As noted before, the PCEB can burst a maximum of four data cycles when reading from PCI memory.

As a PCI target, the PCEB responds to both I/O and memory read transfers. Figure 8 shows the PCEB, as a target, responding to a PCI master read cycle. For multiple read transactions, the PCEB always target terminates after the first data read transaction by asserting STOP# and TRDY#. These signals are asserted at the end of the first data phase. For single read transactions, the PCEB completes the cycle in a normal fashion (by asserting TRDY# without asserting STOP#). Figure 8 shows the fastest PCEB response to an access of an internal configuration register. During EISA Bus read accesses, the PCEB always adds wait states by negating TRDY# until the transfer on the EISA Bus is completed.

When the PCEB, as a target, samples FRAME# active during a read cycle and positively decodes the cycle, it asserts DEVSEL# on the following clock (clock 3 in Figure 8). Note that, if the PCEB subtractively or negatively (82374SB only) decodes the cycle, DEVSEL# is not asserted for two to three PCICLK's after FRAME# is sampled active. (see Section 5.1.9, Device Selection). When the PCEB asserts DEVSEL#, it also drives AD[31:0], even though valid data is not available. TRDY# is also driven from the same clock edge but it is not asserted until the PCEB is ready to drive valid data. TRDY# is asserted on the same clock edge that the PCEB drives valid data on AD[31:0]. If the PCEB presents valid read data during the first data phase and FRAME# remains active (multiple transaction indicated), the PCEB asserts TRDY# and STOP# to indicate target termination of the transfer.

5.1.5 BASIC WRITE

Figure 9 shows the PCEB, as a master, writing to PCI memory in zero wait states. Figure 10 shows the fastest response of the PCEB, as a target, to a memory or I/O write transaction generated by a PCI master.

As a PCI master, the PCEB performs memory write and I/O transfers. If buffering of memory accesses is enabled, write transfers are posted. When writing data to PCI memory, the PCEB writes a maximum of 16 bytes (one line of the Line Buffer) using a burst write cycle. I/O writes are always non-buffered transactions.

The PCEB generates PCI write cycles on behalf of EISA masters and DMA devices, and when the PCEB flushes its internal Line Buffer.

As a PCI target, the PCEB responds to both I/O and memory write transfers. If the EISA Bus is occupied, the PCI write is retried by the PCEB. When the PCEB owns the EISA Bus, the transaction proceeds. For burst I/O writes, the PCEB always target terminates after the first data transaction by asserting STOP# and TRDY# at the end of the first data phase. During a burst memory write, the PCEB always target terminates after the first data phase.

Figure 10 shows the fastest PCEB response to a write cycle targeted to an internal PCI configuration register. During I/O or memory write accesses to the EISA Bus, the PCEB always adds wait states. The PCEB adds wait states by holding TRDY# high until the transfer on the EISA Bus is completed.

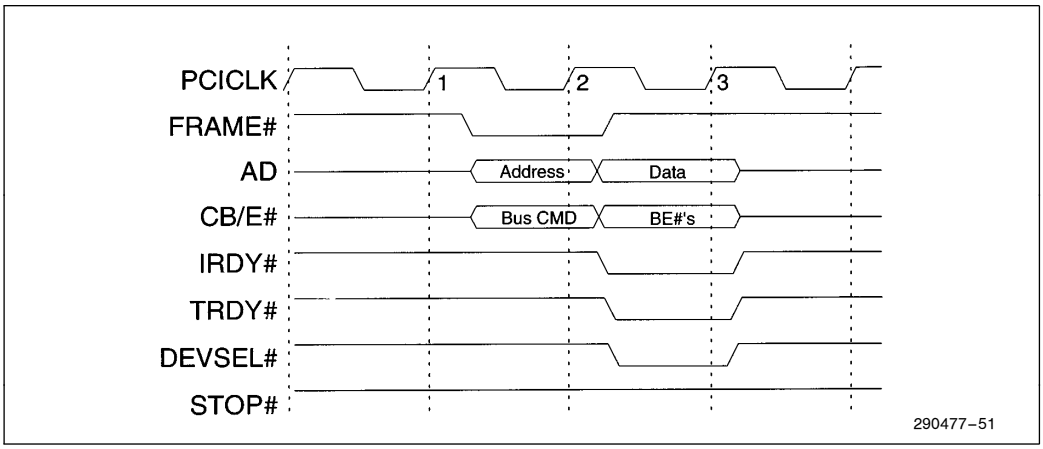


Figure 9. PCEB Write to PCI Memory

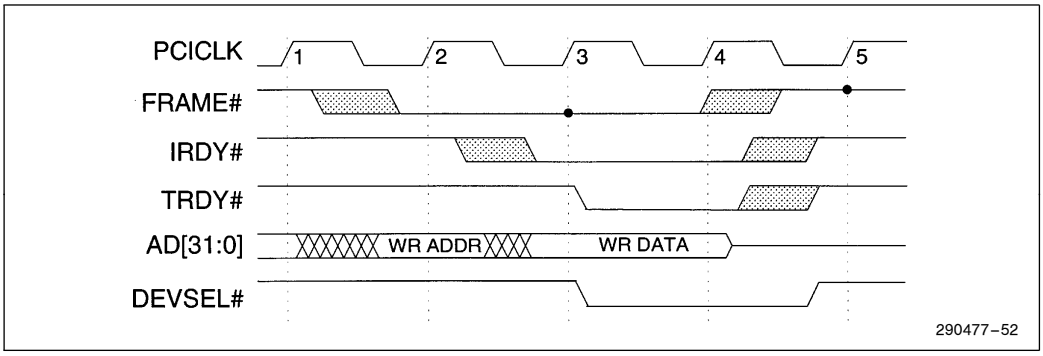


Figure 10. Fastest PCI Write to PCEB

5.1.6 CONFIGURATION CYCLES

One of the requirements of the PCI specification is that upon power up, PCI agents do not respond to any address. The only access allowed is through the IDSEL configuration mechanism. The PCEB is an exception to this since it controls access to the BIOS boot code. All PCEB/ESC subsystem addresses that are enabled after reset are accessible immediately after power up.

The configuration read or write command is used to configure the PCEB. During the address phase of the configuration read or write cycle, the PCEB samples its IDSEL (ID select) signal (not the address lines) to generate DEVSEL#. In this way, IDSEL acts as a chip select. During the address phase, AD[7:2] are used to select a particular configuration register and BE[3:0] to select a particular byte(s). The PCEB only responds to configuration cycles if AD[1:0] = 00. Reference Figure 11 for configuration reads and writes. Note that IDSEL is normally a ''don't care'', except during the address phase of a transaction. Upon decode of a configuration cycle and sampling IDSEL active, the PCEB responds by asserting DEVSEL# and TRDY#. An unclaimed configuration cycle is never forwarded to the EISA Bus.

Configuration cycles are not normally run in burst mode. If this happens, the PCEB splits the transfer into single cycles using the slave termination mechanism.

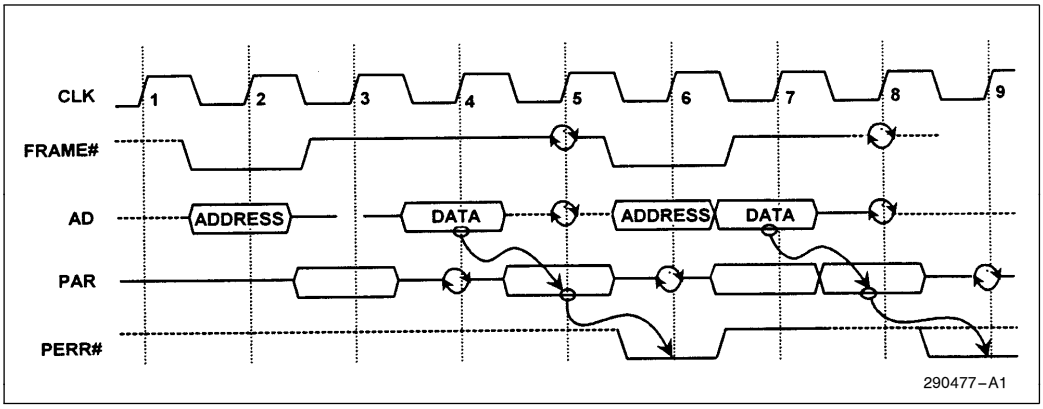


Figure 11. Configuration Cycle

5.1.7 INTERRUPT ACKNOWLEDGE CYCLE

The PCEB responds to an interrupt acknowledge cycle as decoded from the command during a valid address cycle (FRAME# asserted). The AD bus itself is a ''don't care'' to the PCEB during the address phase and, therefore, status of the internal PCI address decoder is not used for forwarding the cycle to the EISA Bus where the system interrupt controller resides.

The PCEB converts the PCI interrupt acknowledge cycle into an EISA I/O read access to the address 04h, with special semantics indicated to the ESC via the inter-chip signaling. Before the PCI interrupt acknowledge cycle can be converted into an EISA I/O read cycle, the EISA Bus must be owned. If the EISA Bus is not owned by the PCEB (EISAHLDA asserted), the PEREQ#/INTA# signal is asserted with PEREQ# semantics (PCI-to-EISA request). After the EISA Bus is acquired by the PCEB, the interrupt acknowledge sequence can proceed. The PCEB starts an I/O read cycle to address 04h and asserts PEREQ#/INTA# with INTA# semantics. The PEREQ#/INTA# remains asserted for the duration of the EISA I/O read cycle. Therefore, only a single pulse is generated on the PEREQ#/INTA# signal. Conversion of the single PCI interrupt acknowledge cycle into two interrupt acknowledge pulses (that is required for 8259 compatibility) occurs inside the ESC where the 8259-based interrupt controller resides. The ESC's EISA decoder uses the PEREQ#/INTA# signal (with INTA# semantics) to distinguish between normal I/O reads to the register located at address 04h (DMA1 Ch2 Base and Current Address) and the interrupt acknowledge sequence. The ESC holds the EISA Bus in wait states until the interrupt vector is returned to the PCEB (via SD[7:0]). The PCEB passes the vector to the PCI via AD[7:0] and then terminates the cycles both on EISA and PCI. Note that for compatibility reasons, only the ESC (containing the DMA controller) can respond to the EISA I/O read from 04h.

Figure 12 shows the PCI portion of the interrupt acknowledge sequence. The EISA portion of the sequence matches normal EISA I/O read timing, except that the PEREQ#/INTA# inter-chip signal is asserted during the bus cycle with INTA# semantics and, during the PCEB/ESC EISA Bus ownership exchange handshake, with PEREQ# semantics. Note that the PCEB responses to a PCI interrupt acknowledge cycle can be disabled by setting bit 5 in the PCI Control Register (PCICON) to 0.

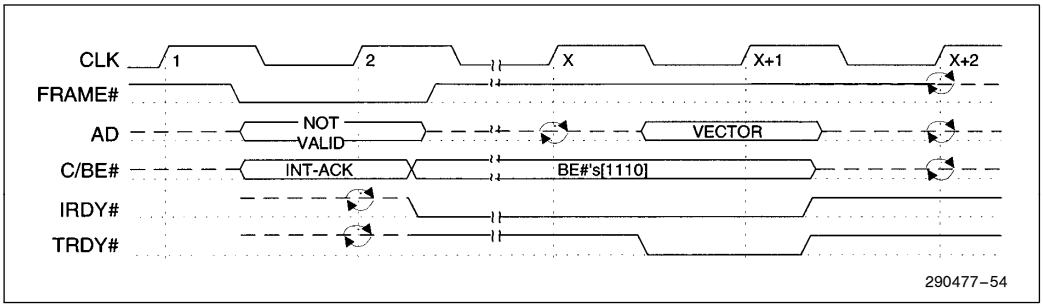


Figure 12. PCI Interrupt Acknowledge Cycle

5.1.8 EXCLUSIVE ACCESS

Refer to Figure 13, Figure 14, and Figure 15 for exclusive access timing relationships.

Target support:
PCI provides an exclusive access mechanism that allows non-exclusive accesses to proceed in the face of exclusive accesses. This is referred to as a Resource Lock. (Note that the exclusive access mechanism that locks the entire bus is Bus Lock.) The PCEB, as a resource, can be locked by any PCI initiator. In the context of locked cycles, the PCEB and entire EISA subsystem are considered a single resource. (EISA subsystem is indirectly locked during an exclusive access to the PCEB.) A locked access to any address contained within the EISA subsystem locks the entire subsystem from the PCI side. The PLOCK# signal is propagated to the EISA LOCK# signal. Note that write posting (PCI-to-EISA) is disabled for PCI locked cycles propagated to the EISA subsystem. The EISA Bus is not released to ESC until the locked sequence is complete. A subsequent PCI initiator access to the EISA subsystem, while it is locked, results in a retry. The PCEB becomes locked when it is the target of the access and PLOCK# is sampled negated during the address phase. The PCEB remains locked until FRAME# and PLOCK# are both sampled negated. When in a locked state, the PCEB only accepts requests when PLOCK# is sampled negated during the address phase. If PLOCK# is asserted during the address phase, the PCEB responds by asserting STOP# with TRDY# negated (RETRY).

As an unlocked target, the PCEB ignores PLOCK# when deciding if it should respond to a PCI address decoder hit. Also, if PLOCK# is sampled asserted during an address phase, the PCEB does not go into a locked state.

As a locked target, the PCEB responds to an initiator when it samples PLOCK# negated during the address phase of the cycle in which the PCEB is the target of the access. The locking master may negate PLOCK# at the end of the last data phase. When FRAME# and PLOCK# are both sampled negated, the PCEB goes to the unlocked state.

Note that the PCEB does not release the EISA Bus when it is in the locked state.

Initiator support:
When an EISA locked access to the PCI is encountered (EISA LOCK# asserted), the cycle is propagated to the PCI Bus as a PCI locked cycle. Line Buffers in the PCEB are bypassed. The PLOCK# signal must be negated (released) before an EISA agent can be granted the EISA Bus. Thus, when the PCEB acquires the PCI Bus on behalf of the EISA agent, a PCI LOCKED cycle can be performed, if needed.

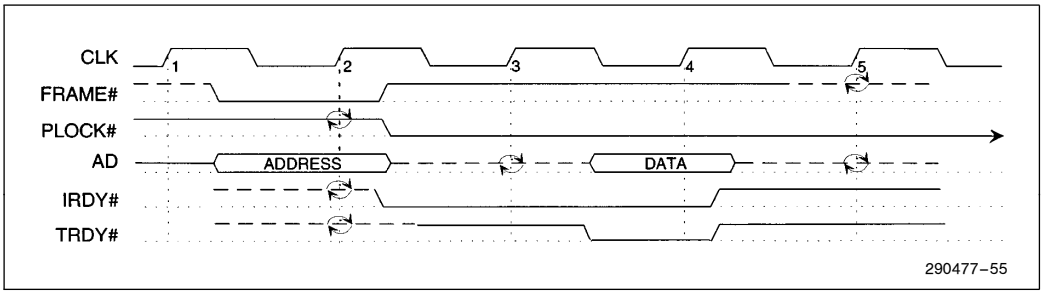


Figure 13. Beginning a Locked Cycle

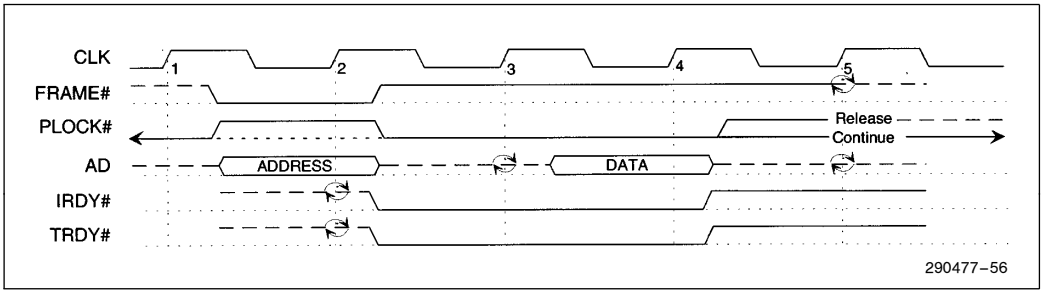


Figure 14. Continuing Locked Cycle

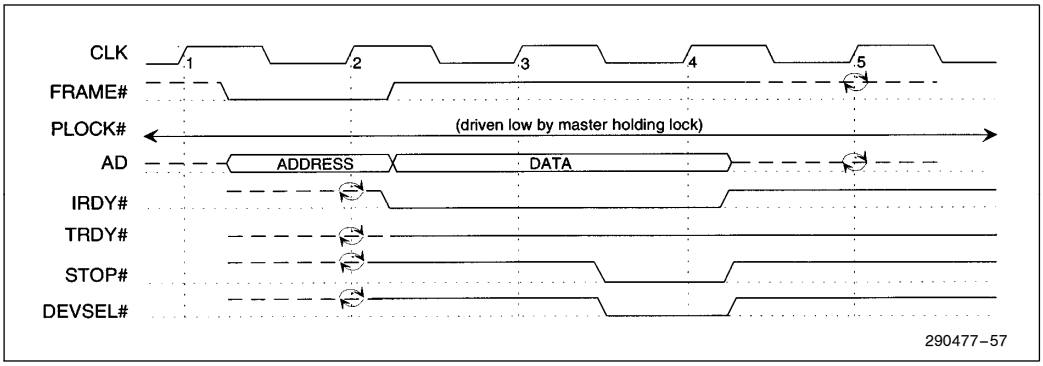


Figure 15. Access to Locked Target with PLOCK# Asserted During Address Phase

5.1.9 DEVICE SELECTION

The PCEB asserts DEVSEL# to indicate that it is the target of the PCI transaction. DEVSEL# is asserted when the PCEB, as a target, positively, subtractively, or negatively (82374SB only) decodes the PCI transaction. In all cases except one, once the PCEB asserts DEVSEL#, the signal remains asserted until FRAME# is negated (IRDY# is asserted) and either STOP# or TRDY# is asserted. The exception is a target abort, described in Section 5.1.10, Transaction Termination.

For most systems, PCI target devices are able to complete a decode and assert DEVSEL# within 2 or 3 clocks of FRAME# (medium and slow in the Figure 16). Accordingly, since the PCEB subtractively or negatively (82374SB only) decodes all unclaimed PCI cycles (except configuration cycles), it provides a configuration option to reduce by 1 clock the edge at which it samples DEVSEL#, allowing faster access to the expansion bus. Use of this option is limited by the slowest positive decode agent on the bus. This is described in more detail in Section 4.0, Address Decoding.

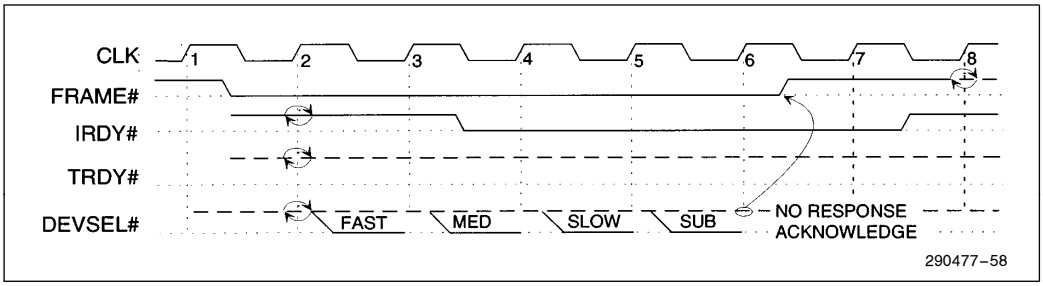


Figure 16. Device Selection (DEVSEL#)

5.1.10 TRANSACTION TERMINATION

Termination of a PCI cycle can be initiated by either a master or a target. The PCEB supports both master and target initiated termination. All transactions are concluded when FRAME# and IRDY# are both sampled negated, indicating that the PCI Bus is idle.

5.1.10.1 Master Initiated Termination

The PCEB supports three types of master initiated termination:

- **Completion:** Refers to the termination when the PCEB finishes the transaction normally. This is the most common type of termination.
- **Time-out:** Refers to termination when the PCEB's GNT# line is negated and its internal Master Latency Timer has expired. The intended transaction is not necessarily concluded. The timer may have expired because of a target-induced access latency, or because the intended operation was very long.
- **Abort:** Refers to termination when there is no target response (no DEVSEL# asserted) to a transaction within the programmed DEVSEL# response time.

Completions and time-outs are common while the abort is an abnormal termination. A normal termination of this type can be seen in Section 5.1.4 and 5.1.5 in the descriptions of the basic PCI read and write transaction.

The PCEB sends out a master abort (Figure 17) when the target does not respond to the PCEB-initiated transaction by asserting DEVSEL#. The PCEB checks DEVSEL# based on the programmed DEVSEL# sample point. If DEVSEL# is not asserted by the programmed sample point, the PCEB aborts the transaction by negating FRAME#, and then, one clock later, negating IRDY#. The master abort condition is abnormal and it indicates an error condition. The PCEB does not retry the cycle.

If the transaction is an EISA-to-PCI memory or I/O write, the PCEB terminates the EISA cycle with EXRDY. If the transaction is an EISA-to-PCI memory or I/O read, the PCEB returns FFFFFFFFh on the EISA Bus. This is identical to the way an unclaimed cycle is handled on the ''normally ready'' EISA Bus. If the Line Buffer is the requester of the PCI transaction, the master abort mechanism ends the PCI cycle, but no data is transferred into or out of the Line Buffer. The Line Buffer does not retry the cycle. The Received Master Abort Status bit in the PCI Status Register is set to 1 indicating that the PCEB issued a master abort.

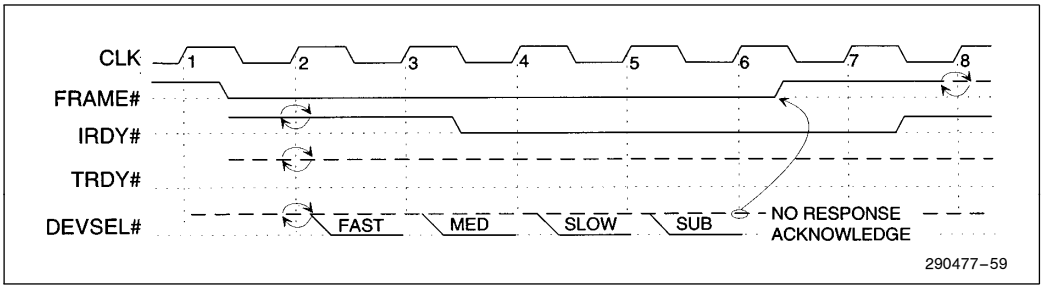


Figure 17. Master Initiated Termination (Master Abort)

5.1.10.2 Target Initiated Termination

The PCEB supports two forms of target-initiated termination:

- **Disconnect:** A disconnect termination occurs when the target is unable to respond within the latency guidelines of the PCI specifications. Note that this is not usually done on the first data phase.
- **Retry:** Retry refers to a termination requested because the PCEB is currently in a state that makes it unable to process the transaction.

Figure 18 and Figure 19 show four types of target-initiated terminations. The PCEB initiates a disconnect for PCI cycles destined to EISA after the first data phase due to incremental latency requirements. The difference between disconnect and retry is that the PCEB does not assert TRDY# for the retry case. This instructs the initiator to retry the transfer at a later time. No data is transferred in a retry termination since TRDY# and IRDY# are never both asserted. The PCEB retries a PCI initiator when:

- the PCEB buffers require management activity.
- the PCEB is locked and another PCI device attempts to select the PCEB without negating PLOCK# during the address phase.
- the EISA Bus is occupied by an EISA/ISA master or DMA.

Target abort is another form of target-initiated termination. Target abort resembles a retry, though the target must also negate DEVSEL#, along with assertion of STOP#. As a target, the PCEB never generates a target abort.

As a master, If the PCEB receives a target abort, it relinquishes the PCI Bus and sets the Received Target Abort Status bit in the PCI Status Register to a 1.

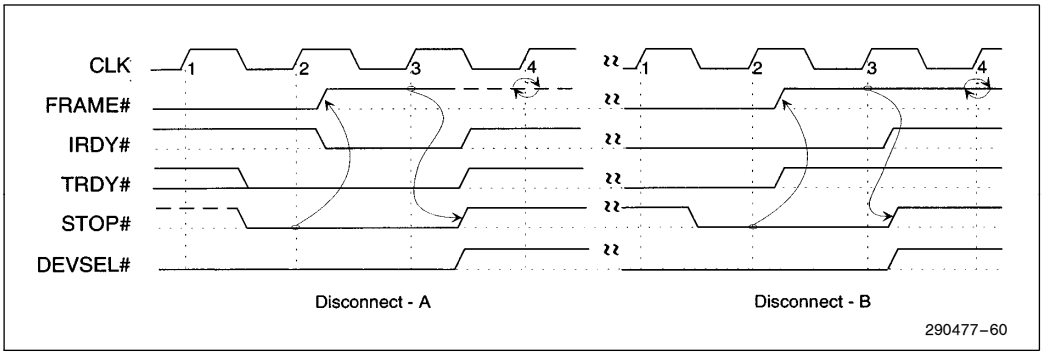


Figure 18. Target Initiated Termination

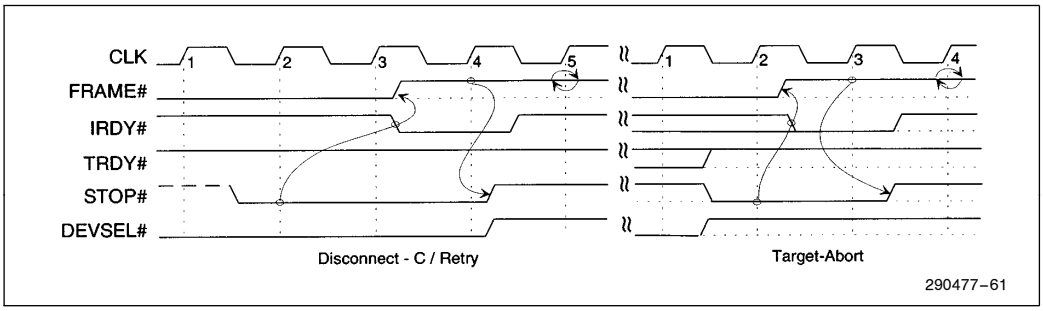


Figure 19. Target Initiated Termination

5.1.10.3 PCEB Target Termination Conditions

As a target, the PCEB terminates a transaction due to the following conditions:

Disconnect

- When a target, the PCEB always responds with a disconnect to a multiple data phase transaction (see the Incremental Latency Timer section),

Retry

- When the pending PCI cycle initiates buffer management activity.
- When the PCEB is locked, as a resource, and a PCI master tries to access the PCEB without negating the PLOCK# signal in the address phase.
- When the EISA Bus is occupied by an EISA/ISA master or DMA.

Target Abort

- The PCEB never generates a target abort.

5.1.10.4 PCEB Master Termination Conditions

As an initiator, the PCEB terminates a transaction due to the following conditions:

- Completion termination is always used by the PCEB signaling to the target that the PCEB is ready to complete the final data phase of the transaction.
- Master abort termination is issued if the PCEB does not receive a DEVSEL# from a target within five PCICLK's after FRAME# assertion. The PCEB sets the Received Master Abort Status bit in the PCI Status Register to a 1.
- Master initiated termination (disconnect) due to Master Latency Timer expiration when the PCEB's PCI Bus grant is removed (internal PCEBGNT# negated).

5.1.10.5 PCEB Responses/Results Of Termination

PCEB's response, as a target, to a master termination:

- Completion termination is the normal way of terminating a transaction.
- If a PCI initiator times out due to LT time-out and ends the current transaction, the PCEB cannot detect a difference between normal completion termination and time-out forced termination.

PCEB's response as a master to target termination:

- If the PCEB receives a target abort, it means that the target device is not capable of handling the transaction. The PCEB does not try the cycle again. If an EISA/ISA master or the DMA is waiting for the PCI cycle to terminate (EXRDY negated), the target abort condition causes the PCEB to assert EXRDY to terminate the EISA cycle. Note that write data is lost and the read data is meaningless. This is identical to the way an unclaimed cycle is handled on the ''normally ready'' EISA Bus. If the Line Buffer is the requester of the PCI transaction, the target abort mechanism ends the PCI cycle, but no valid data transfers are performed into or out of the Line Buffer. The Line Buffer does not try the cycle again. The Received Target Abort Status bit in the PCI Status Register is set to 1 indicating that the PCEB experienced a target abort condition.
- If the PCEB is retried as an initiator on the PCI Bus, it will remove it's request for 2 PCI clocks before asserting it again to retry the cycle.

- If the PCEB is disconnected as an initiator on the PCI Bus, it will respond very much as if it had been retried. The difference between retry and disconnect is that the PCEB did not see any data phase for the retry. Disconnect may be generated by a PCI slave when the PCEB is running a burst memory cycle to empty or to fill one line (16-byte) of the Line Buffers. In this case, the PCEB may need to finish a multi-data phase transfer and recycles through arbitration as required for a retry. An example is when an EISA agent (EISA/ISA master or DMA) issues a read request that the PCEB translates into a 16 byte prefetch (one line) and the PCEB is disconnected before the Line Buffer is completely filled.

5.1.11 PCI DATA TRANSFERS WITH SPECIFIC BYTE ENABLE COMBINATIONS

Non-Contiguous Combination of Byte Enables
As a master, the PCEB might generate non-contiguous combinations of data byte enables because of the nature of assembly operations in the Line Buffers.

As a target, the PCEB might need to respond to a non-contiguous combination of data byte enables. These cycles can not be passed directly to the EISA Bus; the EISA Bus specification does not allow non-contiguous combinations of byte enables. If this situation occurs, the PCEB splits the 32-bit transactions into two 16-bit transactions by first performing the lower word transfer (indicated by BE1# and BE0#) and then the upper word transfer (indicated by BE3# and BE2#).

BE[3:0]# = 1111
As a master, the PCEB might generate this combination of data byte enables during Line Buffer flush operations (burst write) to optimize the usage of the PCI Bus. Correct parity is driven during this transaction on the PCI Bus.

As a target, the PCEB might need to respond to this combination of data byte enables. If BE[3:0]# = 1111, the PCEB completes the transfer by asserting TRDY# and providing parity for read cycles. The PCEB does not forward the cycle to the EISA Bus.

5.2 PCI Bus Latency

The PCI specification provides two mechanisms that limit a master's time on the bus. They ensure predictable bus acquisitions when other masters are requesting bus access. These mechanisms are master-initiated termination supported by a Master Latency Timer (MLT) and a target-initiated termination (specifically, disconnect) supported by a target's incremental latency mechanism.

5.2.1 MASTER LATENCY TIMER (MLT)

The PCEB has a programmable Master Latency Timer (MLT). The MLT is cleared and suspended when the PCEB is not asserting FRAME#. The MLT is controlled via the MLT Register (see Section 4.1, PCEB Configuration Registers). When the PCEB, as a master, asserts FRAME#, it enables its MLT to count. If the PCEB completes its transaction (negates FRAME#) before the count expires, the MLT is ignored. If the count expires before the transaction completes (count = number clocks programmed into the MLT Register), the PCEB initiates a transaction termination as soon as its GNT# is removed. The number of clocks programmed into the MLT Register represents the guaranteed time slice (measured in PCICLKs) allotted to the PCEB; after which it surrenders the bus as soon as its GNT# is removed. (Actual termination does not occur until the target is ready.) Each master on PCI contains a master latency timer. The relative values programmed in each master timer determines how much of the PCI bandwidth is available to that master. Generally, if the EISA bus is heavily loaded with masters, the PCEB MLT register would be programmed with a relatively large value to give the PCEB a larger share of the PCI bus bandwidth.

5.2.2 INCREMENTAL LATENCY MECHANISM

As a target, the PCEB supports the Incremental Latency Mechanism for PCI-to-EISA cycles. The PCI specification states that for multi-data phase PCI cycles, if the incremental latency from current data phase (N) to the next data phase (N + 1) is greater than eight PCICLKs, the target must manipulate TRDY# and STOP# to stop the transaction after the current data phase (N). All PCI-to-EISA cycles (memory read/write and I/O read/write) are automatically terminated (during a burst) after the first data phase because they require more than eight PCICLKs to complete on the EISA Bus.

Therefore, the PCEB does not need to specifically implement an 8 PCICLK timer and the PCEB handles a disconnect in a pre-determined fashion, based on the type of current transaction.

5.3 PCI Bus Parity Support And Error Reporting

PCI provides for parity and asynchronous system errors to be detected and reported separately. The PCEB/ESC chip set implements both mechanisms. The PCEB implements only parity generation and checking and it does not interface to the SERR# signal. Reporting of both PERR# and SERR# indicated errors is implemented in the ESC.

5.3.1 PARITY GENERATION AND CHECKING

The PCEB supports parity generation and checking on the PCI Bus. During the address and data phases, parity covers AD[31:0] and the C/BE[3:0]# lines, regardless of whether or not all lines carry meaningful information. Byte lanes that are not actually transferring data are still required to be driven with stable (albeit meaningless) data and are included in the parity calculation. Parity is calculated such that the number of 1s on AD[31:0], C/BE[3:0]#, and the PAR signals is an even number.

The role of the PCEB in parity generation/checking depends on the phase of the cycle (address or data), the type of bus cycle (read or write), and whether the PCEB is a master or target. The following paragraphs and Figure 20 summarize the behavior of the PCEB during the address and data phase of a PCI Bus cycle.

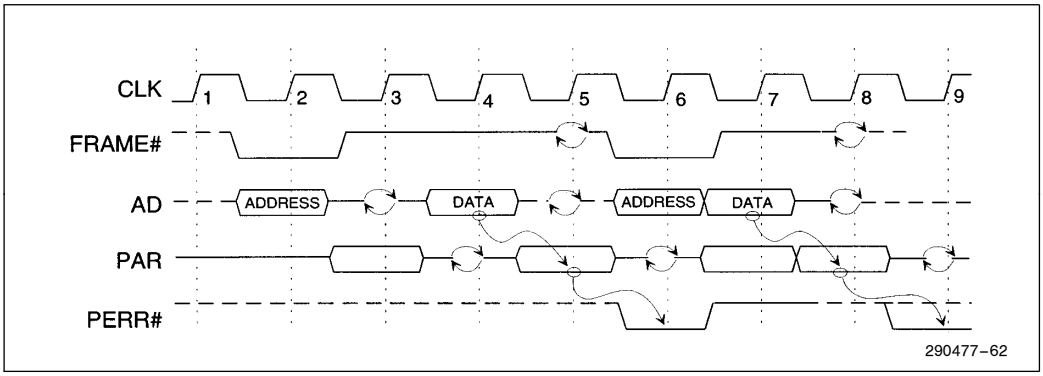


Figure 20. Parity Operation

5.3.1.1 Address Phase

As a master, the PCEB drives AD[31:0] and C/BE[3:0]# and calculates the corresponding parity value and drives it on the PAR signal, 1 clock later. As a target, the PCEB does not check parity during the address phase of a bus cycle.

5.3.1.2 Data Phase

As a master during a write cycle, the PCEB drives AD[31:0] and C/BE[3:0]# and calculates the corresponding parity value and drives it on the PAR signal, 1 clock later.

As a master during a read cycle, the PCEB only drives C/BE[3:0]#. The responding target drives AD[31:0] lines (data) and calculates parity based on the received C/BE[3:0]# and outgoing AD[31:0] signals. The target drives PAR during the following clock. The PCEB calculates parity based on the outgoing C/BE[3:0]# and the incoming AD[31:0] signals at the end of the data phase. It compares it with the incoming value of the PAR signal and asserts PERR# if there is no match.

As a target during a write cycle, the PCEB calculates parity on the incoming AD[31:0] and C/BE[3:0]# signals, and compares the result on the next clock with the incoming value on the PAR signal. If the value does not match, the PCEB asserts PERR#.

As a target during a read cycle, the PCEB calculates parity on the incoming C/BE[3:0]# and outgoing AD[31:0] signals. The PCEB drives the calculated parity value during the next clock. The master of the transaction receives the data, calculates parity on its outgoing C/BE[3:0]# and incoming AD[31:0] signals and compares its calculated value, on the next clock, with the parity value on the PAR signal (supplied by the PCEB). If the values do not match, the master asserts PERR#.

5.3.2 PARITY ERROR—PERR# SIGNAL

When the PCEB is involved in a bus transaction (master or target), it asserts the PERR# signal, if enabled via the PCICMD Register, to indicate a parity error for the bus cycle. PERR# is a sustained tri-state (s/t/s) type of signal (see Section 2.0, Signal Description). Note that PCI parity errors signaled by PERR#, are reported to the host processor via the ESC's system interrupt control logic. When the PCEB detects a parity error during one of its bus transactions, it sets the parity error status bit in the PCI Status Register, regardless of whether the PERR# signal is enabled via the PCICMD Register.

5.3.3 SYSTEM ERRORS

The PCEB does not generate system errors (SERR#). Thus, the PCEB does not have the capability of indicating parity errors during the address phase in which it is a potential target (i.e. not a master). Note that system errors are reported via the ESC (companion chip).

5.4 PCI Bus Arbitration

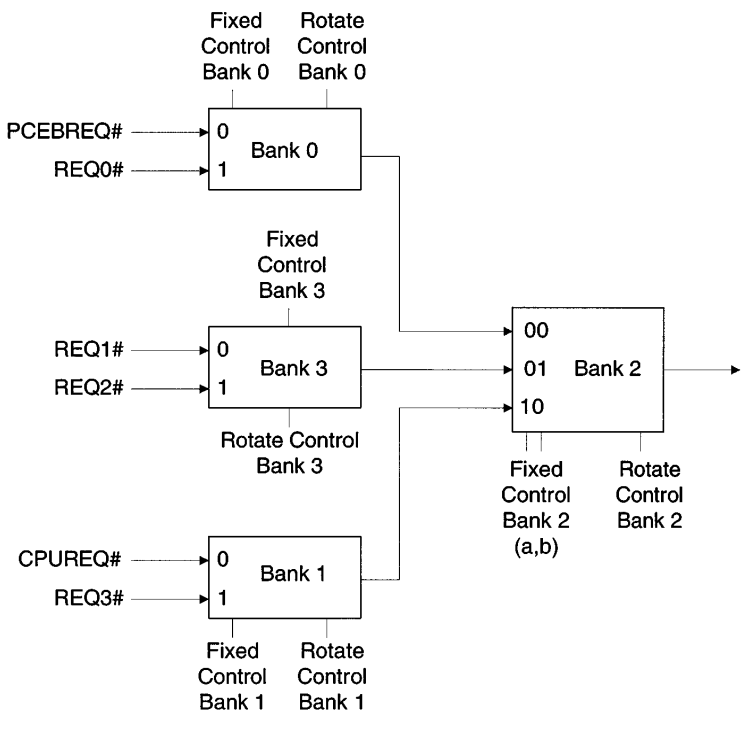
The PCEB contains a PCI Bus arbiter that supports six PCI Bus mastersThe Host/PCI Bridge, PCEB, and four other masters. The PCEB's REQ#/GNT# signals are internal. An external arbiter is not supported. Note that, for proper arbiter operation, CPUREQ# must be sampled high by the PCEB when PCIRST# makes a low-to-high transition. The internal arbiter contains several features that contribute to system efficiency:

- Use of the internal RESUME# signal to re-enable a backed-off initiator in order to minimize PCI Bus thrashing when the PCEB generates a retry.
- A programmable timer to re-enable retried initiators after a number of PCICLK's.
- A programmable PCI Bus lock or PCI resource lock function.
- The CPU Host/PCI can be optionally parked on the PCI Bus.

In addition, the PCEB has three PCI sideband signals (FLUSHREQ#, MEMREQ#, and MEMACK#) that are used to control system buffer coherency and control operations for the Guaranteed Access Time (GAT) mode.

5.4.1 PCI ARBITER CONFIGURATION

The PCI arbitration priority scheme is programmable through the configuration registers. The arbiter consists of four banks that can be configured so that the six masters to be arranged in a purely rotating priority scheme, one of 24 fixed priority schemes, or a hybrid combination (Figure 21).



NOTE:
PCEBREQ#/PCEBGNT# are PCEB internal signals.

Figure 21. Arbiter Conceptual Block Diagram

The PCEB implements PCI arbiter priority configuration registers ARBPRI and ARBPRIX mapped in the PCI's configuration space. Definition of the registers are as follows:

ARBPRI Register

Bit	Description
7	Bank 3 Rotate Control
6	Bank 2 Rotate Control
5	Bank 1 Rotate Control
4	Bank 0 Rotate Control
3	Bank 2 Fixed Priority Mode Select B
2	Bank 2 Fixed Priority Mode Select A
1	Bank 1 Fixed Priority Mode Select
0	Bank 0 Fixed Priority Mode Select

This register defaults to 04h at reset. This selects fixed mode #4 with the CPU the highest priority device guaranteeing that BIOS accesses can take place.

ARBPRIX Register

Bit	Description
7	Bank 3 Fixed Priority Mode select
6:0	Reserved

This register defaults to 00h at reset. The default value selects REQ1# as a higher priority request than REQ2# when Bank 3 operates in the fixed priority mode.

5.4.1.1 Fixed Priority Mode

The twelve selectable fixed priority schemes are listed in Table 6.

Table 6. Fixed Priority Mode Bank Control Bits

Mode	Bank					Priority				
	3	2b	2a	1	0	Highest				Lowest
0	0	0	0	0	0	PCEBREQ#	REQ0#	REQ1# / REQ2#	CPUREQ#	REQ3#
1	0	0	0	0	1	REQ0#	PCEBREQ#	REQ1# / REQ2#	CPUREQ#	REQ3#
2	0	0	0	1	0	PCEBREQ#	REQ0#	REQ1# / REQ2#	REQ3#	CPUREQ#
3	0	0	0	1	1	REQ0#	PCEBREQ#	REQ1# / REQ2#	REQ3#	CPUREQ#
4	0	0	1	0	0	CPUREQ#	REQ3#	PCEBREQ#	REQ0#	REQ1# / REQ2#
5	0	0	1	0	1	CPUREQ#	REQ3#	REQ0#	PCEBREQ#	REQ1# / REQ2#
6	0	0	1	1	0	REQ3#	CPUREQ#	PCEBREQ#	REQ0#	REQ1# / REQ2#
7	0	0	1	1	1	REQ3#	CPUREQ#	REQ0#	PCEBREQ#	REQ1# / REQ2#
8	0	1	0	0	0	REQ1# / REQ2#	CPUREQ#	REQ3#	PCEBREQ#	REQ0#
9	0	1	0	0	1	REQ1# / REQ2#	CPUREQ#	REQ3#	REQ0#	PCEBREQ#
A	0	1	0	1	0	REQ1# / REQ2#	REQ3#	CPUREQ#	PCEBREQ#	REQ0#
B	0	1	0	1	1	REQ1# / REQ2#	REQ3#	CPUREQ#	REQ0#	PCEBREQ#

Table 6. Fixed Priority Mode Bank Control Bits (Continued)

Mode	Bank					Priority				
	3	2b	2a	1	0	Highest				Lowest
	x	1	1	x	x	Reserved				
10	1	0	0	0	0	PCEBREQ#	REQ0#	REQ2# / REQ1#	CPUREQ#	REQ3#
11	1	0	0	0	1	REQ0#	PCEBREQ#	REQ2# / REQ1#	CPUREQ#	REQ3#
12	1	0	0	1	0	PCEBREQ#	REQ0#	REQ2# / REQ1#	REQ3#	CPUREQ#
13	1	0	0	1	1	REQ0#	PCEBREQ#	REQ2# / REQ1#	REQ3#	CPUREQ#
14	1	0	1	0	0	CPUREQ#	REQ3#	PCEBREQ#	REQ0#	REQ2# / REQ1#
15	1	0	1	0	1	CPUREQ#	REQ3#	REQ0#	PCEBREQ#	REQ1# / REQ2#
16	1	0	1	1	0	REQ3#	CPUREQ#	PCEBREQ#	REQ0#	REQ2# / REQ1#
17	1	0	1	1	1	REQ3#	CPUREQ#	REQ0#	PCEBREQ#	REQ2# / REQ1#
18	1	1	0	0	0	REQ2# / REQ1#	CPUREQ#	REQ3#	PCEBREQ#	REQ0#
19	1	1	0	0	1	REQ2# / REQ1#	CPUREQ#	REQ3#	REQ0#	PCEBREQ#
1A	1	1	0	1	0	REQ2# / REQ1#	REQ3#	CPUREQ#	PCEBREQ#	REQ0#
1B	1	1	0	1	1	REQ2# / REQ1#	REQ3#	CPUREQ#	REQ0#	PCEBREQ#
	x	1	1	x	x	Reserved				

Note that these two tables are permutations of the same table with different value of the Bank 3 fixed priority control bit. The fixed bank control bit(s) selects which requester is the highest priority device within that particular bank. Bits[7:4] must be programmed to all 0's (rotate mode disabled) to get these combinations. The selectable fixed priority schemes provide 24 of the 128 possible fixed mode permutations possible for the six masters.

5.4.1.2 Rotating Priority Mode

When any bank rotate control bit is set to a one, that particular bank rotates between the requesting inputs. Any or all banks can be set in rotate mode. If all four banks are set in rotate mode, the six supported masters are all rotated and the arbiter is in a pure rotating priority mode. If, within a rotating bank, the highest priority device (a) does not have an active request, the lower priority device (b or c) will be granted the bus. However, this does not change the rotation scheme. When the bank toggles, device b is the highest priority. Because of this, the maximum latency a device can encounter is two complete rotations.

5.4.1.3 Mixed Priority Mode

Any combination of fixed priority and rotate priority modes can be used in different arbitration banks to achieve a specific arbitration scheme.

5.4.1.4 Locking Masters

When a master acquires the PLOCK# signal, the arbiter gives that master highest priority until PLOCK# is negated and FRAME# is negated. This insures that a master that locked a resource will eventually be able to unlock that same resource.

5.4.2 ARBITRATION SIGNALING PROTOCOL

An agent requests the PCI Bus by asserting its REQ#. When the arbiter determines that an agent may use the PCI Bus, it asserts the agent's GNT#. Figure 22 shows an example of the basic arbitration cycle. Two agents (A and B) are used to illustrate how the arbiter alternates bus accesses. Note in Figure 22 that the current owner of the bus may keep its REQ# (REQ#-A) asserted when it requires additional transactions.

REQ#-A is asserted prior to or at clock 1 to request use of the PCI Bus. Agent A is granted access to the bus (GNT#-A is asserted) at clock 2. Agent A may start a transaction at clock 2 because FRAME# and IRDY# are negated and GNT#-A is asserted. Agent A's transaction starts when FRAME# is asserted (clock 3). Agent A requests another transaction by keeping REQ#-A asserted.

When FRAME# is asserted on clock 3, the arbiter determines that agent B has priority and asserts GNT#-B and negates GNT#-A on clock 4. When agent A completes its transaction on clock 4, it relinquishes the bus. All PCI agents can determine the end of the current transaction when both FRAME# and IRDY# are negated. Agent B becomes the PCI Bus owner on clock 5 (FRAME# and IRDY# are negated) and completes its transaction on clock 7. Note that REQ#-B is negated and FRAME# is asserted on clock 6, indicating that agent B requires only a single transaction. The arbiter grants the next transaction to agent A because its REQ# is still asserted.

5.4.2.1 REQ# And GNT# Rules

Figure 22 illustrates basic arbitration. Once asserted, GNT# may be negated according to the following rules:

1. If GNT# is negated at the same time that FRAME# is asserted, the bus transaction is valid and will continue.
2. One GNT# can be negated coincident with another being asserted, if the bus is not in the idle state. Otherwise, a one clock delay is incurred between the negation of the current master's GNT# and assertion of the next master's GNT#, to comply with the PCI specification.
3. While FRAME# is negated, GNT# may be negated, at any time, in order to service a higher priority master, or in response to the associated REQ# being negated.
4. If the MEMREQ# and MEMACK# are asserted, once the PCEB is granted the PCI Bus, the arbiter will not remove the internal grant until the PCEB removes it's request.

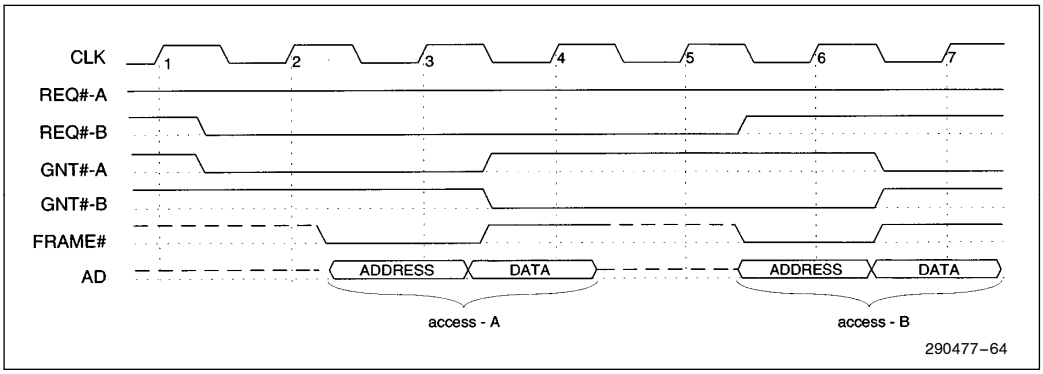


Figure 22. Basic Arbitration

5.4.2.2 Back-to-Back Transactions

Figure 23 illustrates arbitration for a back-to-back access. There are two types of back-to-back transactions by the same initiator; those that do not require a turn-around-cycle (see Section 5.1.3.1, Turn-Around-Cycle Definition) and those that do. A turn-around-cycle is not required when the initiator's second transaction is to the same target as the first transaction (to insure no TRDY# contention), and the first transaction is a write. This is a fast back-to-back. Under all other conditions, the initiator must insert a minimum of one turn-around-cycle.

During a fast back-to-back transaction, the initiator starts the next transaction immediately, without a turn-around-cycle. The last data phase completes when FRAME# is negated, and IRDY# and TRDY# are asserted. The current initiator starts another transaction on the same PCICLK that the last data is transferred for the previous transaction.

As a master, the PCEB does not know if it is accessing the same target, and, thus, does not generate fast back-to-back accesses. As a slave, the PCEB is capable of decoding fast back-to-back cycles.

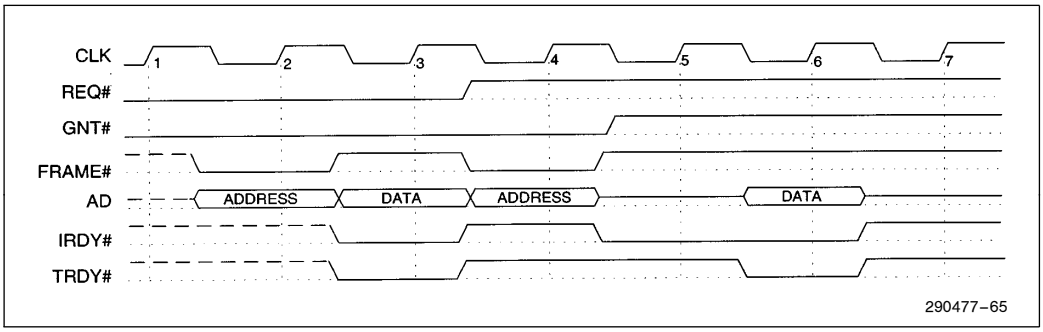


Figure 23. Arbitration for Back-to-Back Access

5.4.3 RETRY THRASHING RESOLVE

When a PCI initiator's access is retried, the initiator releases the PCI Bus for a minimum of two PCI clocks and then normally requests the PCI Bus again. To avoid thrashing of the bus with retry after retry, the PCI arbiter's state tracer provides REQ# masking. Tracking retried masters requires latching GNT# during FRAME# so that the correct retried master can be masked. The state tracer masks a REQ# after that particular agent is retried on the PCI Bus. The state tracer differentiates between two retry events. The two events include PCEB target retries and all other retries

For initiators that were retried by the PCEB as a target, the masked REQ# is flagged to be cleared upon RESUME# active. All other retries trigger the Master Retry Timer (described in Section 5.4.4.2, Master Retry Timer). When this timer expires, the mask is cleared.

5.4.3.1 Resume Function

The PCEB forces a retry to a PCI master (resulting in masking the REQ# of that master) for the following:

1. Buffer management activities (See Section 6.0, Data Buffering).
2. The EISA Bus is occupied by an EISA/ISA master or DMA.
4. The PCEB is locked as a resource and PLOCK# is asserted during the address phase.

5.4.3.2 Master Retry Timer

For any other retried PCI cycle, the arbiter masks the REQ# and flags it to be cleared by the expiration of a programmable timer. The first retry in this category triggers the programmable timer. Subsequent retries in this category are masked but do not reset the timer. Expiration of this programmable timer unmasks all REQ#s that are masked for this reason. The Retry Timer is programmable to 0 (disabled), 16, 32, or 64 PCICLKs.

If no other PCI masters are requesting the PCI Bus, all of the REQ#s masked for the timer are cleared and the timer is set to 0. Note that when there is a pending request that is internally masked, the PCEB does not park the CPU on the PCI Bus (i.e. PCI agent that uses CPUREQ#/CPUGNT# signal pair).

5.4.4 BUS LOCK MODE

As an option, the PCEB arbiter can be configured to run in Bus Lock Mode or Resource Lock Mode (default). The Bus Lock Mode is used to lock the entire PCI Bus. This may improve performance in some systems that frequently run quick read-modify-write cycles (i.e. access to the VGA frame buffer using the XCHG x86 instruction that automatically asserts the CPU LOCK# signal). Bus Lock Mode emulates the LOCK environment found in today's PC by restricting bus ownership when the PCI Bus is locked. While Bus Lock Mode improves performance in some systems, it may cause performance problems in other systems. With Bus Lock enabled, the arbiter recognizes a LOCK# being driven by an initiator and does not allow any other PCI initiator to be granted the PCI Bus until LOCK# and FRAME# are both negated, indicating the master released lock. When Bus Lock is disabled, the default resource lock mechanism is implemented (normal resource lock) and a higher priority PCI initiator could intervene between the cycles that are part of the locked sequence and run non-exclusive accesses to any unlocked resource.

CAUTION:

Bus Lock mode should not be used with non-GAT mode. If the system is initialized for both Bus Lock mode and non-GAT mode a deadlock situation might occur in the case where the first access to the locked device is a write instead of a read and the locked device has data in its internal posted write buffer. In GAT mode and/or Resource Lock mode this condition can not happen. If it is absolutely necessary to operate the system in the above mentioned combination of modes, then the posted write buffers of the device that might be involved in locked operations (typically semaphore in main memory) must be disabled.

5.4.5 MEMREQ#, FLSHREQ#, AND MEMACK# PROTOCOL

Before an EISA master or DMA can be granted the PCI Bus, it is necessary that all PCI system posted write buffers be flushed. Also, since the EISA-originated cycle could access memory on the Host/PCI Bridge, it is possible that the EISA master or DMA could be held in wait states (via EXRDY) waiting for the Host/PCI Bridge arbitration for longer than the 2.1 μs EISA/ISA specification. The PCEB has an optional mode called Guaranteed Access Time mode (GAT) that ensures that this timing specification is not violated. This is accomplished by delaying the EISA grant signal to the requesting master or DMA until the EISA Bus, PCI Bus, and the system memory bus are arbitrated for and owned.

The three sideband signals, MEMREQ#, FLSHREQ#, and MEMACK# are used to support the system Posted Write Buffer flushing and Guaranteed Access Time mechanism. The MEMACK# signal is the common acknowledge signal for both mechanisms. Note that, when MEMREQ# is asserted, FLSHREQ# is also asserted. Table 7 shows the relationship between MEMREQ# and FLSHREQ#.

Table 7. FLSHREQ# and MEMREQ#

FLSHREQ#	MEMREQ#	Meaning
1	1	Idle
0	1	Flush buffers pointing towards the PCI Bus to avoid EISA deadlock
1	0	Flush buffers pointing towards main memory for buffer coherency in APIC systems
0	0	GAT mode. Guarantees PCI Bus immediate access to main memory

5.4.5.1 Flushing System Posted Write Buffers

Once an EISA Bus owner (EISA/ISA master or the DMA) begins a cycle on the EISA Bus, the cycle can not be backed-off. It can only be held in wait states via EXRDY. In order to know the destination of EISA master cycles, the cycle needs to begin. After the cycle is started, no other device can intervene and gain ownership of the EISA Bus until the cycle is completed and arbitration is performed. A potential deadlock condition exists when an EISA-originated cycle to the PCI Bus forces a mandatory transaction to EISA, or when the PCI target is inaccessible due to an interacting event that also requires the EISA Bus. To avoid this potential deadlock, all PCI posted write buffers in the system must be disabled and flushed, before an EISA/ISA master or DMA can be granted the EISA Bus. The buffers must remain disabled while the EISA Bus is occupied. The following steps indicate the PCEB (and ESC) handshake for flushing the system posted write buffers.

1. When an EISA/ISA master, DMA or refresh logic requests the EISA Bus, the ESC component asserts EISAHOLD to the PCEB.
2. The PCEB completes the present cycle (and does not accept any new cycle) and gives the EISA Bus to the ESC by floating its EISA interface and asserting EISAHLDA. Before giving the bus to the ESC, the PCEB checks to see if it itself is locked as a PCI resource. It can not grant the EISA Bus as long as the PCEB is locked.

   At this point the PCEB's EISA-to-PCI Line Buffers and other system buffers (Host/PCI Bridge buffers) that are pointing to PCI are not yet flushed. The reason for this is that the ESC might request the bus in order to run a refresh cycle that does not require buffer flushing. That is not known until the EISA arbitration is frozen (after EISAHLDA is asserted).

   a. If the ESC needs to perform a refresh cycle, then it negates NMFLUSH# (an ESC-to-PCEB flush control signal). ESC drives the EISA Bus until it completes the refresh cycle and then gives the bus to the PCEB by negating EISAHOLD.
   b. If the ESC requested the EISA Bus on behalf of the EISA master, DMA or ISA master, then it asserts NMFLUSH# and tri-states the EISA Bus. The PCEB asserts the FLSHREQ# signal to the Host/PCI Bridge (and other bridges) to disable and flush posted write buffers.
3. When the Host/PCI Bridge completes its buffer disabling and flushing, it asserts MEMACK# to the PCEB. Other bridges in the system may also need to disable and flush their posted write buffers pointing towards PCI. This means that other devices may also generate MEMACK#. All of the MEMACK#s need to be ''wire-OR'd''. When the PCEB receives MEMACK# indicating that all posted write buffers have been flushed, it asserts NMFLUSH# to the ESC and the ESC gives the bus grant to the EISA device.
4. The PCEB continues to assert FLSHREQ# while the EISA/ISA master or DMA owns the EISA Bus. While FLSHREQ# is asserted the Host/PCI Bridge must keep its posted write buffers flushed.
5. MEMACK# should be driven inactive as soon as possible by the Host/PCI Bridge and other bridges after FLSHREQ# is negated. The PCEB waits until it detects MEMACK# negated before it can generate another FLSHREQ#.

5.4.5.2 Guaranteed Access Time Mode

When the PCEB's Guaranteed Access Time Mode is enabled (via the ARBCON Register), MEMREQ# and MEMACK# are used to guarantee that the ISA 2.1μs CHRDY specification is not violated. Note that EISA's 2.5 μs maximum negation time of the EXRDY signal is a subset of the ISA requirement. Thus, 2.1 μs satisfies both bus requirements.

When an **EISA/ISA master or DMA slave** requests the EISA Bus (MREQ# or DREQ# active), the EISA Bus, the PCI Bus, and the memory bus must be arbitrated for and all three must be owned before the EISA/ISA master or DMA is granted the EISA Bus. The following lists the sequence of events:

1. An EISA/ISA master, DMA, or refresh logic requests the EISA Bus. The ESC asserts EISAHOLD signal to the PCEB.
2. The PCEB completes the present cycle (i.e. does not accept any new cycle) and gives the bus to the ESC by floating its EISA interface and asserting EISAHLDA. Before giving the bus to the ESC, the PCEB checks to see if it is locked as a PCI resource. It can not grant the EISA Bus as long as the PCEB is locked.

At this point, the PCEB's EISA-to-PCI Line Buffers and other system buffers (e.g., Host/PCI Bridge buffers) that are pointing to the PCI Bus are not flushed. The reason is that the ESC might request the bus to run a refresh cycle that does not require buffer flushing. This is not known until the EISA arbitration is frozen (after EISAHLDA is asserted).

3. Depending on whether the pending cycle is a refresh, the ESC initiates one of the following two actions:
   a. If the ESC needs to perform a refresh cycle, then it asserts NMFLUSH# (an ESC-to-PCEB flush control signal). The ESC drives the EISA Bus until it completes the refresh cycle and then gives the bus to the PCEB by negating EISAHOLD.
   b. If the ESC requested the EISA Bus on behalf of the EISA master, DMA or ISA master, then it asserts NMFLUSH# and tri-states the EISA Bus. If the PCEB is programmed in GAT (Guaranteed Access Time mode), the MEMREQ# and FLSHREQ# signals are asserted simultaneously to indicate request for direct access to main memory and a request to flush the system's posted write buffers pointing towards the PCI (including the PCEB's internal buffers). These requirements are necessary to insure that once the PCI and EISA Buses are dedicated to the PCEB, the cycle generated by the PCEB will not require the PCI or EISA Buses, thus creating a deadlock. MEMREQ# and FLSHREQ# are asserted as long as the EISA/ISA master or DMA owns the EISA Bus.
4. Once the Host/PCI Bridge has disabled and flushed its posted write buffers, and the memory bus is dedicated to the PCI interface, it asserts MEMACK#. Other bridges in the system may also need to disable and flush their posted write buffers pointing towards PCI due to the FLSHREQ# signal. This means that other devices may also generate a MEMACK#. All of the MEMACK#s need to be ''wire-OR'd''. When the PCEB receives MEMACK#, it assumes that all of the critical posted write buffers in the system have been flushed and that the PCEB has direct access to main memory, located behind the Host/PCI Bridge.
5. When MEMACK# is asserted by the PCEB, it will request the PCI Bus (internal PCEBREQ# signal). Before requesting the PCI Bus, the PCEB checks to see that the PCI Bus does not have an active lock. The PCI Bus is granted to the PCEB when it wins the bus through the normal arbitration mechanism. Once the PCEB is granted the PCI Bus (internalPCEBGNT#), the PCEB checks to see if PLOCK# is negated before it grants the EISA Bus. If the PCI Bus is locked when the PCEB is granted the PCI Bus, the PCEB releases the REQ# signal and waits until the PLOCK# is negated before asserting REQ# again. Once the PCEB owns the PCI Bus (internal PCEBGNT#), and the MEMACK# and MEMREQ# signals are asserted, the PCI arbiter will not grant the PCI Bus to any other PCI master except the PCEB until the PCEB releases its PCI REQ# line.
6. When the PCEB is granted the PCI Bus (internal PCEBGNT#) and LOCK# is inactive, it asserts NMFLUSH# to the ESC and the ESC gives the bus grant to the EISA device.

7. When the EISA Bus is no longer owned by an EISA master or DMA, the PCEB negates MEMREQ# and FLSHREQ# and the PCI request signal (internal PCEBREQ#. The negation of MEMREQ# and FLSHREQ# indicates that direct access to the resource behind the bridge is no longer needed and that the posted write buffers may be enabled. Note that MEMACK# should be driven inactive as soon as possible by the Host/PCI Bridge and other bridges after MEMREQ# is negated. The PCEB waits until it detects MEMACK# negated before it can generate another MEMREQ# or FLSHREQ#.

The use of MEMREQ#, FLSHREQ#, and MEMACK# does not guarantee GAT mode functionality with ISA masters that don't acknowledge CHRDY. These signals just guarantee the CHRDY inactive specification.

5.4.5.3 Interrupt Synchronization-Buffer Flushing

The ESC contains the system interrupt controller consisting of an 8259 compatible interrupt controller and an I/O APIC. For the 8259 compatible interrupt controller, the PCEB/ESC chip set is the default destination of the PCI interrupt acknowledge cycles. Interrupts in the system are commonly used as a synchronization mechanism. If interrupts are used by the EISA agents to notify the Host CPU that data is written to main memory, then posted data buffers must be flushed before the vector is returned during the interrupt acknowledge sequence. The PCEB handles this transparently to the rest of the system hardware/software. It retries the PCI interrupt acknowledge cycles and flushes the PCEB Line Buffers, if necessary.

The Advanced Programmable Interrupt Controller (APIC) uses a private message passing bus to send interrupt information to the companion APIC(s) residing at the host CPU(s). To support interrupts as a synchronization mechanism, system buffer coherency must be guaranteed before interrupts can be processed. With ESC's interrupt controller operating in APIC mode the PCEB and ESC use the PCEB/ESC interchip signal AFLUSH# to maintain system buffer coherency.

5.4.6 BUS PARKING

PCI Bus parking can be enabled/disabled via the ARBCON Register. Parking is only allowed for the device that is connected to CPUREQ# (i.e. the Host/PCI Bridge). REQ[3:0]#, and the internal PCEBREQ# are not allowed to park on the PCI Bus. When bus parking is enabled, CPUGNT# is asserted when no other agent is currently using or requesting the bus. This achieves the minimum PCI arbitration latency possible.

Arbitration Latency

Parked: 0 PCICLKs for parked agent, 2 PCICLKs for all other.

Not Parked: 1 PCICLK for all agents.

Upon assertion of CPUGNT# due to bus parking enabled and the PCI Bus idle, the CPU (i.e., parked agent) must ensure AD[31:0], C/BE[3:0]#, and (one PCICLK later) PAR are driven. If bus parking is disabled, then the PCEB drives these signals when the bus is idle.

5.4.7 PCI ARBITRATION AND PCEB/ESC EISA OWNERSHIP EXCHANGE

There are two aspects of PCEB/ESC EISA Bus ownership exchange that are explained in this section. They are related to GAT mode and RESUME/RETRY operations.

The PCEB is the default owner of the EISA Bus. When control of the EISA Bus is given to the ESC, all PCI operations targeted to the EISA subsystem (including the PCEB) are retried. Retry causes assertion of the PEREQ#/INTA# signal with PEREQ# semantics. In this way, the PCEB indicates to the ESC that it needs to obtain ownership of the EISA Bus.

5.4.7.1 GAT Mode And PEREQ# Signaling

In GAT mode, the PCEB owns the PCI Bus on behalf of the EISA master and other PCI agents (e.g., the Host/PCI Bridge) can not generate PCI cycles. Therefore, the PCEB never generates a back-off (i.e. retry), as long as the EISA Bus is controlled by the ESC. This might cause starvation of the PCI agents (including the Host/PCI Bridge i.e., CPU) even in the case of a moderately loaded EISA subsystem. The solution is that PEREQ#, in the GAT mode, is generated when any of the PCI Bus request signals are asserted. For particular Host/PCI Bridge designs (e.g. PCMC) this will be not be an adequate solution since their PCI request can be activated only based on the CPU generated cycle directed to PCI. This will not be possible since the Host Bus (CPU bus) in the GAT mode is controlled by the Host/PCI Bridge and not by the CPU. The solution to this type of design is to generate PEREQ# immediately after entering the GAT mode. This feature is controlled via ARBCON Register (bit 7).

5.4.7.2 PCI Retry And EISA Latency Timer (ELT) Mechanism

When a PCI cycle is retried by the PCEB (in non-GAT mode) because the EISA Bus is controlled by the ESC (EISAHLDA asserted), an internal flag is set for the corresponding PCI master. This flag masks the request of a particular master until the PCEB acquires the ownership of EISA and the RESUME condition clears the flag. If the PCI master, which is now unmasked, does not acquire the ownership of the PCI Bus within the time period before ESC asserts EISAHOLD again, the EISA Bus can be surrendered to the ESC. Unmasked masters will eventually gain the access to the PCI Bus but the EISA Bus will not be available and the master will be retried again. This scenario can be repeated multiple times with one or more PCI masters and starvation will occur.

To solve this situation, the PCEB arbitration logic incorporates an EISA Latency Timer mechanism. This mechanism is based on the programmable timer that is started each time that the ESC requires the bus (EISAHOLD asserted) and there is a PCI agent that has been previously retried because of activity on the EISA Bus. As soon as the ELT timer expires, any PCI cycle which is currently in progress is retried and the EISA Bus is given back to the ESC after the current PCI-to-EISA transaction completes. If all the PCI requesters, masked because of EISAHLDA, are serviced before the ELT timer expires, the EISA Bus is immediately surrendered to the ESC. The ELT provides a minimum time slice for PCI masters to access the EISA bus even if EISA masters, ISA masters or DMA devices are attempting to acquire the EISA bus.

Generally, the ELT is set to a larger value if latency sensitive PCI masters which typically access EISA are present in the system. Larger ELT values, however, do increase the worst case latency for EISA devices which typically access devices on PCI (e.g. main memory).

The EISA Latency Timer (ELT) is controlled by the ELTCR Register. The value written into ELTCR is system dependent. It is typically between 1 and 3 μs.

6.0 DATA BUFFERING

The PCEB contains data buffers (Figure 24) to isolate the PCI Bus from the EISA Bus and to provide concurrent EISA and PCI Bus operations and APIC operations. The Line Buffers are used for EISA-to-PCI memory reads and writes. A control bit in the EPMRA Registers permis the Line Buffers to be enabled (accesses are buffered) or disabled (accesses are non-buffered). Non-buffered accesses use the bypass path. Note that PCI and EISA I/O read/write cycles and PCI configuration cycles are always non-buffered and use the bypass path.

When data is temporarily stored in the buffers between the EISA Bus and PCI Bus, there are potential data coherency issues. The PCEB guarantees data coherency by intervening when data coherency could be lost and either flushing or invalidating the buffered data, as appropriate.

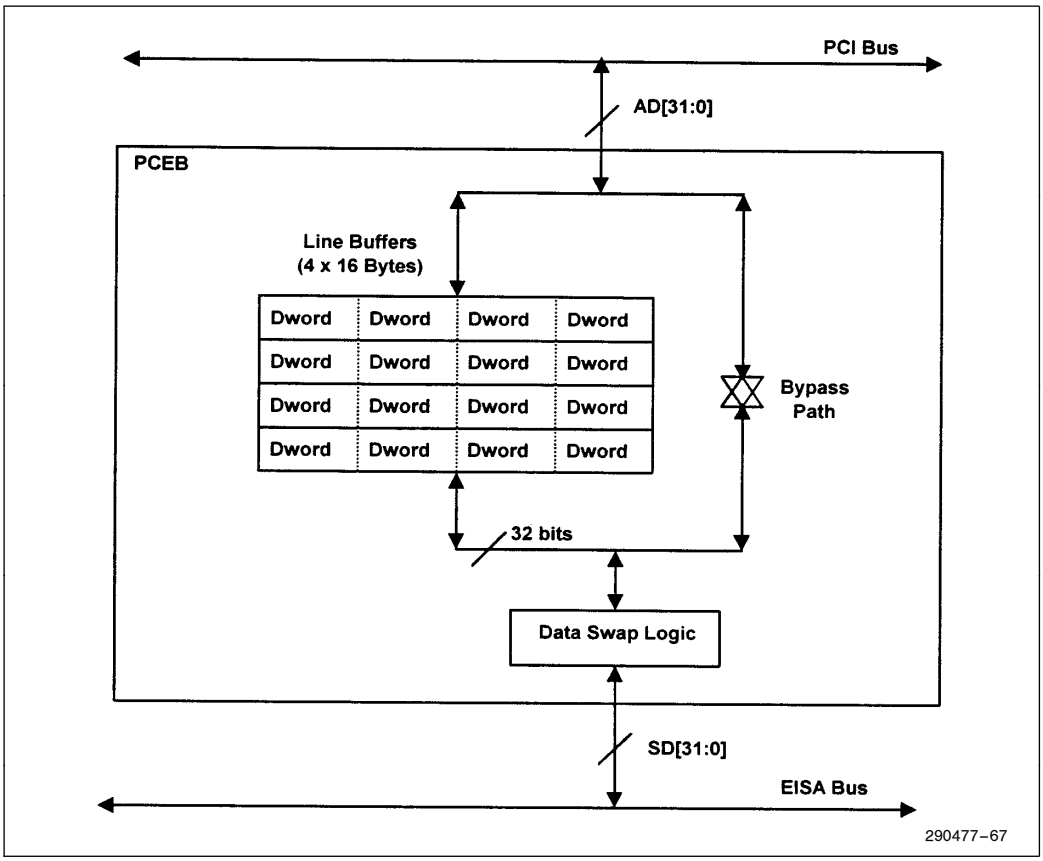


Figure 24. PCEB Data Buffers

6.1 Line Buffers

The PCEB contains four Line Buffers that are each four Dwords wide (16 bytes). The Line Buffers are bi-directional and are used by the EISA/ISA master and DMA to assemble/disassemble data. The data in each Line Buffer is aligned on 16 byte boundaries. When data is placed in one of the Line Buffers, the PCEB maintains the corresponding 16-byte boundary address until the data in the line is transferred to its destination or invalidated.

The Line Buffers can be enabled/disabled by writing to the PCICON Register. In addition, when the Line Buffers are enabled via the PCICON Register, buffering for accesses to the four programmable EISA-to-PCI memory regions (Region [4:1]) can be selectively disabled via the EPMRA Register.

During buffer operations, the four Line Buffers, collectively, are either in a write state or in a read state. These states are described in the following sections.

6.1.1 WRITE STATE

If a Line Buffer contains valid write data, it is in a *write state*. In the write state, data from the EISA/ISA master or DMA is posted in the Line Buffers. Posting means that the write operation on the EISA Bus completes when the data is latched in the buffer. The EISA master does not have to wait for the write to complete to its destination (memory on the PCI Bus). Posting permits the EISA Bus cycle to complete in a minimum time and permits concurrent EISA and PCI Bus operations. During posting, data accumulates in the Line Buffer until it is flushed (written to PCI memory) over the PCI Bus. A Line Buffer is scheduled for flushing by the PCEB when:

- the line becomes full.
- a subsequent write is a line miss (not within the current line boundary address range).
- the write is to an address of a lower Dword than the previous write. Note that writes to lower addresses within the same Dword do not cause a flush. Note also, that if two (or more) consecutive EISA Bus cycles are writes to the same Dword (i.e., the same byte or word locations within the Dword, or the same Dword for Dword writes) , the accessed buffer data is overwritten. However, if any of the flush conditions described in this list occur between the writes, the line is flushed before the next write and data is not overwritten.
- the last address location in the Line Buffer is accessed.
- a subsequent cycle is a read.
- the EISA Bus changes ownership.
- an interrupt acknowledge cycle is encountered.
- the ESC performs an EISA refresh cycle.
- the ESC's I/O APIC receives an interrupt request.

When a line is scheduled for flushing, the PCEB begins arbitration for the PCI Bus. If more than one line is scheduled to be flushed, the Line Buffers are flushed in a ''first scheduled, first to be flushed'' order. If the line to be flushed contains valid data in only one Dword, the PCEB uses a single data transfer cycle on the PCI Bus. Otherwise, flushing operations use burst transfers.

During flushing, write data within a Line Buffer is packetized into Dword quantities, when possible, for a burst transfer over the 32-bit PCI Bus. Packetizing occurs at two levels - Dwords within a line and bytes/words within a Dword. When a Line Buffer is flushed, all of the valid Dwords within the line are packetized into a single PCI burst write cycle. In addition, all valid data bytes within a Dword boundary are packetized into a single data phase of the burst cycle. Packetizing reduces the PCI arbitration latency and increases the effective PCI Bus bandwidth. When multiple Line Buffers are schedule for flushing, each Line Buffer is packetized separately. Packetizing across Line Buffer boundaries is not permitted.

During flushing, strong ordering is preserved at the Dword level (i.e., the Dwords are flushed to PCI memory in the same order that they were written into the Line Buffer). Note, however, that strong ordering is not preserved at the byte or word levels (i.e., even if byte or word transfers were used by the EISA/ISA master or DMA to sequentially write to a Dword within a Line Buffer, all of the bytes in the resulting Dword boundary are simultaneously flushed to PCI memory).

Because strong ordering is not preserved within a Dword boundary, care should be used when accessing memory-mapped I/O devices. If the order of byte or word writes to a memory-mapped I/O device needs to be preserved, buffered accesses should not be used. By locating memory-mapped I/O devices in the four programmable EISA-to-PCI memory regions, buffering to these devices can be selectively disabled.

6.1.2 READ STATE

If a Line Buffer contains valid read data, it is in a *read state*. Read data is placed in the Line Buffer by two PCEB mechanisms - fetching and prefetching. Data is placed in the Line Buffer on demand (fetching) when the data is requested by a read operation from the EISA/ISA master or DMA. The PCEB also prefetches data that has not been explicitly requested but is anticipated to be requested. Once in the Line Buffer, data is either read by the EISA/ISA master or DMA (and then invalidated) or invalidated without being read. Read data is invalidated when:

- data in the Line Buffer is read (transferred to the EISA/ISA master or DMA). This prevents reading of the same data more than once.
- a subsequent read is a line miss (not to the previously accessed Line Buffer). Valid data in the current Line Buffer is invalidated. If a new line had been prefetched during access to the current line, data in the prefetched line is not invalidated, unless the access also misses this line. In this case, the data in the prefetched line is invalidated.
- a subsequent cycle is a write. Data in all Line Buffers are invalidated.

If the requested data is in the Line Buffer, a line hit occurs and the PCEB transfers the data to the EISA/ISA master or DMA (and invalidates the hit data in the buffer). If EISA Bus reads hit two consecutive line addresses, the PCEB prefetches the next sequential line of data from PCI memory (using a PCI Bus burst transfer). This prefetch occurs concurrently with EISA Bus reads of data in the already fetched Line Buffer. If consecutive addresses are not accessed, the PCEB does not prefetch the next line.

A line miss occurs if the requested data is not in the Line Buffer. If a line miss occurs, the PCEB invalidates data in the missed Line Buffer. If the requested data is in a prefetched line, the read is serviced. If a line was not prefetched or the read missed the prefetched line, the PCEB invalidates any prefetched data and fetches the Dword containing the requested data. During this fetch, the PCEB holds off the EISA/ISA master or DMA with wait states (by negating EXRDY). When the requested data is in the Line Buffer, it is transferred to the EISA Bus. Simultaneously with the EISA Bus transfer, the PCEB prefetches the rest of the line data (Dwords whose addresses are within the line and above the Dword address of the requested data). The Dword containing the requested data and the rest of the Dwords in the line (located at higher addresses) are fetched from PCI memory using a burst transfer, unless the requested data is in the last Dword of a line. In this case, a single cycle read occurs on the PCI Bus.

For purposes of data read operations, all four 4-Dword buffers are used to form two 8-Dword lines (32 bytes each). There are only tow address pointers, one for each line. Fetching fractions of a line is accomplished as described above (i.e., starting from the first requested Dword).

The MSBURST# input signal is used to supplement control of the prefetch sequence. The MSBURST# signal is activated only when an EISA master desires to do burst transfers to access sequential data (although this is not an absolute EISA rule, i.e., theoretically the data can be non-sequential after an EISA slave indicates its ability via SLBURST#). This will occur during the first data transfer.

The Line Buffer control logic dynamically switches between two prefetch modes—Half Line Prefetch (16 bytes fetch) and Full Line Prefetch (32 bytes fetch)

The prefetch control logic has implemented a Sequential Access Flag which is cleared before the initial prefetch. Initial prefetch (first data fetch) starts in the Half Line Prefetch mode and is extended to Full Line Prefetch mode immediately after MSBURST# is sampled asserted at which time the Sequential Access Flag is automatically set (this is done on-the-fly during the first line fetch). If after the initial prefetch the Sequential Access Flag has not been set (MSBURST# remained not asserted) and the control logic recognizes two consecutive hits (in incrementally sequential Dwords including the first one which is originally requested), the Sequential Access Flag is set and the prefetch control logic switches to Full Line Prefetch mode. An additional 32-byte line (or fraction depending on alignment) will be fetched.

When the Sequential Access Flag is set, prefetching is accomplished using the Full Line Prefetch mode. Each time a line buffer (32 bytes) is available, an additional line will be fetched as long as the Sequential Access Flag remains set.

When out-of-order access is recognized within the prefetched data or a miss occurs when there is valid fetched data, the Sequential Access Flag is cleared and the prefetch mode changes to Half Line Prefetch. Also, the Sequential Access Flag is cleared when MSBURST# transitions from active to inactive.

When the Sequential Access Flag is not asserted, the prefetch control logic operates in Half Line Prefetch mode during which only 16 bytes of data is fetched at a time. The same test for sequential access is repeated, and if access is recognized, the Sequential Access Flag is set and the control switches to Full Line Prefetch mode.

6.2 Buffer Management Summary

Table 8 shows Line Buffer for different cycles. Note that the first three columns together define the cycles that may trigger buffer activity.

Table 8. Buffer Management Summary

Master (Origin)	Cycle Type	Slave (Destination)	Line Buffer Data in Write State	Line Buffer Data in Read State
PCI	Memory Read	EISA	Flush	No Action
PCI	Memory Write	EISA	No Action	Invalidate
PCI	I/O Read	EISA	Flush	No Action
PCI	I/O Write	EISA	No Action	Invalidate
PCI	Interrupt Acknowledge	PCEB/ESC	Flush	No Action
PCI	Configuration Cycle	PCEB Registers	No Action	No Action
PCI	Memory Read/Write	PCI	No Action	No Action
PCI	I/O Read/Write	PCI	No Action	No Action
EISA	Bus Ownership Change	—	Flush	No Action
EISA	Memory Read/Write	EISA	No Action	No Action

Table 8. Buffer Management Summary (Continued)

Master (Origin)	Cycle Type	Slave (Destination)	Line Buffer Data in Write State	Line Buffer Data in Read State
EISA	Memory Read/Write	PCI	(Note 1)	(Note 1)
EISA	I/O Read/Write	EISA	No Action	No Action
EISA	I/O Read/Write	PCI	Flush	Invalidate
ESC's I/O APIC	APIC Bus Message Transfer	Local APIC	Flush	No Action

NOTES:
1. Change from write to read operation or from read to write causes the Line Buffers to be flush or invalidate, respectively.
2. LOCKed cycles (both from PCI and EISA) are not buffered within the PCEB. They are processed using the bypass path.

7.0 EISA INTERFACE

The PCEB provides a fully EISA Bus compatible master and slave interface. This interface provides address and data signal drive capability for eight EISA slots and supports the following types of cycles:

- PCI-initiated memory and I/O read/write accesses to an EISA/ISA device.
- EISA/ISA/DMA-initiated memory and I/O read/write accesses to a PCI device (i.e. via the Line Buffers, if necessary).
- Accesses contained within the EISA Bus (only data swap buffers involved).

For transfers between the EISA Bus and PCI Bus, the PCEB translates the bus protocols. For PCI master-initiated cycles to the EISA Bus, the PCEB is a slave on the PCI Bus and a master on the EISA Bus. For EISA master-initiated cycles to the PCI Bus, the PCEB is a slave on the EISA Bus and a master on the PCI Bus.

NOTE:

1. The PCEB is not involved in refresh cycles on the EISA Bus. When the REFRESH# signal is asserted, the PCEB disables EISA Bus address decoding.
2. Wait state generation on the EISA Bus is performed by the ESC. ISA memory slaves (8 or 16 bits) and ISA I/O slaves can shorten their default or standard cycles by asserting the NOWS# signal line. It is the responsibility of the ESC to shorten these cycles when NOWS# is asserted. Note that ISA I/O 16-bit devices can shorten their cycles by asserting NOWS#. If CHRDY and NOWS# are driven low during the same cycle, NOWS# will not be used and wait states are added as a function of CHRDY. For more details on the wait state generation and the NOWS# signal, refer to the ESC data sheet.
3. All locked PCI cycles (PLOCK# asserted) destined to the EISA Bus are converted to EISA locked cycles using the LOCK# signal protocol. The PCEB is a locked resource during these cycles and maintains control of the EISA Bus until the locked PCI sequence is complete.
4. All locked EISA cycles (LOCK# asserted) destined to PCI are converted to PCI locked cycles using the PLOCK# signal protocol. The PLOCK# signal remains active as long as the EISA LOCK# signal is asserted.

5. The PCEB contains EISA data swap buffers for data size translations between mismatched PCI Bus and EISA Bus transfers and between mismatched devices contained on the EISA Bus. Thus, if data size translation is needed, the PCEB is involved in cycles contained to the EISA Bus, even if the PCEB is neither the master or slave. For data size translation operations, see Section 8.0, EISA Data Swap Buffers.
6. For ISA master cycles to PCI memory or I/O, the ESC translates the ISA signals to EISA signals. The PCEB, as an EISA slave, forwards the cycle to the PCI Bus.
7. For ISA master cycles to ISA/EISA slaves, the PCEB is not involved, except when the cycle requires data size translations. See the ESC data sheet for cycles that are contained within the EISA Bus (i.e., EISA-to-EISA, EISA-to-ISA, ISA-to-ISA, and ISA-to-EISA device cycles).
8. In this section, LA[31:24]# and LA[23:2] are collectively referred to as LA[31:2].

7.1 PCEB As An EISA Master

The PCEB is an EISA master for PCI-initiated cycles targeted to the EISA Bus. When the PCEB decodes the PCI cycle as a cycle destined to the EISA Bus (via subtractive or negative (82374SB only) decoding, as described in Section 4.0, Address Decoding), the PCEB becomes a slave on the PCI Bus. If the PCEB owns the EISA Bus, the cycle is forwarded to the EISA/ISA device. If the PCEB does not own the EISA Bus (EISAHOLDA is asserted to the ESC), the PCI master is retried and the PCEB issues an EISA Bus request to the ESC. For PCI-to-EISA I/O and memory read/write accesses, the PCEB runs standard EISA Bus cycles.

When cycles are forwarded to a matched EISA/ISA slave, the PCEB is the EISA master and controls the transfer until the cycle is terminated. For mismatched cycles to an EISA/ISA slave, the PCEB backs off the EISA Bus as described in Section 7.1.3, Back-Off Cycle.

7.1.1 STANDARD EISA MEMORY AND I/O READ/WRITE CYCLES

The standard EISA cycle completes one transfer each two BCLK periods (zero wait states). The standard EISA memory or I/O cycle begins when the PCEB presents a valid address on LA[31:2] and drives M/IO# high for a memory cycle and low for an I/O cycle. The address can become valid at the end of the previous cycle to allow address pipelining. The EISA slave decodes the address and asserts the appropriate signals to indicate the type of slave and whether it can perform any special timings. The slave asserts EX32# or EX16# to indicate support of EISA cycles.

For extended cycles, the EISA slave introduces wait states using the EXRDY signal. Wait states allow a slower slave to get ready to complete the transfer. The slave negates EXRDY after it decodes a valid address and samples START# asserted. The slave may hold EXRDY negated for a maximum of 2.5 μs to complete a transfer, and must release EXRDY synchronous to the falling edge of BCLK to allow a cycle to complete. Note that the PCEB, as an EISA master, never introduces wait states.

Figure 25 shows three data transfer cycles between an EISA master and an EISA slave. The first transfer is an extended transfer (EXRDY negated), followed by two standard cycles. For PCI cycles that are forwarded to the EISA Bus, the PCEB is the EISA master. The PCEB asserts START# to indicate the start of a cycle. The PCEB also drives W/R# to indicate a read or write cycle and BE[3:0]# to indicate the active bytes. The LA[31:2] and the BE[3:0] remain valid until after the negation of START#. A slave that needs to latch the address does so on the trailing edge of START#.

The ESC asserts CMD# simultaneously with the negation of START# to control data transfer to or from the slave. If a read cycle is being performed, the slave presents the requested data when CMD# is asserted and holds it valid until CMD# is negated by the ESC. For a write cycle, the PCEB presents the data prior to the assertion of CMD# and the slave latches it on or before the trailing edge of CMD#.

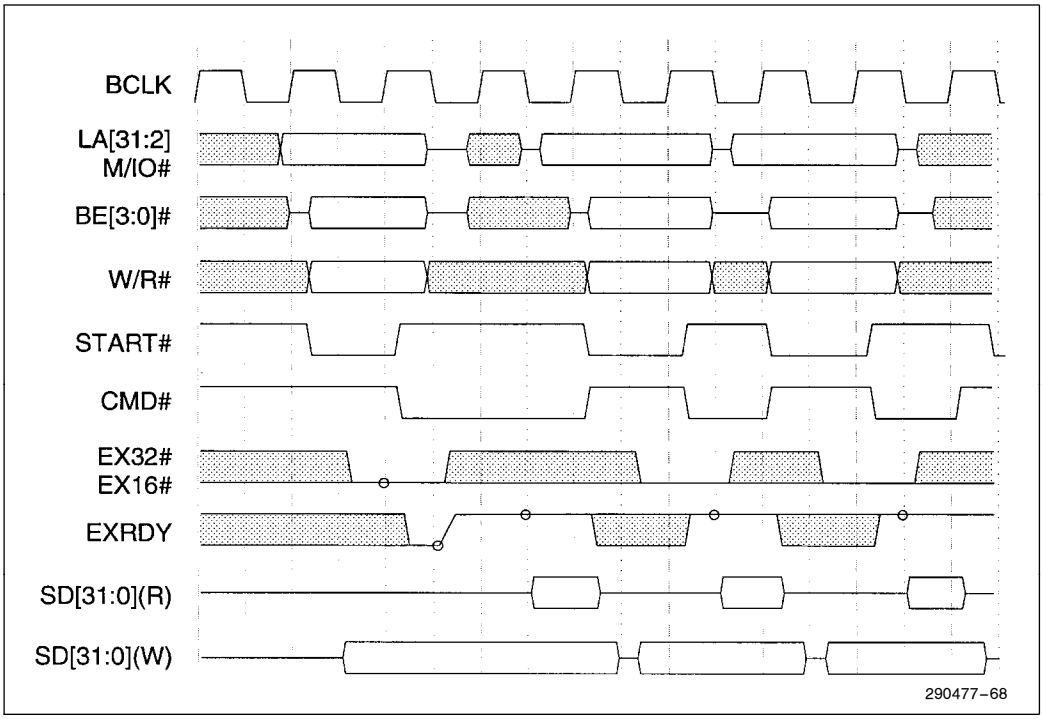


Figure 25. EISA Memory and I/O Read/Write Cycle (One Extended and Two Standard Cycles)

7.1.2 EISA BACK-OFF CYCLE

For mismatched cycles to an EISA/ISA slave, the PCEB, as a master, backs off the EISA Bus by floating the START#, BE[3:0]# and SD[31:0] signals one and half BCLKs after START# has been asserted. The ESC controls the EISA Bus for the duration of the cycle. This allows the ESC to perform data translation, if necessary. At the end of the cycle, the ESC transfers control back to the PCEB by asserting EX16# and EX32# on the falling edge of BCLK, before the rising edge of BCLK that the last CMD# is negated. Refer to the ESC data sheet for further details on master back-off and the cycle transfer control operations.

Figure 26 shows an example of a back-off sequence during a 32-bit EISA master to 16-bit EISA slave Dword read and write operation. The thick lines indicate the change of control between the master and the ESC.

PCEB Reading From a 16-bit EISA Slave

As a 32-bit EISA master, the PCEB begins by placing the address on LA[31:2] and driving M/IO#. The 16-bit EISA slave decodes the address and asserts EX16#. The PCEB asserts START#, W/R#, and BE[3:0]#. The ESC samples EX32# and EX16# on the rising edge of BCLK following the assertion of START# and asserts CMD#. At the same time, the PCEB negates START# and samples EX32#. When EX32# is sampled negated, the PCEB floats START# and BE[3:0]#. Note that, the PCEB continues to drive a valid address on LA[31:0].

The ESC negates CMD# after one BCLK period unless the slave adds wait states (negates EXRDY). The ESC latches SD[15:0] into the PCEB's data swap buffer on the trailing edge of CMD#. The ESC controls the PCEB data swap buffers via the PCEB/ESC Interface. The ESC then asserts START# and presents BE[3:0] (upper word enabled). The ESC negates START# and asserts CMD#. The slave latches the address on the trailing edge of START# and presents data on SD[15:0]. The ESC negates CMD# after one BCLK, unless the slave negates EXRDY. The ESC latches SD[15:0] into the PCEB data swap buffers on the trailing edge of CMD# and instructs the PCEB data swap buffer to copy D[15:0] to D[31:0] and asserts EX32#. Note that, since the transfer is intended for the PCEB, the data is not re-driven back out onto the EISA Bus. The ESC floats the START# and BE[3:0]#. The PCEB regains control of the EISA Bus after sampling EX32# and EX16# asserted.

PCEB Writing To a 16-bit EISA Slave

As a 32-bit EISA master, the PCEB begins by placing the address on LA[31:2] and driving M/IO#. The 16-bit EISA slave decodes the address and asserts EX16#. The PCEB asserts START#, W/R#, BE[3:0]#, and SD[31:0]. The ESC samples EX32# and EX16# on the rising edge of BCLK following the assertion of START# and asserts CMD#. At the same time, the PCEB negates START# and samples EX32#. When EX32# is sampled negated, the PCEB floats START#, SD[31:0], and BE[3:0]#. The data is latched in the PCEB's data swap buffers. Note that the PCEB continues to drive a valid address on LA[31:2].

The ESC instructs the PCEB to drive the data out on SD[31:0] and asserts CMD# after sampling EX32# negated. The slave may sample SD[15:0] while CMD# is asserted. The ESC negates CMD# after one BCLK, unless the slave adds wait states (negates EXRDY). The ESC then presents BE[3:0] (upper word enabled) and asserts START#. The ESC instructs the PCEB to copy SD[31:0] to SD[15:0], negates START# and asserts CMD#. The ESC negates CMD# after one BCLK, unless the slave negates EXRDY. The slave latches the address on the trailing edge of START# and samples SD[15:0] on the trailing edge of CMD#. The ESC returns control of the EISA Bus to the PCEB by floating BE[3:0]# and START#, then asserting EX32#. The PCEB samples EX32# and EX16# asserted on the rising edge of BCLK.

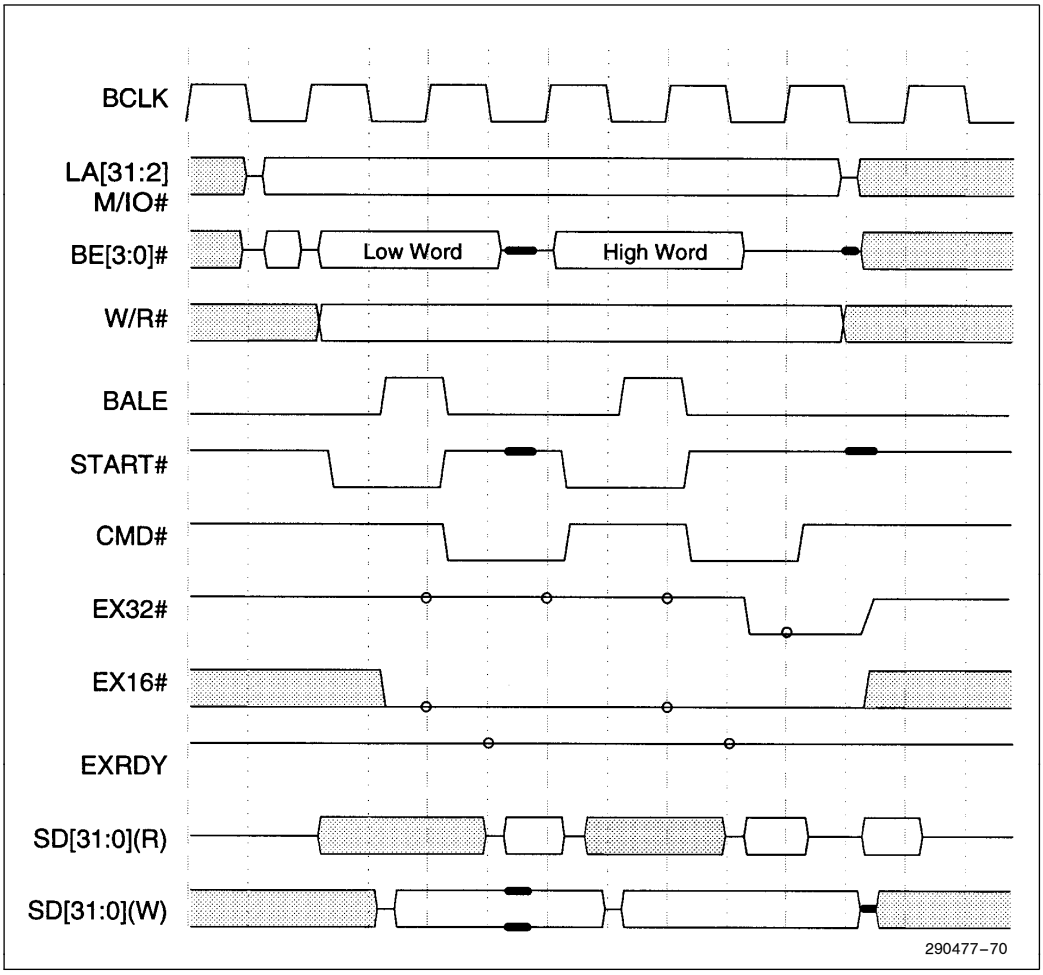


Figure 26. EISA Back-Off Cycle

7.2 PCEB As An EISA Slave

The PCEB is an EISA slave for EISA/ISA/DMA-initiated cycles targeted to the PCI Bus. If the PCEB positively decodes the address (access to one of the EISA programmed main memory segments or access to one of the programmable EISA-to-PCI memory or I/O regions), the PCEB becomes an EISA slave and the cycle is forwarded to the PCI Bus. If the PCEB does not positively decode the address, the cycle is contained to the EISA Bus. For cycles contained to the EISA Bus (i.e., EISA-to-EISA, EISA-to-ISA, ISA-to-ISA, and ISA-to-EISA device cycles), the PCEB is only involved when data size translation is needed.

The PCEB responds as a 32-bit EISA slave. If the EISA master size is not 32 bits, the cycle is a mismatch and invokes data size translation. For details on data size translation, refer to Section 8.0, EISA Data Swap Buffers.

All EISA master memory read cycles to PCI memory start as extended cycles, unless the cycle triggers a read hit to one of the four Line Buffers. If the data is available in the Line Buffers, the PCEB supplies the data to the EISA master without adding wait states. Otherwise, the cycle is extended (wait states added via EXRDY) until the data is available. Note that for non-buffered accesses, the EISA cycle is always extended until data is available from the PCI Bus.

If the Line Buffers are enabled, write cycles to PCI memory are posted in the Line Buffers. If the write can be immediately posted, wait states are not generated on the EISA Bus. Otherwise, the cycle is extended (via wait states) until the data can be posted. Note that writes can be posted to available Line Buffers concurrently with other Line Buffers being flushed to the PCI Bus.

All EISA master I/O read/write accesses to PCI I/O space are non-buffered and always start as extended cycles. Data transfer on the EISA Bus occurs when the requested data is available from the PCI Bus.

For mismatched cycles to the PCEB, the EISA/ISA master backs off the EISA Bus as described in Section 7.1.3, Back-Off Cycle.

7.2.1 EISA MEMORY AND I/O READ/WRITE CYCLES

The standard EISA cycle completes one transfer each two BCLK periods (zero wait states). The standard EISA memory or I/O cycle begins with the EISA master presenting a valid address on LA[31:2] and driving M/IO# high for a memory cycle and low for an I/O cycle. The address can become valid at the end of the previous cycle to allow address pipelining. When the PCEB positively decodes the address, it asserts EX32# to indicate 32-bit support. For memory cycles, the PCEB also asserts SLBURST# to indicate support for burst transfers.

For extended cycles, the PCEB introduces wait states using the EXRDY signal. The PCEB may hold EXRDY negated for a maximum of 2.5 μs to complete a transfer, and releases EXRDY synchronous to the falling edge of BCLK to allow a cycle to complete.

Figure 27 shows three data transfers between an EISA master and an EISA slave. The first transfer is an extended transfer (EXRDY negated), followed by two standard cycles. For EISA cycles that are forwarded to the PCI Bus, the PCEB is an EISA slave. The EISA master asserts START# to indicate the start of a cycle. The EISA master also drives W/R# to indicate a read or write cycle and BE[3:0]# to indicate the active bytes. The LA[31:2] and the BE[3:0] remain valid until after the negation of START#. The PCEB latches the address on the trailing edge of START#.

The ESC asserts CMD# simultaneously with the negation of START# to control data transfer to or from the PCEB. If a read cycle is being performed, the PCEB presents the requested data when CMD# is asserted and holds it valid until CMD# is negated by the ESC. For a write cycle, the EISA master must present the data prior to the assertion of CMD# and the PCEB latches it on the trailing edge of CMD#.

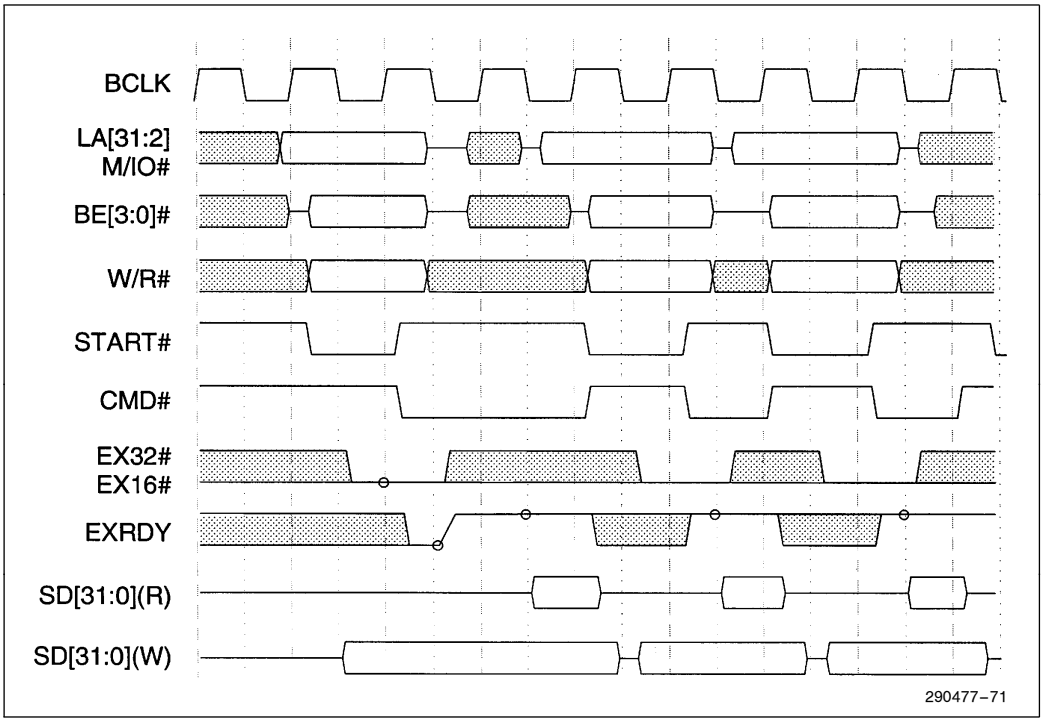


Figure 27. EISA Memory and I/O Read/Write Cycles (One Extended and Two Standard Cycles)

7.2.2 EISA MEMORY BURST CYCLES

The EISA burst cycles permit a continuous sequence of read or write cycles in zero wait-states (1 BCLK per transfer). A burst transfer is either all reads or all writes. Mixed cycles are not allowed. As an EISA slave, the PCEB supports burst memory reads and burst memory writes from/to its Line Buffers. Figure 28 shows an example of a burst sequence for both memory reads and writes on the EISA Bus. During the particular burst sequence, five data transfers occur with a wait state added on the third data transfer.

The first transfer in a burst transfer begins like the standard cycle described above. The EISA master presents a valid address on LA[31:2]. The PCEB, after decoding the address and M/IO#, responds by asserting SLBURST#. The EISA master must sample SLBURST# on the rising edge of BCLK at the trailing edge of START#. The EISA master asserts MSBURST# on the falling edge of BCLK and presents a second address to the PCEB. The ESC holds CMD# asserted while the burst is being performed. If MSBURST# is not asserted by the master, the cycle is run as a standard cycle.

If the cycle is a burst read, the EISA master presents burst addresses on the falling edge of every BCLK. The PCEB presents the data for that address, which is sampled one and half BCLKs later. If the cycle is a burst write, the EISA master presents the data on the rising edge of BCLK, a half cycle after presenting the address. The PCEB samples memory write data on the rising BCLK edge when CMD# is asserted (regardless of the state of MSBURST#). The EISA master terminates the burst cycles by negating MSBURST# and completing the last transfer.

To add wait states during a burst sequence, the PCEB negates EXRDY before the falling edge of BCLK (with CMD# asserted). The EISA master samples EXRDY on the falling edge of BCLK and extends the cycle until EXRDY is asserted. The EISA master can still change the next address even though EXRDY is negated.

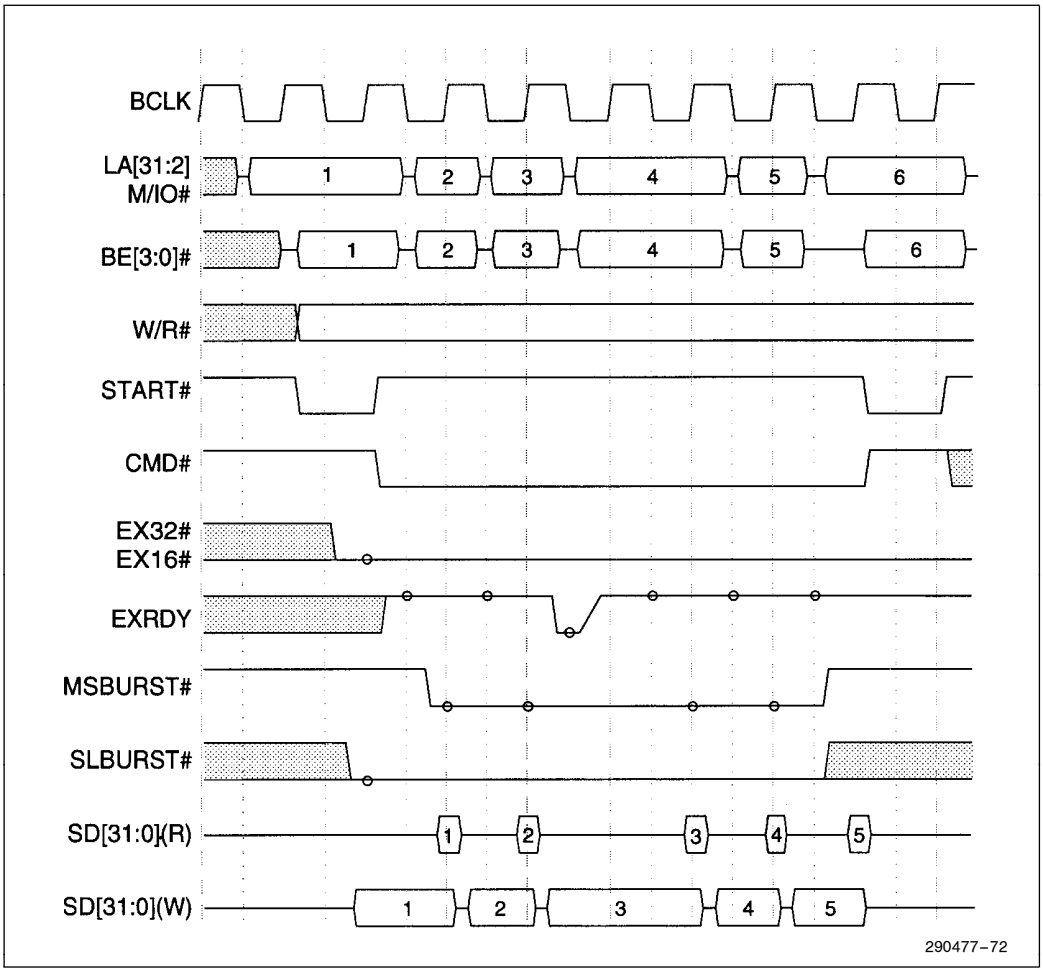


Figure 28. EISA Burst Cycle

7.3 I/O Recovery

The I/O recovery mechanism in the PCEB guarantees a minimum amount of time between back-to-back 8-bit and 16-bit PCI cycles to ISA I/O slaves. Delay times (in BCLKs) for 8-bit and 16-bit cycles are individually programmed via the IORT Register. Accesses to an 8-bit device followed by an access to a 16-bit device use the 8-bit recovery time. Similarly, accesses to a 16-bit device followed by an access to an 8-bit device use the 16-bit recovery time. The PCEB cycles to EISA I/O, DMA cycles, and EISA/ISA bus masters to I/O slaves do not require any delay between back-to-back I/O accesses.

Note that I/O recovery is only required for ISA I/O devices. However, since the PCEB does not distinguish between 8-bit ISA and 8-bit EISA, the delay is also applied to 8-bit EISA I/O accesses (i.e. the ESC).

8.0 EISA DATA SWAP BUFFERS

The PCEB contains a set of buffers/latches that perform data swapping and data size translations on the EISA Bus when the master and slave data bus sizes do not match (e.g., 32-bit EISA master accessing a 16-bit EISA slave). During a data size translation, the PCEB performs one or more of the following operations, depending on the master/slave type (PCI/EISA/ISA), transfer direction (read/write), and the number of byte enables active (BE[3:0]#):

- Data assembly or disassembly
- Data copying (up or down)
- Data re-drive

These operations are described in this section. An example is provided in Section 8.3, The Re-Drive Operation, that shows a cycle where all three functions are used.

The PCEB performs data size translations on the EISA Bus using the data swap buffer control signals generated by the ESC. These signals are described in Section 10.0, PCEB/ESC Interface.

8.1 Data Assembly And Disassembly

The data assembly/disassembly process occurs during PCI, EISA/ISA, and DMA cycles when the master data size is greater than the slave data size. For example, if a 32-bit PCI master is performing a 32-bit read cycle to an 8-bit ISA slave, the ESC intervenes and performs four 8-bit reads. The data is assembled in the PCEB (Figure 29). Once assembled, the PCEB transfers the data as a single Dword to the 32-bit PCI master during the fourth cycle. For a 32-bit write cycle, the PCEB disassembles the Dword by performing four write cycles to the slave. The actual number of cycles required to perform an assembly/disassembly process and make a transfer is a function of the number of bytes (BE[3:0]#) requested and the master/slave size combination.

During EISA master assembly/disassembly transfers, cycle control is transferred from the master to the ESC. The master relinquishes control by backing off the bus (i.e., by floating its START#, BE[3:0], and SD[31:0] signals on the first falling edge of BCLK after START# is negated). The ESC controls the assembly/disassembly process in the PCEB via the data swap buffer control signals on the PCEB/ESC interface. At the end of the assembly/disassembly process, cycle control is transferred back to the bus master (by the ESC asserting EX16# and EX32#). An additional BCLK is added at the end of the transfer to allow the exchanging of cycle control to occur. During DMA transfers, cycle control is maintained by the ESC for the entire cycle.

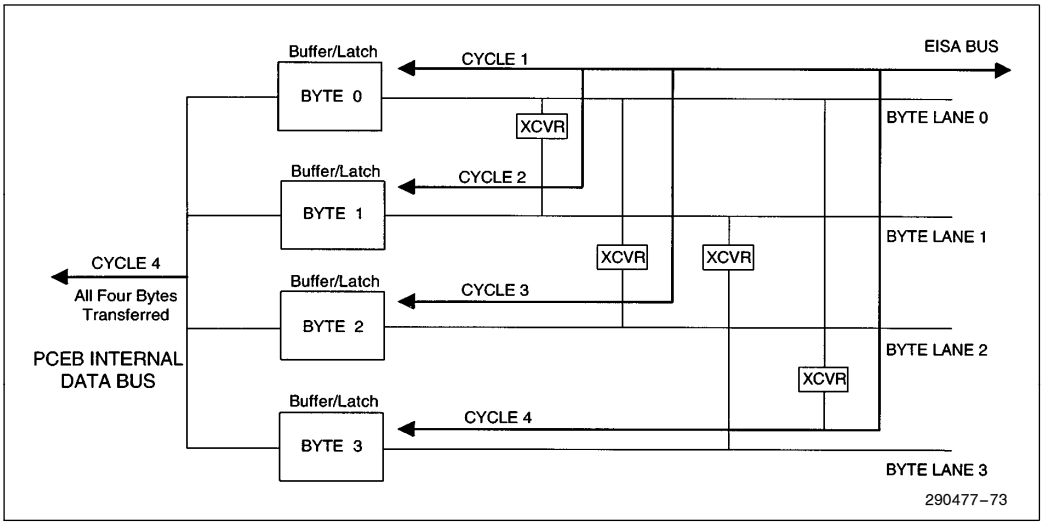


Figure 29. Assembly Function: PCI 32-bit Read from an 8-bit EISA or ISA SlaveBE[3:0]# = 0000

8.2 The Copy Operation (Up Or Down)

The copy operation (Figure 30) is invoked during data transfers between byte lanes. This operation allows the assembly/ disassembly of the data pieces during the cycles between mismatched master/slave combinations. For example, Section 8.1, Data Assembly and Disassembly, describes a 32-bit master read from an 8-bit slave where the data is copied up during the assembly process. Copy-up is used for data assembly and copy-down is used for data disassembly.

The copy-up and copy-down operations are also used during transfers where assembly or disassembly are not required. These transfers are:

- When the master size is smaller than the slave size (e.g. 16-bit EISA master cycle to a 32-bit EISA slave).
- Between a mis-matched master/slave combination when only a byte or a word needs to be transferred (e.g. 32-bit EISA master cycle to an 8-bit ISA slave and only a byte needs to be transferred).

The number of bytes copied up or down is a function of the number of bytes requested (BE[3:0]#) and the master/slave size combinations. During EISA master cycles where the data copying is performed, cycle control is transferred from the bus master to the ESC, except during transfers where the master's data size is smaller than the slave's data size. During DMA transfers, bus control is maintained by the ESC throughout the transfer.

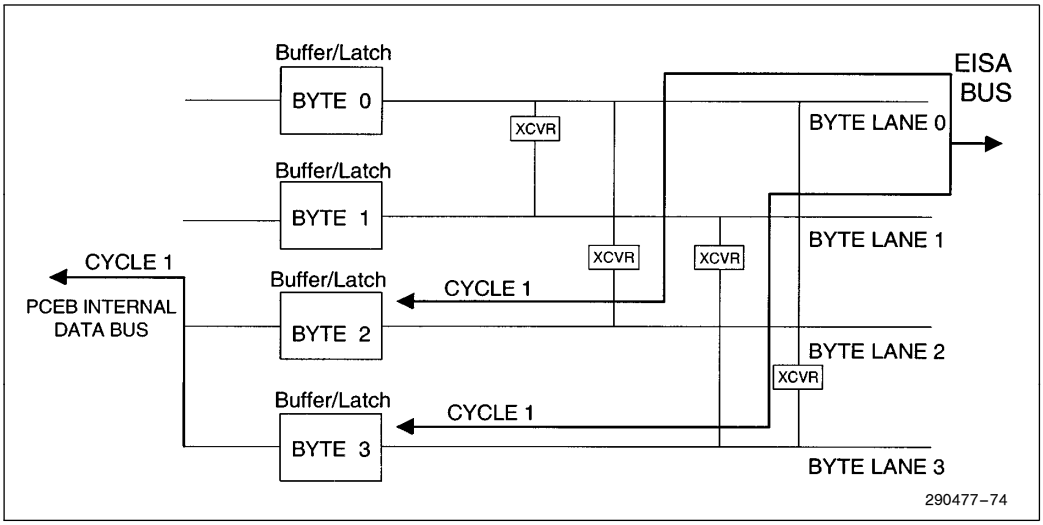


Figure 30. Copy Function: PCI 16-bit Read from a 16-bit EISA or ISA Slave—BE[3:0]# = 0011

8.3 The Re-Drive Operation

The re-drive operation (Figure 31) is used when both the master and the slave, other than PCEB, are on the EISA Bus and the master/slave size combination is mis-matched. Specifically, re-drive occurs:

- during EISA master and DMA cycles (excluding DMA compatible cycles) where the master's data size is greater than the slave's data size.
- during EISA master cycles to ISA slaves where the master/slave match in the size.
- during DMA burst write cycles to a non-burst memory slave.

During a re-drive cycle, the data is latched from the EISA Bus, and then driven back onto the appropriate EISA byte lanes. During a read cycle, the re-drive occurs after the necessary sub-cycles have been completed and the read data has been assembled. For example, when a 32 bit EISA master (other than PCEB) performs 32 bit read from an 8 bit EISA slave, the following sequence of events occurs:

1. The 32-bit EISA master initiates the read cycle. Since the master/slave combination is a mis-match, the master backs off the bus. The EISA master floats its START#, BE[3:0]# and SD[31: 0] lines. The cycle control is then transferred to the ESC.
2. The ESC brings in the first 8 bit data (byte 0) in the first cycle. The ESC asserts SDLE0# to the PCEB.
3. When SDLE0# is asserted, the PCEB latches byte 0 into the least significant byte lane.
4. In the second cycle, the ESC reads the next 8 bit data (byte 1). The PCEB uses SDLE1#, SDCPYUP and SDCPYEN0-1# to latch byte 1 and copy it to the second least significant byte lane (copy-up). This process continues for byte 2 and byte 3. On the fourth cycle, the Dword assembly is complete. During each of the 4 cycles, the ESC generates BE[3:0]# combinations.

5. The ESC instructs the PCEB to re-drive the assembled word to the master by asserting SDOE[2:0]#. In this case, all three SDOE[2:0]# signals are asserted.
6. When SDOE[2:0]# are asserted, the PCEB drives the 32 bit assembled data on SD[31:0] to be latched by the master. The ESC generates the byte enables (BE[3:0]#).
7. The ESC completes the transfer.
8. At the end of the cycle, The ESC transfers control of the EISA Bus back to the EISA master.

During a write cycle, the re-drive occurs after the write data from the master has been latched, and before the data has been disassembled. For example, during a 32-bit write by a 32-bit EISA master to an 8-bit EISA slave, in the first cycle of transfer, the data swap buffers latch the write data (Dword) from the master and drives the first byte back onto the lower byte lane of the EISA Bus. The EISA slave uses the byte enable (BE[3:0]#) combination put out by the EISA master during the first cycle to latch the least significant byte. For the subsequent cycles, the BE[3:0]# combination is generated by the ESC. The PCEB re-drives the second, third and the fourth byte on the second, third and the fourth cycles of the transfer. The number of cycles run is a function of the number of bytes requested (BE[3:0]#), and the master/slave size combinations.

During EISA master and DMA write cycles between master and slave combinations on the EISA/ISA Bus, where only copying is required and no assembly/disassembly is required, the data swap buffer treats this as a re-drive cycle. For example, during a write transfer between a 32-bit EISA master and a 16-bit EISA or ISA slave, where the master is driving data on the upper two byte lanes (BE[3:0]# = 0011), the data swap buffers latch the data on the byte lanes 2 and 3 (Figure 32). The data swap buffers will then re-drive the data onto byte lanes 2 and 3 while copying the data down to byte lanes 0 and 1, for latching by the slave device.

When the PCEB is involved as a master or slave, the re-drive function is disabled. When the PCEB reads 32-bit data from an 8-bit slave the following sequence of events occurs:

1. Same steps as steps 1-4 in the previous example.
2. Once the assembly is complete, the PCEB internally latches the data.
3. The control is transferred back to the PCEB.

NOTE:

During EISA master cycles that require re-driving, the control is transferred from the EISA master to the ESC before the data is re-driven on the data bus. However; during the DMA cycles, the cycle control is maintained by the ESC throughout the entire cycle.

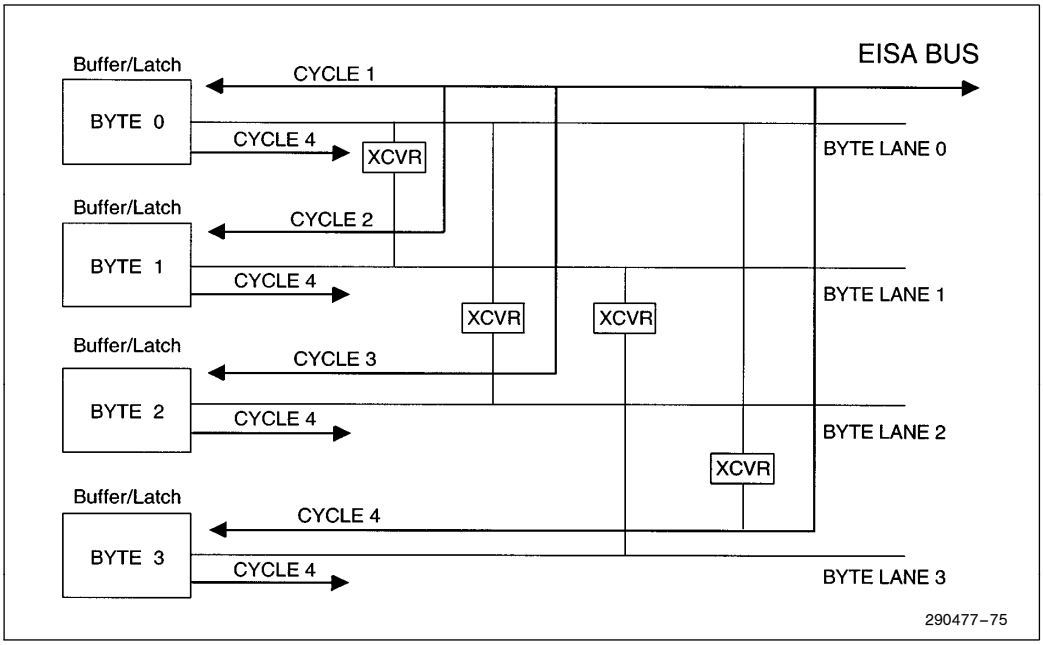


Figure 31. Re-Drive Function: 32-bit EISA Master Accessing an 8-bit EISA or ISA Slave 32-bit Read—BE[3:0]# = 0000

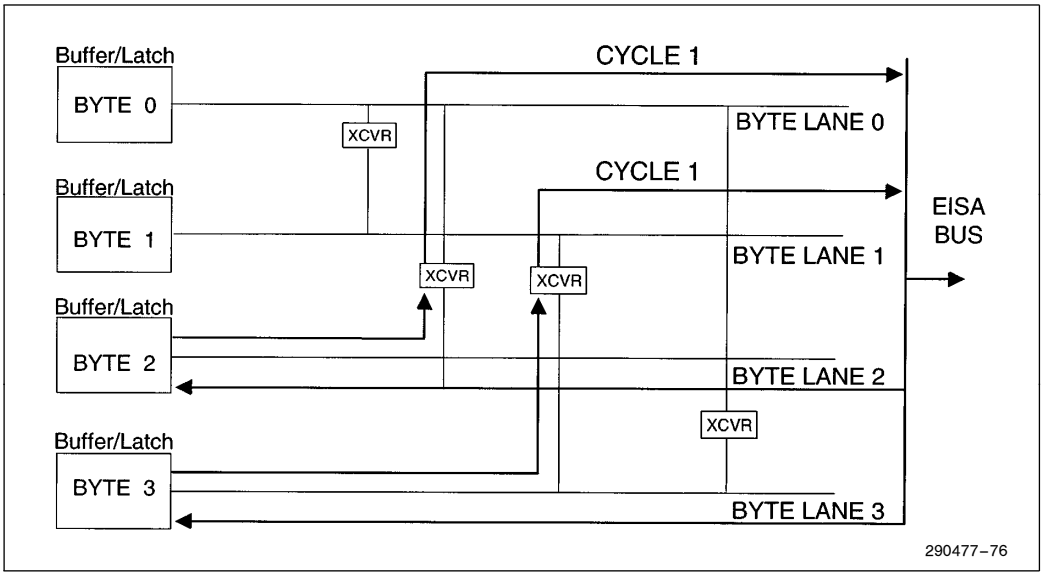


Figure 32. Copy witn Re-Drive: 32-bit EISA Master Accessing a 16-bit EISA or ISA Slave-One Word Write—BE[3:0]# = 0011

9.0 BIOS TIMER

The PCEB provides a system BIOS Timer that decrements at each edge of its 1.03/1.04 MHz clock (derived from the 8.25/8.33 MHz BCLK). Since the state of the counter is undefined at power-up, the BIOS Timer Register must be programmed before it can be used. The timer can be enabled/disabled by writing to the BIOS Timer Address Register.

The BIOS Timer Register can be accessed as a single 16-bit quantity or as 32-bit quantity. For 32-bit accesses, the upper 16 bits are don't care (reserved). The BIOS Timer I/O address location is software programmable. The address is determined by the value programmed into the BTMR Register and can be located on Dword boundaries anywhere in the 64 KByte PCI I/O space.

The BIOS Timer clock has a frequency of 1.03 or 1.04 MHz, depending on the value of BCLK (derived either from 25 MHz or 33 MHz PCICLK). This allows time intervals to be counted from 0 to approximately 65 ms. The accuracy of the counter is $\pm$ 1 μs.

9.1 BIOS Timer Operations

A write operation (either 16-bit or 32-bit) to the BIOS Timer Register initiates the counting sequence. After initialization, the BIOS timer starts decrementing until it reaches zero. When the value in the timer reaches zero, the timer stops decrementing and register value remains at zero until the timer is re-initialized.

After the timer is initialized, the current value can be read at any time. The timer can be re-programmed (new initial value written to the BIOS Timer Register) before the register value reaches zero. All write and read operations to the BIOS Timer Register should include all 16 counter bits. Separate accesses to the individual bytes of the counter must be avoided since this can cause unexpected results (incorrect count intervals).

10.0 PCEB/ESC INTERFACE

The PCEB/ESC interface (Figure 33) provides the inter-chip communications between the PCEB and ESC. The interface provides control information between the two components for PCI/EISA arbitration, data size translations (controlling the PCEB's EISA data swap buffers), and interrupt acknowledge cycles.

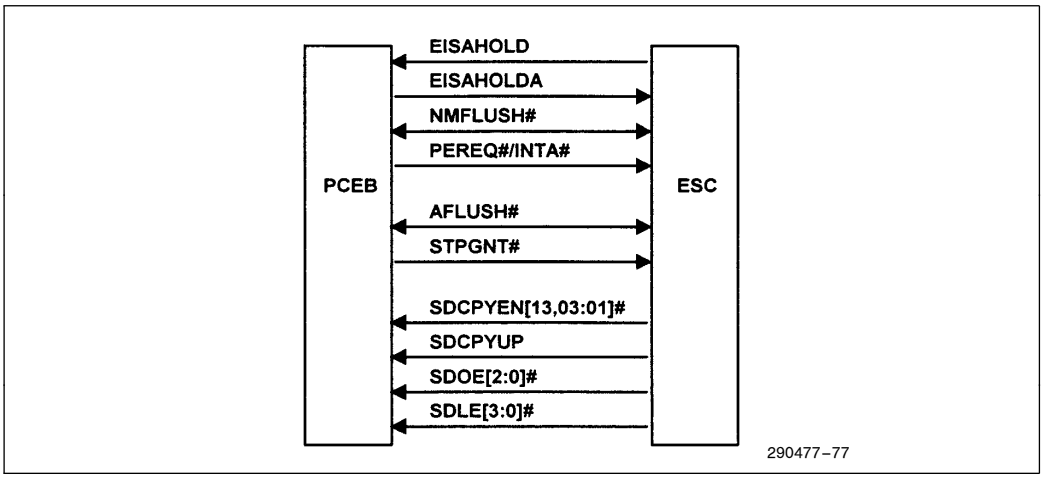


Figure 33. PCEB/ESC Interface Signals

10.1 Arbitration Control Signals

The PCEB contains the arbitration circuitry for the PCI Bus and the ESC contains the arbitration circuitry for the EISA Bus. The PCEB/ESC Interface contains a set of arbitration control signals (EISAHOLD, EISAHOLDA, NMFLUSH#, and PEREQ#/INTA#) that synchronize bus arbitration and ownership changes between the two bus environments. The signals also force PCI device data buffer flushing, if needed, to maintain data coherency during EISA Bus ownership changes.

The PCEB is the default owner of the EISA Bus. If another EISA/ISA master or DMA wants to use the bus, the ESC asserts EISAHOLD to instruct the PCEB to relinquish EISA Bus ownership. The PCEB completes any current EISA Bus transaction, tri-states its EISA Bus signals, and asserts EISAHOLDA to inform the ESC that the PCEB is off the bus.

For ownership changes, other than for a refresh cycle, the ESC asserts the NMFLUSH# signal to the PCEB (for one PCICLK) to instruct the PCEB to flush its Line Buffers pointing to the PCI Bus (Figure 34). The assertion of NMFLUSH# also instructs the PCEB to initiate flushing and to temporarily disable system buffers on the PCI Bus (via MEMREQ#, MEMACK#, and FLSHREQ#). The buffer flushing maintains data coherency, in the event that the new EISA Bus master wants to access the PCI Bus. Buffer flushing also prevents dead-lock conditions between the PCI Bus and EISA Bus. Since the ESC/PCEB do not know ahead of time, whether the new master is going to access the PCI Bus or a device on the EISA Bus, buffers pointing to the PCI Bus are always flushed when there is a change of EISA Bus ownership, except for refresh cycles. For refresh cycles, the ESC controls the cycle and, thus, knows that the cycle is not an access to the PCI Bus and does not initiate a flush request to the PCEB. After a refresh cycle, the ESC always surrenders control of the EISA Bus back to the PCEB.

NMFLUSH# is a bi-directional signal that is negated by the ESC when buffer flushing is not being requested. The ESC asserts NMFLUSH# to request buffer flushing. When the PCEB samples NMFLUSH# asserted, it starts driving the signal in the asserted state and begins the buffer flushing process. (The ESC tri-states NMFLUSH# after asserting it for the initial 1 PCICLK period.) The PCEB keeps NMFLUSH# asserted until all buffers are flushed and then it negates the signal for 1 PCICLK. When the ESC samples NMFLUSH# negated, it starts driving the signal in the negated state, completing the handshake. When the ESC samples NMFLUSH# negated, it grants ownership to the winner of the EISA Bus arbitration (at the time NMFLUSH# was negated). Note that for a refresh cycle, NMFLUSH# is not asserted by the ESC.

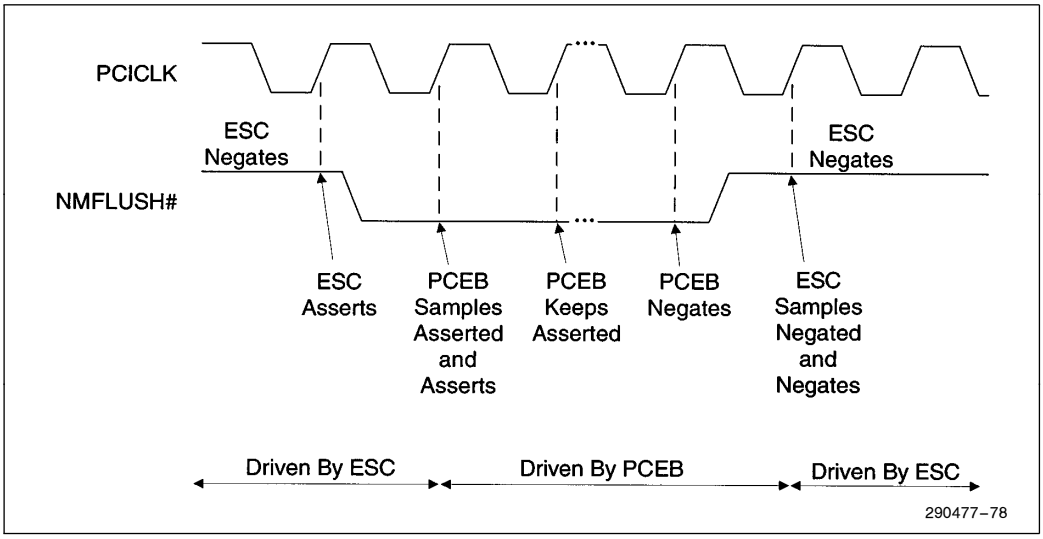


Figure 34. NMFLUSH# Protocol

When the EISA master completes its transfer and gets off the bus (i.e., removes its request to the ESC), the ESC negates EISAHOLD and the PCEB, in turn, negates EISAHOLDA. At this point, the PCEB resumes its default ownership of the EISA Bus.

If a PCI master requests access to the EISA Bus while the bus is owned by a master other than the PCEB, the PCEB retries the PCI cycle and requests ownership of the EISA Bus by asserting PEREQ#/INTA# to the ESC. PEREQ#/INTA# is a dual function signal that is a PCEB request for the EISA Bus (PEREQ# function) when EISAHOLDA is asserted. In response to the PCEB request for EISA Bus ownership, the ESC removes the grant to the EISA master. When the EISA master completes its current transactions and relinquishes the bus (removes its bus request), the ESC negates EISAHOLD and the PCEB, in turn, negates EISAHOLDA. At this point, a grant can be given to the PCI device for a transfer to the EISA Bus. Note that the INTA# function of the PEREQ#/INTA# signal is described in Section 10.3, Interrupt Acknowledge Control.

10.2 System Buffer Coherency Control-APIC

During an interrupt sequence, the system buffers must be flushed before the ESC's I/O APIC can send an interrupt message to the local APIC (CPU's APIC). The ESC and PCEB maintain buffer coherency when the ESC receives an interrupt request for its I/O APIC using the AFLUSH# signal.

10.3 Power Management (82375SB)

In response to the 82375SB ESC's STPCLK# assertion, the CPU sends out a stop grant bus cycle to indicate that it has entered the stop grant state. The PCEB uses the STPGNT# signal to inform the ESC of the stop grant cycle.

10.4 EISA Data Swap Buffer Control Signals

The cycles in the EISA environment may require data size translations before the data can be transferred to its intermediate or final destination. As an example, a 32-bit EISA master write cycle to a 16-bit EISA slave requires a disassembly of a 32-bit Dword into 16-bit words. Similarly, a 32-bit EISA master read cycle to a 16-bit slave requires an assembly of two 16 bit words into a 32-bit Dword. The PCEB contains EISA data swap buffers to support data size translations on the EISA Bus. The operation of the data swap buffers is described in Section 8.0, EISA Data Swap Buffers. The ESC controls the operation of the PCEB's data swap buffers with the following PCEB/ESC interface signals. These signals are outputs from the ESC and an inputs to the PCEB.

- SDCPYEN[13,03:01]#
- SDCPYUP
- SDOE[2:0]#
- SDLE[3:0]#

Copy Enable Outputs (SDCPYEN[13,03:01]#)

These signals enable the byte copy operations between data byte lanes 0, 1, 2 and 3 as shown in the Table 9. ISA master cycles do not perform assembly/disassembly operations. Thus, these cycles use SDCPYEN[13,03:01]# to perform the byte routing and byte copying between lanes. EISA master cycles however, can have assembly/ disassembly operations. These cycles use SDCPYEN[13,03:01]# in conjunction with SDCPYUP and SDLE[3:0]#.

Table 9. Byte Copy Operations

Signal	Copy between Byte Lanes
SDCPYEN01#	Byte 0 (bits[7:0]) and Byte 1 (bits[15:8])
SDCPYEN02#	Byte 0 (bits[7:0]) and Byte 2 (bits[23:16])
SDCPYEN03#	Byte 0 (bits[7:0]) and Byte 3 (bits[31:24])
SDCPYEN13#	Byte 1 (bits[15:8]) and Byte 3 (bits[31:24])

System Data Copy Up (SDCPYUP)

SDCPYUP controls the direction of the byte copy operations. When SDCPYUP is asserted (high), active lower bytes are copied onto the higher bytes. The direction is reversed when SDCPYUP is negated (low).

System Data Output Enable (SDOE[2:0]#)

These signals enable the output of the data swap buffers onto the EISA Bus (Table 10). SDOE[2:0] are re-drive signals in case of mis-matched cycles between EISA to EISA, EISA to ISA, ISA to ISA and the DMA cycles between the devices on EISA.

Table 10. Output Enable Operations

Signal	Byte Lane
SDOE0#	Applies to Byte 0 (bits[7:0])
SDOE1#	Applies to Byte 1 (bits[15:8])
SDOE2#	Applies to Byte 2 and Byte 3 (bits[31:16])

System Data to Internal (PCEB) Data Latch Enables (SDLE[3:0]#)

These signals latch the data from the EISA Bus into the data swap latches. The data is then either sent to the PCI Bus via the PCEB or re-driven onto the EISA Bus. SDLE[3:0]# latch the data from the corresponding EISA Bus byte lanes during PCI Reads from EISA, EISA writes to PCI, DMA cycles between an EISA device and the PCEB. These signals also latch data during mismatched cycles between EISA to EISA, EISA to ISA, ISA to ISA, the DMA cycles between the devices on EISA, and any cycles that require copying of bytes, as opposed to copying and assembly/disassembly.

10.5 Interrupt Acknowledge Control

PEREQ#/INTA# (PCI to EISA Request or Interrupt Acknowledge) is a dual function signal and the selected function depends on the status of EISAHLDA. When EISAHLDA is negated, this signal is an interrupt acknowledge (INTA#) and supports interrupt processing. If interrupt acknowledge is enabled via the PCEB's PCICON Register and EISAHOLDA is negated, the PCEB asserts PEREQ#/INTA# when a PCI interrupt acknowledge cycle is being serviced. This informs the ESC that the forwarded EISA I/O read from location 04h is an interrupt acknowledge cycle. Thus, the ESC uses this signal to distinguish between a request for the interrupt vector and a read of the ESC's DMA register located at 04h. The ESC responds to the read request by placing the interrupt vector on SD[7:0].

11.0 ELECTRICAL CHARACTERISTICS

NOTICE: This data sheet contains information on products in the sampling and initial production phases of development. The specifications are subject to change without notice. Verify with your local Intel Sales office that you have the latest data sheet before finalizing a design.

11.1 Absolute Maximum Ratings

Case Temperature Under Bias . . . . . −65°C to 110°C

Storage Temperature . . . . . −65°C to 150°C

Supply Voltages with
Respect to Ground . . . . . −-0.5V to V_{CC} + 0.5V

Voltage On Any Pin . . . . . −-0.5V to V_{CC} + 0.5V

Power Dissipation (fully loaded) 0.95W

Power Dissipation (four slots) 0.75W

**WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

12.0 PINOUT AND PACKAGE INFORMATION

12.1 Pin Assignment

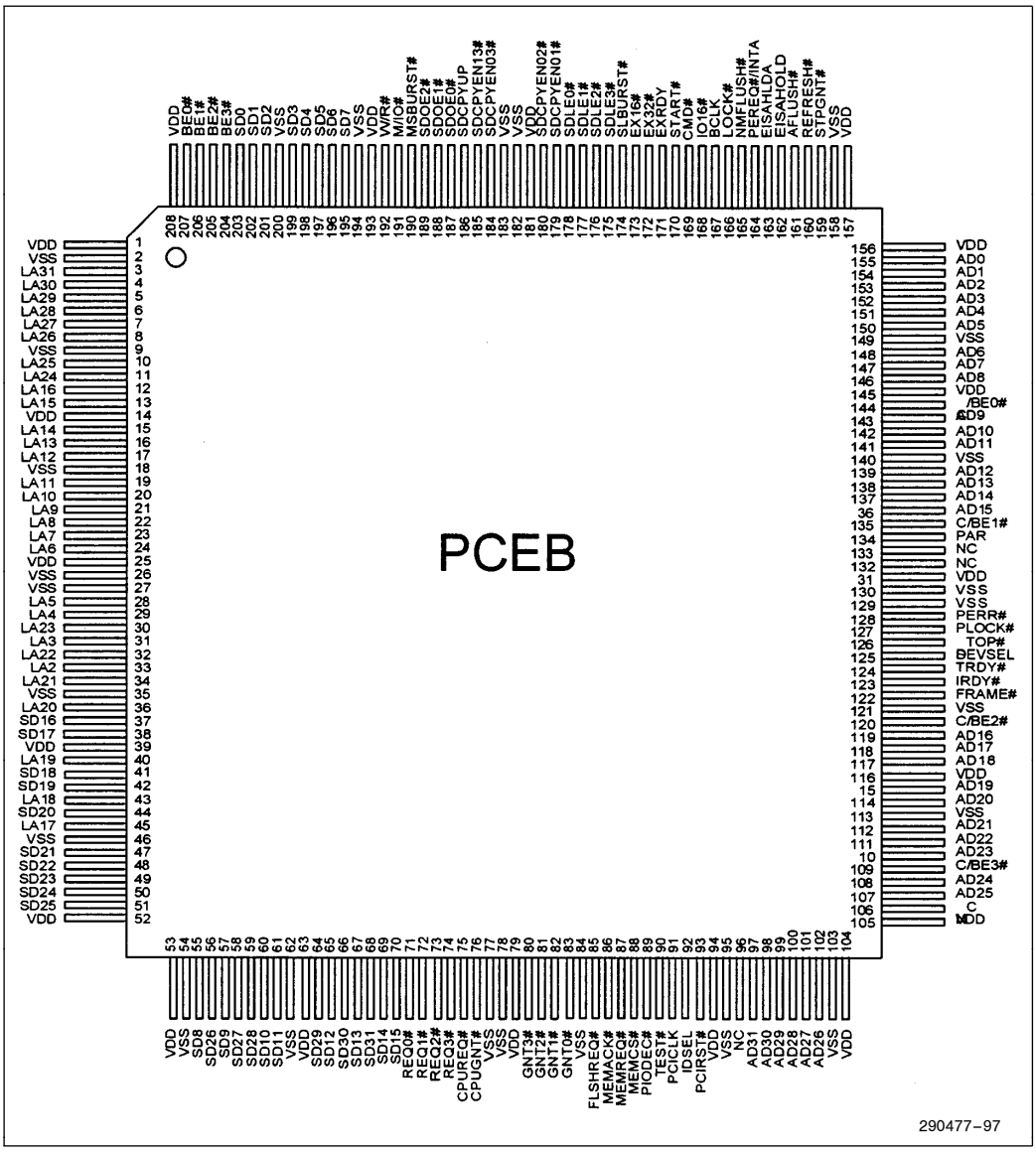


Figure 35. Pinout

Table 11. Alphabetical PCEB Pin Assignment

Name	Pin #	Type
AD0	155	t/s
AD1	154	t/s
AD2	153	t/s
AD3	152	t/s
AD4	151	t/s
AD5	150	t/s
AD6	148	t/s
AD7	147	t/s
AD8	146	t/s
AD9	143	t/s
AD10	142	t/s
AD11	141	t/s
AD12	139	t/s
AD13	138	t/s
AD14	137	t/s
AD15	136	t/s
AD16	119	t/s
AD17	118	t/s
AD18	117	t/s
AD19	115	t/s
AD20	114	t/s
AD21	112	t/s
AD22	111	t/s
AD23	110	t/s
AD24	108	t/s
AD25	107	t/s
AD26	102	t/s
AD27	101	t/s
AD28	100	t/s
AD29	99	t/s
AD30	98	t/s

Name	Pin #	Type
AD31	97	t/s
AFLUSH#	161	t/s
BCLK	167	in
BE0#	207	t/s
BE1#	206	t/s
BE2#	205	t/s
BE3#	204	t/s
C/BE0#	144	t/s
C/BE1#	135	t/s
C/BE2#	120	t/s
C/BE3#	109	t/s
CMD#	169	in
CPUGNT#	76	out
CPUREQ#	75	in
DEVSEL#	125	s/t/s
EISAHLDA	163	out
EISAHOLD	162	in
EX16#	173	in
EX32#	172	o/d
EXRDY	171	o/d
FLSHREQ#	85	out
FRAME#	122	s/t/s
GNT0#	83	out
GNT1#	82	out
GNT2#	81	out
GNT3#	80	out
IDSEL	92	in
IO16#	168	o/d
IRDY#	123	s/t/s
LA2	33	t/s
LA3	31	t/s
LA4	29	t/s

Table 11. Alphabetical PCEB Pin Assignment (Continued)

Name	Pin #	Type
LA5	28	t/s
LA6	24	t/s
LA7	23	t/s
LA8	22	t/s
LA9	21	t/s
LA10	20	t/s
LA11	19	t/s
LA12	17	t/s
LA13	16	t/s
LA14	15	t/s
LA15	13	t/s
LA16	12	t/s
LA17	45	t/s
LA18	43	t/s
LA19	40	t/s
LA20	36	t/s
LA21	34	t/s
LA22	32	t/s
LA23	30	t/s
LA24	11	t/s
LA25	10	t/s
LA26	8	t/s
LA27	7	t/s
LA28	6	t/s
LA29	5	t/s
LA30	4	t/s
LA31	3	t/s
LOCK#	166	t/s
M/IO#	191	t/s
MEMACK#	86	in
MEMCS#	88	out
MEMREQ#	87	out

Name	Pin #	Type
MSBURST#	190	t/s
NC	96	NC
NC	106	NC
NC	132	NC
NC	133	NC
NMFLUSH#	165	t/s
PAR	134	t/s
PCICLK	91	in
PCIRST#	93	in
PEREQ#/INTA#	164	out
PERR#	128	s/t/s
PIODEC#	89	in
PLOCK#	127	s/t/s
REFRESH#	160	in
REQ0#	71	in
REQ1#	72	in
REQ2#	73	in
REQ3#	74	in
SD0	203	t/s
SD1	202	t/s
SD2	201	t/s
SD3	199	t/s
SD4	198	t/s
SD5	197	t/s
SD6	196	t/s
SD7	195	t/s
SD8	55	t/s
SD9	57	t/s
SD10	60	t/s
SD11	61	t/s
SD12	65	t/s
SD13	67	t/s

Table 11. Alphabetical PCEB Pin Assignment (Continued)

Name	Pin #	Type
SD14	69	t/s
SD15	70	t/s
SD16	37	t/s
SD17	38	t/s
SD18	41	t/s
SD19	42	t/s
SD20	44	t/s
SD21	47	t/s
SD22	48	t/s
SD23	49	t/s
SD24	50	t/s
SD25	51	t/s
SD26	56	t/s
SD27	58	t/s
SD28	59	t/s
SD29	64	t/s
SD30	66	t/s
SD31	68	t/s
SDCPYEN01#	179	in
SDCPYEN02#	180	in
SDCPYEN03#	184	in
SDCPYEN13#	185	in
SDCPYUP	186	in
SDLE0#	178	in
SDLE1#	177	in
SDLE2#	176	in
SDLE3#	175	in
SDOE0#	187	in
SDOE1#	188	in
SDOE2#	189	in
SLBURST#	174	t/s
START#	170	t/s

Name	Pin #	Type
STOP#	126	s/t/s
STPGNT#	159	out
TEST#	90	in
TRDY#	124	s/t/s
V_{DD}	1	V
V_{DD}	14	V
V_{DD}	25	V
V_{DD}	39	V
V_{DD}	52	V
V_{DD}	53	V
V_{DD}	63	V
V_{DD}	79	V
V_{DD}	94	V
V_{DD}	104	V
V_{DD}	105	V
V_{DD}	116	V
V_{DD}	131	V
V_{DD}	145	V
V_{DD}	156	V
V_{DD}	157	V
V_{DD}	181	V
V_{DD}	193	V
V_{DD}	208	V
V_{SS}	2	V
V_{SS}	9	V
V_{SS}	18	V
V_{SS}	26	V
V_{SS}	27	V
V_{SS}	35	V
V_{SS}	46	V
V_{SS}	54	V
V_{SS}	62	V

Table 11. Alphabetical PCEB Pin Assignment (Continued)

Name	Pin #	Type
V_{SS}	77	V
V_{SS}	78	V
V_{SS}	84	V
V_{SS}	95	V
V_{SS}	103	V
V_{SS}	113	V
V_{SS}	121	V
V_{SS}	129	V
V_{SS}	130	V

Name	Pin #	Type
V_{SS}	140	V
V_{SS}	149	V
V_{SS}	158	V
V_{SS}	182	V
V_{SS}	183	V
V_{SS}	194	V
V_{SS}	200	V
W/R#	192	t/s

Table 12. Numerical PCEB Pin Assignment

Pin #	Name	Type
1	V_{DD}	V
2	V_{SS}	V
3	LA31	t/s
4	LA30	t/s
5	LA29	t/s
6	LA28	t/s
7	LA27	t/s
8	LA26	t/s
9	V_{SS}	V
10	LA25	t/s
11	LA24	t/s
12	LA16	t/s
13	LA15	t/s
14	V_{DD}	V
15	LA14	t/s
16	LA13	t/s
17	LA12	t/s
18	V_{SS}	V
19	LA11	t/s
20	LA10	t/s
21	LA9	t/s

Pin #	Name	Type
22	LA8	t/s
23	LA7	t/s
24	LA6	t/s
25	V_{DD}	V
26	V_{SS}	V
27	V_{SS}	V
28	LA5	t/s
29	LA4	t/s
30	LA23	t/s
31	LA3	t/s
32	LA22	t/s
33	LA2	t/s
34	LA21	t/s
35	V_{SS}	V
36	LA20	t/s
37	SD16	t/s
38	SD17	t/s
39	V_{DD}	V
40	LA19	t/s
41	SD18	t/s
42	SD19	t/s
43	LA18	t/s

Table 12. Numerical PCEB Pin Assignment (Continued)

Pin #	Name	Type
44	SD20	t/s
45	LA17	t/s
46	V_{SS}	V
47	SD21	t/s
48	SD22	t/s
49	SD23	t/s
50	SD24	t/s
51	SD25	t/s
52	V_{DD}	V
53	V_{DD}	V
54	V_{SS}	V
55	SD8	t/s
56	SD26	t/s
57	SD9	t/s
58	SD27	t/s
59	SD28	t/s
60	SD10	t/s
61	SD11	t/s
62	V_{SS}	V
63	V_{DD}	V
64	SD29	t/s
65	SD12	t/s
66	SD30	t/s
67	SD13	t/s
68	SD31	t/s
69	SD14	t/s
70	SD15	t/s
71	REQ0#	in
72	REQ1#	in
73	REQ2#	in
74	REQ3#	in
75	CPUREQ#	in

Pin #	Name	Type
76	CPUGNT#	out
77	V_{SS}	V
78	V_{SS}	V
79	V_{DD}	V
80	GNT3#	out
81	GNT2#	out
82	GNT1#	out
83	GNT0#	out
84	V_{SS}	V
85	FLSHREQ#	out
86	MEMACK#	in
87	MEMREQ#	out
88	MEMCS#	out
89	PIODEC#	in
90	TEST#	in
91	PCICLK	in
92	IDSEL	in
93	PCIRST#	in
94	V_{DD}	V
95	V_{SS}	V
96	NC	NC
97	AD31	t/s
98	AD30	t/s
99	AD29	t/s
100	AD28	t/s
101	AD27	t/s
102	AD26	t/s
103	V_{SS}	V
104	V_{DD}	V
105	V_{DD}	V
106	NC	NC
107	AD25	t/s

Table 12. Numerical PCEB Pin Assignment (Continued)

Pin #	Name	Type
108	AD24	t/s
109	C/BE3#	t/s
110	AD23	t/s
111	AD22	t/s
112	AD21	t/s
113	V_{SS}	V
114	AD20	t/s
115	AD19	t/s
116	V_{DD}	V
117	AD18	t/s
118	AD17	t/s
119	AD16	t/s
120	C/BE2#	t/s
121	V_{SS}	V
122	FRAME#	s/t/s
123	IRDY#	s/t/s
124	TRDY#	s/t/s
125	DEVSEL#	s/t/s
126	STOP#	s/t/s
127	PLOCK#	s/t/s
128	PERR#	s/t/s
129	V_{SS}	V
130	V_{SS}	V
131	V_{DD}	V
132	NC	NC
133	NC	NC
134	PAR	t/s
135	C/BE1#	t/s
136	AD15	t/s
137	AD14	t/s
138	AD13	t/s
139	AD12	t/s

Pin #	Name	Type
140	V_{SS}	Vv
141	AD11	t/s
142	AD10	t/s
143	AD9	t/s
144	C/BE0#	t/s
145	V_{DD}	V
146	AD8	t/s
147	AD7	t/s
148	AD6	t/s
149	V_{SS}	V
150	AD5	t/s
151	AD4	t/s
152	AD3	t/s
153	AD2	t/s
154	AD1	t/s
155	AD0	t/s
156	V_{DD}	V
157	V_{DD}	V
158	V_{SS}	V
159	STPGNT#	out
160	REFRESH#	in
161	AFLUSH#	t/s
162	EISAHOLD	in
163	EISAHLDA	out
164	PEREQ#/INTA#	out
165	NMFLUSH#	t/s
166	LOCK#	t/s
167	BCLK	in
168	IO16#	o/d
169	CMD#	in
170	START#	t/s
171	EXRDY	o/d

Table 12. Numerical PCEB Pin Assignment (Continued)

Pin #	Name	Type
172	EX32#	o/d
173	EX16#	in
174	SLBURST#	t/s
175	SDLE3#	in
176	SDLE2#	in
177	SDLE1#	in
178	SDLE0#	in
179	SDCPYEN01#	in
180	SDCPYEN02#	in
181	V_{DD}	V
182	V_{SS}	V
183	V_{SS}	V
184	SDCPYEN03#	in
185	SDCPYEN13#	in
186	SDCPYUP	in
187	SDOE0#	in
188	SDOE1#	in
189	SDOE2#	in
190	MSBURST#	t/s

Pin #	Name	Type
191	M/IO#	t/s
192	W/R#	t/s
193	V_{DD}	V
194	V_{SS}	V
195	SD7	t/s
196	SD6	t/s
197	SD5	t/s
198	SD4	t/s
199	SD3	t/s
200	V_{SS}	V
201	SD2	t/s
202	SD1	t/s
203	SD0	t/s
204	BE3#	t/s
205	BE2#	t/s
206	BE1#	t/s
207	BE0#	t/s
208	V_{DD}	V

12.2 Package Characteristics

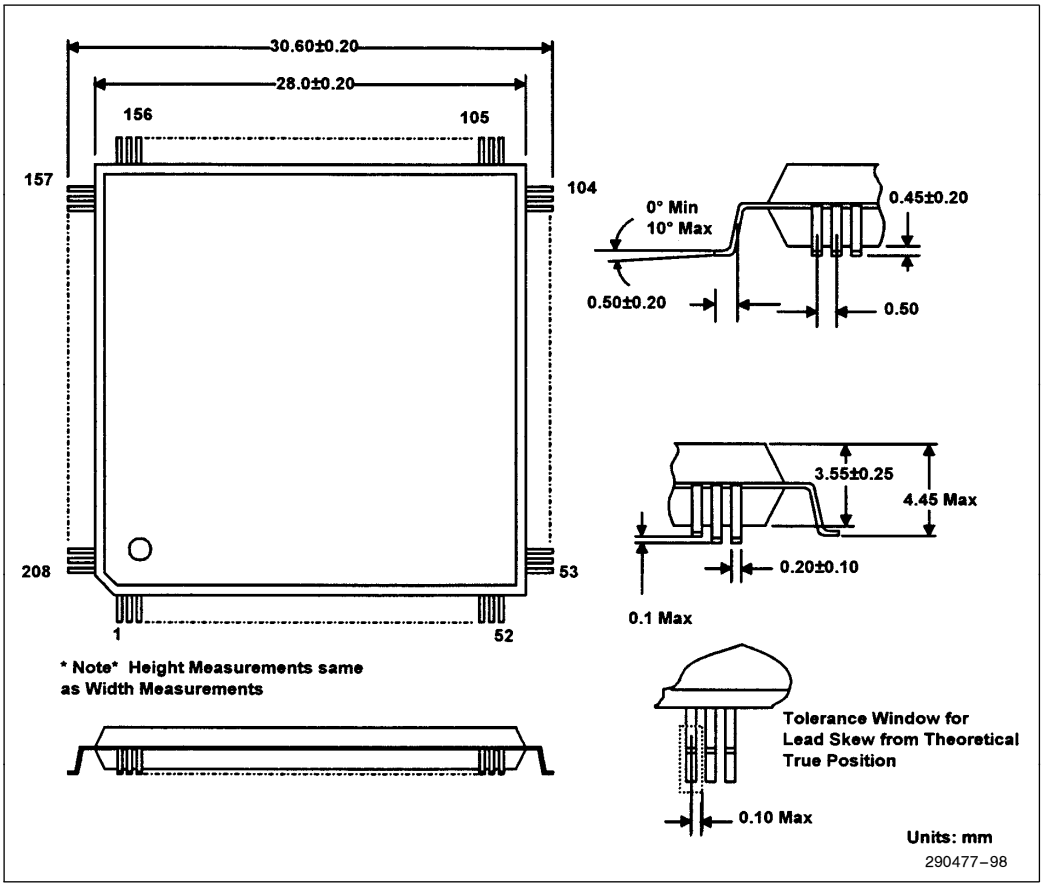


Figure 36. 208-Pin Quad Flat Pack (QFP) Dimensions

13.0 TESTABILITY

13.1 NAND Tree

A NAND Tree is provided primarily for VIL/VIH testing. The NAND Tree is also useful for Automated Test Equipment (ATE) at board level testing. The NAND Tree allows the tester to test the solder connections for each individual signal pin.

The TEST# pin, along with BCLK, PIODEC# and EX16#, activates the NAND Tree. The following combinations of PIODEC#, EX16#, and TEST# causes each buffer to be tri-stated.:

PIODEC# = 1 and EX16# = 0 and TEST# = 0
or
PIODEC# = 0 and EX16# = 1

Care must be taken as the test is in progress to ensure that one of the preceding combinations is valid. Otherwise, the test mode will be exited.

Asserting TEST# causes the output pulse train to appear on the EISAHLDA pin. BCLK must be driven low in order to enable the NAND Tree.

The sequence of the ATE test is as follows:

1. Drive TEST# low, EX16# high, PIODEC# low, and BCLK low.
2. Drive each pin high, except for the pins mentioned in the above discussion (TEST#, PIODEC#, and BCLK).
3. Starting at pin 168 (IO16#) and continuing with pins 169, 170, etc., individually drive each pin low, remembering to toggle PIODEC# from low to high when EX16# is toggled from high to low. Also, when PIODEC# is driven low, EX16# must be driven high. Expect EISAHLDA to toggle after each corresponding input pin is toggled. The final pin in the tree is pin 166 (LOCK#). BCLK is not part of the tree, and EISAHLDA is operated only as an output. Also, note that no-connect (NC), Vcc, and Vxx pins are not part of the NAND Tree.
4. Turn off tester drivers before enabling the PCEB's buffers (via PIODEC#, TEST#, and EX16#).
5. Reset the PCEB prior to proceeding with further testing.

Table 13. NAND Tree Cell Order

Pin #	Name	Pin #	Name	Pin #	Name
168	IO16 #(1)	205	BE2 #	38	SD17
169	CMD #	206	BE1 #	40	LA19
170	START	207	BE0 #	41	SD18
171	EXRDY	3	LA31 #	42	SD19
172	EX32 #	4	LA30 #	43	LA18
173	EX32 #	5	LA29 #	44	SD20
174	SLBURST #	6	LA28 #	45	LA17
175	SDLE3 #	7	LA27 #	47	SD21
176	SDLE2 #	8	LA26 #	48	SD22
177	SDLE1 #	10	LA25 #	49	SD23
178	SDLE0 #	11	LA24 #	50	SD24
179	SDCPYEN01 #	12	LA16	51	SD25
180	SDCPYEN02 #	13	LA15	55	SD8
184	SDCPYEN03 #	15	LA14	56	SD26
185	SDCPYEN13 #	16	LA13	57	SD9
186	SDCPYUP	17	LA12	58	SD27
187	SDOE0 #	18	V_{SS}	59	SD28
188	SDOE1 #	19	LA11	60	SD10
189	SDOE2 #	20	LA10	61	SD11
190	MSBURST #	21	LA9	64	SD29
191	M/IO #	22	LA8	65	SD12
192	W/R #	23	LA7	66	SD30
195	SD7	24	LA6	67	SD13
196	SD6	28	LA5	68	SD31
197	SD5	29	LA4	69	SD14
198	SD4	30	LA23	70	SD15
199	SD3	31	LA3	71	REQ0 #
201	SD2	32	LA22	72	REQ1 #
202	SD1	33	LA2	73	REQ2 #
203	SD0	34	LA21	74	REQ3 #
204	BE3 #	36	LA20	75	CPUREQ #
		37	SD16	76	CPUGNT

Table 13. NAND Tree Cell Order

Pin #	Name	Pin #	Name	Pin #	Name
80	GNT3 #	110	AD23	139	AD12
81	GNT2 #	111	AD22	141	AD11
82	GNT1 #	112	AD21	142	AD10
83	GNT0 #	114	AD20	143	AD9
85	FLSHREQ #	115	AD19	144	C/BE0 #
86	MEMACK #	117	AD18	146	AD6
87	MEMREQ #	118	AD17	147	AD7
88	MEMCS #	119	AD16	148	AD6
89	PIODEC # (3)	120	C/BE2 #	150	AD5
91	PCICLK	122	FRAME #	151	AD4
92	IDSEL	123	IRDY #	152	AD3
93	PCIRST #	124	TRDY #	153	AD2
97	AD31	125	DEVSEL #	154	AD1
98	AD30	126	STOP #	155	AD0
99	AD29	127	PLOCK #	160	REFRESH #
100	AD28	128	PERR #	161	AFLUSH #
101	AD27	134	PAR	162	EISAHOLD
102	AD26	135	C/BE1 #	164	PEREQ # /INTA #
107	AD25	136	AD15	165	NMFLUSH #
108	AD24	137	AD14	166	LOCK #
109	C/BE3 #	138	AD13		

NOTES:
1.Start of NAND Tree.
2.Must be 1 when PIODEC # is 0 and must be 0 when PIODEC # is 1.
3.Must be 0 when EX16 # is 1 and must be 1 when EX16 # is 0.

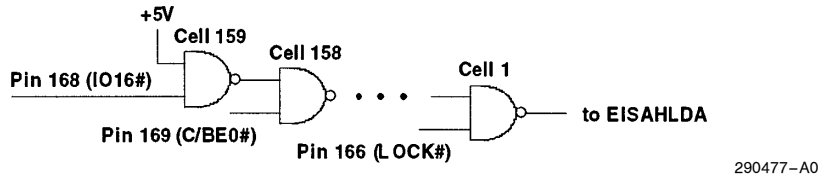


Figure 37. NAND Tree

This datasheet has been download from:

www.datasheetcatalog.com

Datasheets for electronics components.