82420/82430 PCIset ISA and EISA Bridges

intel®

Intel Corporation makes no warranty for the use of its products and assumes no responsibility for any errors which may appear in this document nor does it make a commitment to update the information contained herein.

Intel retains the right to make changes to these specifications at any time, without notice.

Contact your local sales office to obtain the latest specifications before placing your order.

The following are trademarks of Intel Corporation and may only be used to identify Intel products:

376™
Above™
ActionMedia®
BITBUS™
Code Builder™
DeskWare™
Digital Studio™
DVI®
EtherExpress™
ETOX™
ExCA™
Exchange and Go™
FaxBACK®
Grand Challenge™
i®
ICE™
iCOMP™
iLBX™
Inboard™
Indeo™
i287™
i386™
i387™
i486™
i487™
i750®
i860™
i960®
Intel287™
Intel386™
Intel387™
Intel486™
Intel487™
Intel®
intel inside®
Intellec®
iPSC®
iRMX®
iSBC®
iSBX™
iWARP™
LANDesk™
LANPrint®
LANProtect™
LANSelect®
LANShell®
LANSight™
LANSpace®
LANSpool®
MAPNET™
Matched™
MCS®
Media Mail™
NetPort®
NetSentry™
NetSight™
OpenNET™
OverDrive™
Paragon™
Pentium™
ProSolver™
RapidCAD™
READY-LAN™
Reference Point®
RMX/80™
RxServer™
SatisFAXtion®
SmartWire™
SnapIn 386™
Storage Broker™
StorageExpress™
SugarCube™
The Computer Inside™
TokenExpress™
Visual Edge™
WYPIWYF®

MDS is an ordering code only and is not used as a product name or trademark. MDS is a registered trademark of Mohawk Data Sciences Corporation.

CHMOS and HMOS are patented processes of Intel Corp.

Intel Corporation and Intel's FASTPATH are not affiliated with Kinetics, a division of Excelan, Inc. or its FASTPATH trademark or products.

Additional copies of this manual or other Intel literature may be obtained from:

Intel Corporation
Literature Sales
P.O. Box 7641
Mt. Prospect, IL 60056-7641

82420/82430 PCIset
Cache/Memory Subsystem

CONTENTS

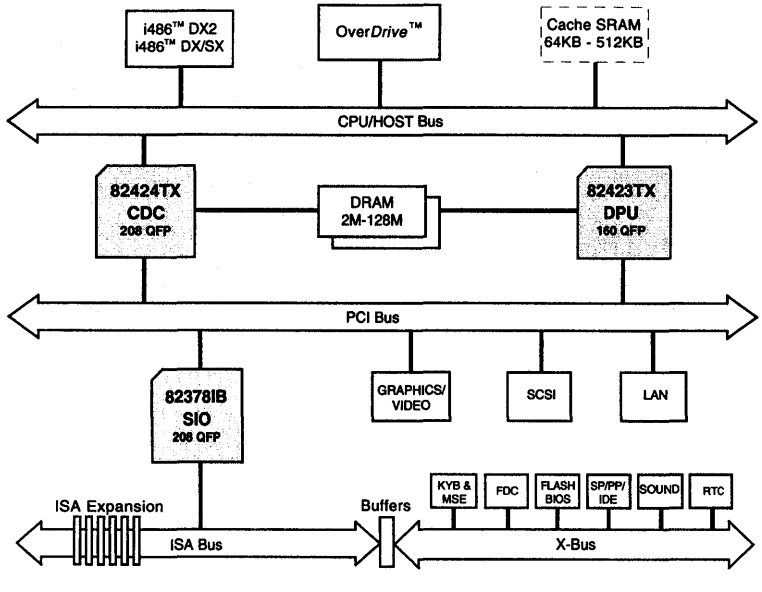
INTEL 82420 PCIset

Intel's 82420 PCIset enables workstation level of performance for Intel486™ CPU desktop systems. The Peripheral Component Interconnect Bus (PCI) is driving a new architecture for PC's—eliminating the I/O bottleneck of standard expansion busses. PCI provides a glueless interface for high performance peripherals such as LAN, SCSI, graphics and video to be placed onto a fast local bus. By utilizing this technology and incorporating read/write bursts along with write buffers into the 82420 PCIset, a new level of PC graphics is now possible for today's Intel486 CPU desktop systems.

The Intel 82420 PCIset is comprised of three components: the 82424TX Cache DRAM Controller (CDC), the 82423TX Data Path Unit (DPU), and the 82378IB System I/O (SIO). The CDC and DPU provide the core system architecture while the SIO is a PCI master/slave agent which bridges the core architecture to the ISA standard expansion bus. Intel also offers two components, the 82374EB (ESC) and 82375EB (PCEB), that work in conjunction to bridge the PCI bus to the EISA expansion bus. Refer to the ESC and PCEB data sheets for information regarding the EISA bridge components.

The chip set supports the Intel486 family as well as the write-back caching capability of Intel's future OverDrive™ processor for the Intel486 DX2. The high performance memory subsystem supports 4-1-2-1 DRAM accesses at 33 MHz and concurrent operation between PCI bus masters while the CPU accesses memory. An integrated second level cache can be programmed for write-through or write-back operation.

Intel 82420 PCIset



290467-1

Intel486 SX, Intel486 DX, Intel486 DX2, and OverDrive are trademarks of Intel Corporation.

April 1993
Order Number: 290467-002

Product Highlights

82424TX—Cache DRAM Controller (CDC)

- Concurrent Linefill during Copyback Cycles
- Supports Intel486 CPU Family and OverDrive Processors
- Supports Future OverDrive Upgrade Processor in Write-Back Cache Mode
- 64K–512K Level 2 Cache Support
- Level 2 Cache Configurable as Write-Back or Write-Through
- 208-Pin QFP Package

82423TX—Data Path Unit (DPU)

- Highly Integrated
- Four Dword Write Buffers
- Zero Wait States for CPU Write Cycles
- PCI Burst Write Capability
- 160-Pin QFP Package

82378IB—System I/O Component (SIO)

- Supports Fast DMA Type A, B, or F Cycles
- Supports DMA Scatter/Gather
- Arbitration Logic for Four PCI Masters
- Reusable across Multiple Platforms
- Directly Drives Six External ISA Slots
- Integrates Many of Today's Common I/O Functions
- 208-Pin QFP Package

Product Description

The 82424TX Cache DRAM Controller (CDC) is a single-chip bridge from the CPU to the PCI bus. It provides the integrated functionality of a second level cache controller, a DRAM controller, and a PCI bus controller. It also features an optimized memory subsystem. The CDC is a dual ported device with one port as the host port and the other as the PCI port.

The 82423TX Data Path Unit (DPU) integrates the host data, memory data, and PCI data interface, DPU control/parity and four deep posted write buffers. With glue and buffers integrated directly into the DPU, the Intel 82420 PCIset reduces board space requirements. The DPU's posted write buffers allow CPU write cycles to be executed as 0 wait states.

The 82378IB System I/O (SIO) is a dual ported device which acts as a bridge between the PCI and standard ISA I/O bus. This component can be used with future Intel PCIsets. The SIO integrates the functionality of an ISA controller, PCI controller, fast 32-bit DMA controller, and standard system I/O functions.

82430 PCIset
FOR THE PENTIUM™ PROCESSOR

- **Supports the Pentium™ Processor at 60 MHz or 66.667 MHz**
- **Interfaces the Host and Standard Buses to the Peripheral Component Interconnect (PCI) Local Bus Operating at 30 MHz or 33.33 MHz**
  - **— Up to 132 Mbytes/sec Transfer Rate**
  - **— Full Concurrency between CPU Host Bus and PCI Bus Transactions**
- **Integrated Cache Controller Provided for Optional Second Level Cache**
  - **— 256 Kbyte or 512 Kbyte Cache**
  - **— Write-Back or Write-Through Policy**
  - **— Standard or Burst SRAM**
- **Integrated Tag RAM for Cost Savings on Second Level Cache**
- **Provides a 64-Bit Interface to DRAM Memory**
  - **— From 2 Mbytes to 192 Mbytes of Main Memory**
  - **— 70 ns and 60 ns DRAMs Supported**
- **Supports the Pipelined Address Mode of the Pentium Processor for Higher Performance**
- **Optional ISA or EISA Standard Bus Interface**
  - **— Single Component ISA Controller**
  - **— Two Component EISA Bus Interface**
  - **— Minimal External Logic Required**
- **Supports Burst Read and Writes of Memory from the Host and PCI Buses**
- **Five Integrated Write Posting and Read Prefetch Buffers Increase CPU and PCI Master Performance**
- **Host CPU Writes to PCI in Zero Wait State PCI Bursts with Optional TRDY# Connection**
- **Integrated Low Skew Host Bus Clock Driver for Cost and Board Space Savings**
- **PCIset Operates Synchronous to the 66.667 MHz CPU and 33.33 MHz PCI Clocks**
- **Byte Parity Support for the Host/PCI and Main Memory Buses**
  - **— Optional Parity on the Second Level Cache**

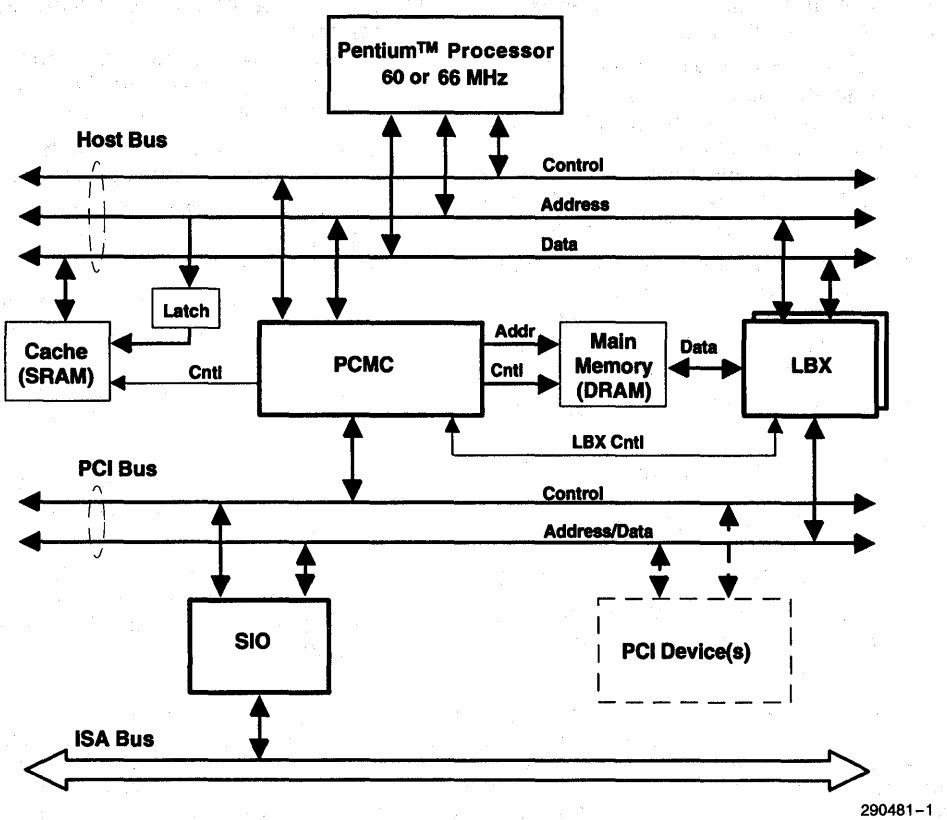
The 82430 PCIset provides the Host/PCI bridge, cache/main memory controller, and an I/O subsystem core (either PCI/EISA or PCI/ISA bridge) for the next generation of high-performance personal computers based on the Pentium microprocessor. System designers can take advantage of the power of the PCI (Peripheral Component Interconnect) bus for the local I/O while maintaining access to the large base of EISA and ISA expansion cards, and corresponding software applications. Extensive buffering and buffer management within the bridges ensures maximum efficiency in all three bus environments (Host CPU, PCI, and EISA/ISA Buses).

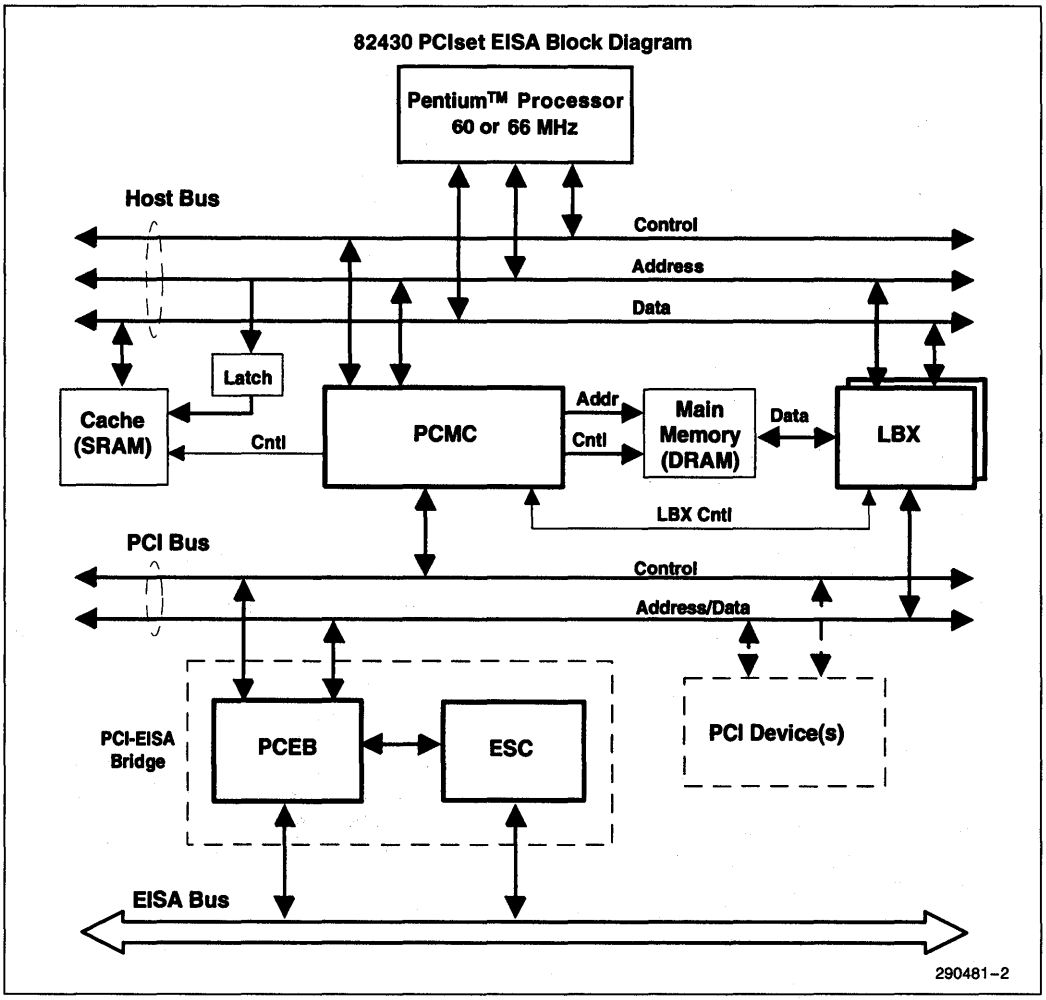
The 82430 PCIset consists of the 82434LX PCI/Cache/Memory Controller (PCMC) and the 82433LX Local Bus Accelerator (LBX) components, plus, either a PCI/ISA bridge or a PCI/EISA bridge. The PCMC and LBX provide the core cache and main memory architecture and serve as the Host/PCI bridge. For an ISA-based system, the 82430 PCIset includes the 82378IB System I/O (SIO) component as the PCI/ISA bridge. For an EISA-based system, the 82430 PCIset includes the 82375EB PCI/EISA Bridge (PCEB) and the 82374EB EISA System Component (ESC). The PCEB and ESC work in tandem to form the complete PCI/EISA bridge. Both the ISA and EISA-based systems are shown on the following pages.

Pentium is a trademark of Intel Corporation.

March 1993
Order Number: 290481-001

82430 PCIset ISA Block Diagram



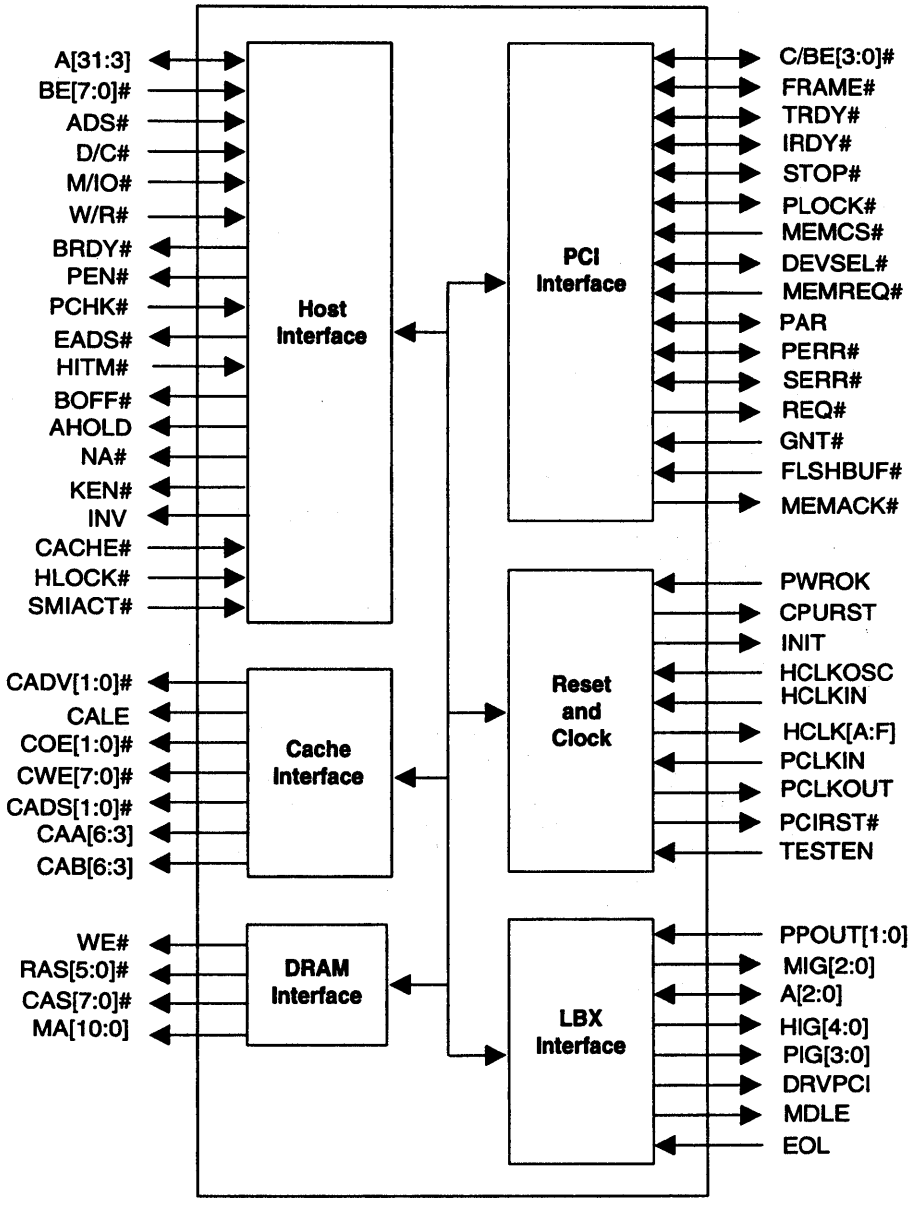

82430 PCIset EISA Block Diagram
Pentium™ Processor
60 or 66 MHz
Host Bus
Control
Address
Data
Latch
Cache
(SRAM)
Cntl
PCMC
Addr
Cntl
Main
Memory
(DRAM)
Data
LBX
LBX Cntl
PCI Bus
Control
Address/Data
PCI-EISA
Bridge
PCEB
ESC
PCI Device(s)
EISA Bus
290481-2

82434LX
PCI/CACHE/MEMORY CONTROLLER (PCMC)

- **Supports the 64-Bit Pentium™ Processor at 60 MHz and 66.667 MHz**
- **Supports Pipelined Addressing Capability of the Pentium Microprocessor**
- **High Performance CPU/PCI/Memory Interfaces via Posted-Write/Read-Prefetch Buffers**
- **Fully Synchronous 33.33 MHz PCI Bus Interface with Full Bus Master Capability**
- **Supports the Pentium Processor Cache (First Level Cache) in either Write-Through or Write-Back Mode**
- **Programmable Attribute Map of DOS and BIOS Regions for System Flexibility**
- **Integrated Low Skew Clock Driver for Distributing 66.667 MHz Clock**
- **Integrated Second Level Cache Controller**
  - **— Integrated Cache Tag RAM**
  - **— Write-Through and Write-Back Cache Modes**
  - **— Direct-Mapped Organization**
  - **— Supports Standard and Burst SRAMs**
  - **— 256 KByte and 512 KByte Sizes**
  - **— Cache Hit Cycle of 3-1-1-1 on Reads and Writes Using Burst SRAMs**
  - **— Cache Hit Cycle of 3-2-2-2 on Reads and 4-2-2-2 on Writes Using Standard SRAMs**
- **Integrated DRAM Controller**
  - **— Supports 2 MBytes to 192 MBytes of Cacheable Main Memory**
  - **— Supports DRAM Access Times of 70 ns and 60 ns**
  - **— CPU Writes Posted to DRAM at 4-1-1-1**
  - **— Refresh Cycles Decoupled from ISA Refresh to Reduce the DRAM Access Latency**
  - **— Refresh by RAS#-Only, or CAS#-before-RAS#, in Single or Burst of Four**
- **Host/PCI Bridge**
  - **— Translates CPU Cycles into PCI Bus Cycles**
  - **— Translates Back-to-Back Sequential CPU Writes into PCI Burst Cycles**
  - **— Burst Mode Writes to PCI in Zero PCI Wait States (i.e., Data Transfer Every Cycle)**
  - **— Full Concurrency between CPU-to-Main Memory and PCI-to-PCI Transactions**
  - **— Full Concurrency between CPU-to-Second Level Cache and PCI-to-Main Memory Transactions**
  - **— Same Core Cache and Memory System Logic Design for ISA or EISA Systems**
  - **— Cache Snoop Filter Ensures Data Consistency for PCI-to-Main Memory Transactions**
- **PCMC (208-Pin QFP) Uses 5V CMOS Technology**

The 82434LX PCI, Cache, Memory Controller (PCMC) integrates the cache and main memory DRAM control functions and provides the bus control for transfers between the CPU, cache, main memory, and the Peripheral Component Interconnect (PCI) Local Bus. The cache controller supports both write-through and write-back cache policies and cache sizes of 256 KBytes and 512 KBytes. The cache memory can be implemented with either standard or burst SRAMs. The PCMC cache controller integrates a high-performance Tag RAM to reduce system cost. Up to twelve single-sided SIMMs or six double-sided SIMMs provide a maximum of 192 MBytes of main memory. The PCMC is intended to be used with the 82433LX Local Bus Accelerator (LBX). The LBX provides the Host-to-PCI address path and data paths between the CPU/cache, main memory, and PCI. The LBX also contains posted write buffers and read-prefetch buffers. Together, these two components provide a full function data path to main memory and form a PCI bridge to the Host subsystem (CPU/cache).

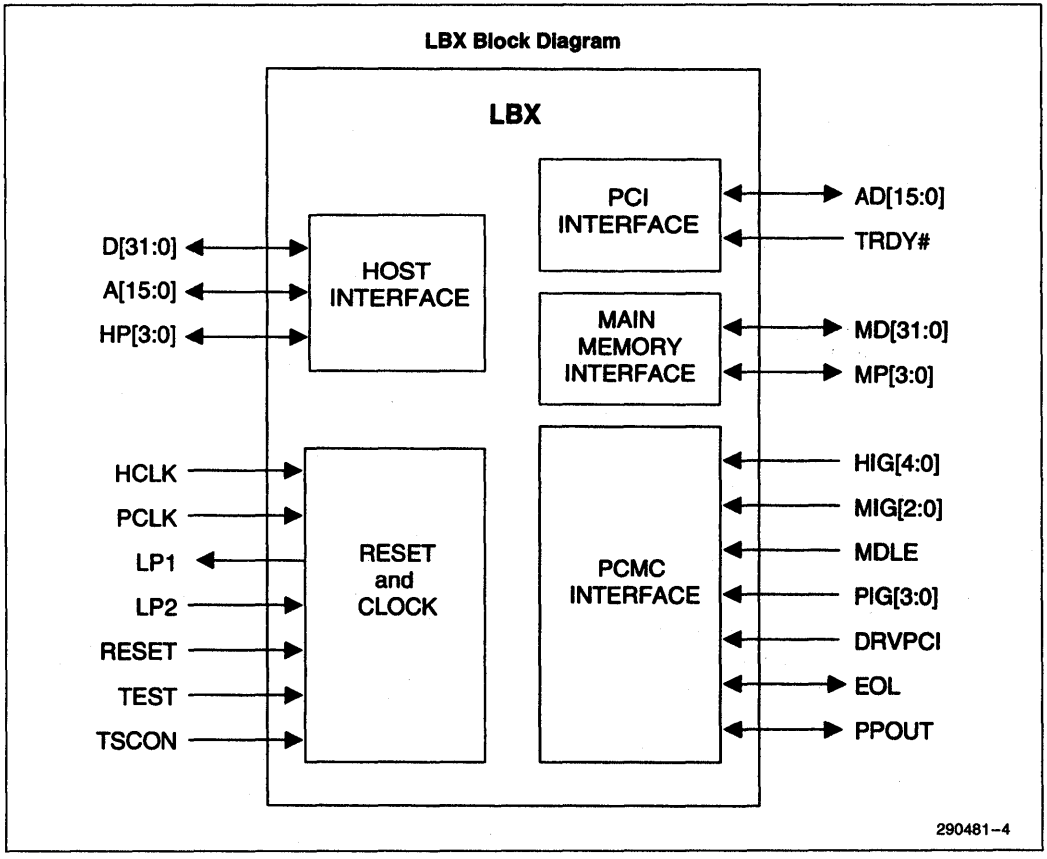
PCMC Block Diagram



82433LX
LOCAL BUS ACCELERATOR (LBX)

- **Supports the Full 64-Bit Pentium™ Processor Data Bus at 66.667 MHz**
- **Provides a 64-Bit Interface to DRAM Memory and a 32-Bit Interface to PCI**
- **Five Integrated Write Posting and Read Prefetch Buffers Increase CPU and PCI Master Performance**
  - **— CPU-to-Memory Posted Write Buffer 4 Quadwords Deep**
  - **— PCI-to-Memory Posted Write Buffer Two Buffers, 4 Dwords Each**
  - **— PCI-to-Memory Read Prefetch Buffer 4 Quadwords Deep**
  - **— CPU-to-PCI Posted Write Buffer 4 Dwords Deep**
  - **— CPU-to-PCI Read Prefetch Buffer 4 Dwords Deep**
- **Host-to-Memory and Host-to-PCI Write Posting Buffers Permit Near Zero Wait State Write Performance**
- **Dual-Port Architecture Allows Concurrent Operations on the Host and PCI Buses**
- **Operates Synchronous to the 66.667 MHz CPU and 33.33 MHz PCI Clocks**
- **Supports Burst Read and Writes of Memory from the Host and PCI Buses**
- **Host CPU Writes to PCI in Zero Wait State PCI Bursts with Optional TRDY# Connection**
- **Byte Parity Support for the Host and Memory Buses**
  - **— Optional Parity Generation for Host to Memory Transfers**
  - **— Optional Parity Checking for the Secondary Cache Residing on the Host Data Bus**
  - **— Parity Checking for Host and PCI Memory Reads**
  - **— Parity Generation for PCI to Memory Writes**
- **160-Pin QFP Package**
- **5V CMOS Technology**

Two 82433LX Local Bus Accelerator (LBX) components provide a 64-bit data path between the Host CPU/cache and main memory, a 32-bit data path between the Host CPU and Peripheral Component Interconnect (PCI) Local Bus, and a 32-bit data path between the PCI bus and main memory. The dual-port architecture allows concurrent operations on the Host and PCI Buses. The LBXs incorporate three write posting buffers and two read prefetch buffers to increase Pentium processor and PCI Master performance. The LBX supports byte parity for the Host and main memory buses. The LBX is intended to be used with the 82434LX PCI/Cache/Memory Controller (PCMC). During bus operations between the Host, main memory, and PCI, the PCMC commands the LBXs to perform functions such as latching address and data, merging data, and enabling output buffers. Together, these three components form a "host bridge" which provides a full function dual-port data path interface, linking the Host CPU and PCI bus to main memory.

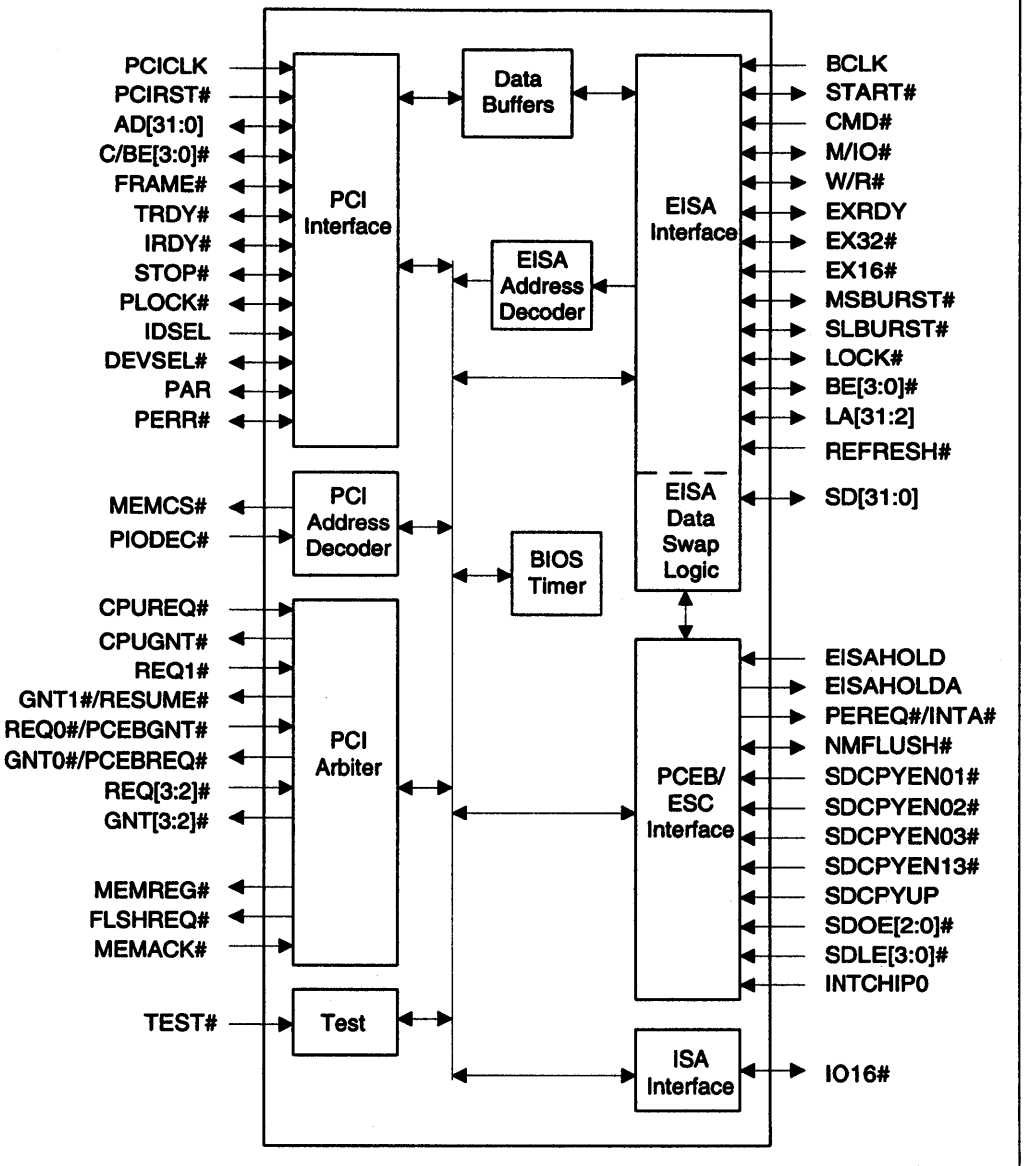

LBX Block Diagram
LBX
PCI INTERFACE
HOST INTERFACE
MAIN MEMORY INTERFACE
RESET and CLOCK
PCMC INTERFACE
D[31:0]
A[15:0]
HP[3:0]
HCLK
PCLK
LP1
LP2
RESET
TEST
TSCON
AD[15:0]
TRDY#
MD[31:0]
MP[3:0]
HIG[4:0]
MIG[2:0]
MDLE
PIG[3:0]
DRVPCI
EOL
PPOUT
290481-4

82375EB
PCI/EISA BRIDGE (PCEB)

- **Provides the Bridge between the PCI 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 at 25 MHz to 33.33 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**
  - **— Instruct All PCI Devices to Flush Buffers Pointing to PCI Bus before Granting EISA Access to PCI**
- **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**
- **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**
- **208-Pin QFP Package**
- **5V CMOS Technology**

The 82375EB PCI-EISA Bridge (PCEB) provides the master/slave functions on both the Peripheral Component Interconnect (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 logic 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.

PCEB Block Diagram

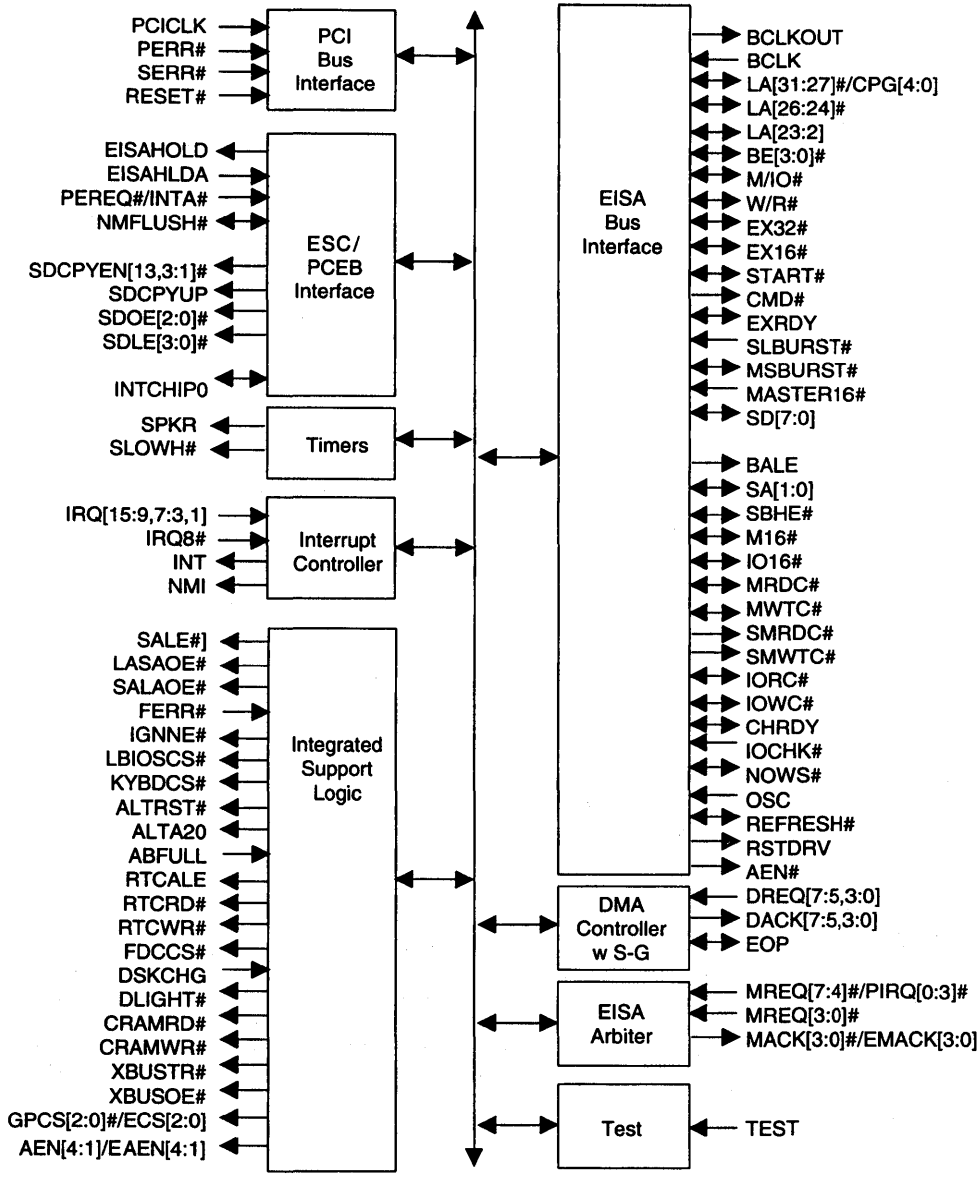


82374EB
EISA SYSTEM CONTROLLER (ESC)

- **Integrates EISA Compatible Bus Controller**
  - **— Translates Cycles between EISA and ISA Bus**
  - **— Supports EISA Burst and Standard Cycles**
  - **— Supports ISA No Wait State Cycles**
  - **— Supports Byte Assembly/ Disassembly for 8-Bit, 16-Bit and 32-Bit Transfers**
  - **— Supports Bus Frequency up to 8.33 MHz**
- **Supports Eight EISA Slots**
  - **— Directly Drives Address, Data and Control Signals for Eight Slots**
  - **— Decodes Address for Eight Slot Specific AENs**
- **Provides Enhanced DMA Controller**
  - **— Provides Scatter-Gather Function**
  - **— Supports Type A, Type B, Type C (Burst), and Compatible DMA Transfers**
  - **— Provides Seven Independently Programmable Channels**
  - **— Integrates Two 82C37A Compatible DMA Controllers**
- **Provides High Performance Arbitration**
  - **— Supports Eight EISA Masters and PCEB**
  - **— Supports ISA Masters, DMA Channels, and Refresh**
  - **— Provides Programmable Arbitration Scheme for Fixed, Rotating, or Combination Priority**
- **Integrates Support Logic for X-Bus Peripherals and More**
  - **— Generates Chip Selects/Encoded Chip Selects for Floppy and Keyboard Controller, IDE, Parallel/Serial Ports, and General Purpose Peripherals**
  - **— Provides Interface for Real Time Clock**
  - **— Generates Control Signals for X-Bus Data Transceiver**
  - **— Integrates Port 92, Mouse Interrupt, and Coprocessor Error Reporting**
- **Integrates the Functionality of Two 82C59 Interrupt Controllers and Two 82C54 Timers**
  - **— Provides 14 Programmable Channels for Edge or Level Interrupts**
  - **— Provides 4 PCI Interrupts Routable to Any of 11 Interrupt Channels**
  - **— Supports Timer Function for Refresh Request, System Timer, Speaker Tone, Fail Safe Timer, and Periodic CPU Speed Control**
- **Generates Non-Maskable Interrupts (NMI)**
  - **— PCI System Errors**
  - **— PCI Parity Errors**
  - **— EISA Bus Parity Errors**
  - **— Fail Safe Timer**
  - **— Bus Timeout**
  - **— Via Software Control**
- **Provides BIOS Interface**
  - **— Supports 512 KBytes of Flash or EPROM BIOS on the X Bus**
  - **— Allows BIOS on PCI**
  - **— Supports Integrated VGA BIOS**
- **208-Pin QFP Package**
- **5V CMOS Technology**

The 82374EB EISA System Component (ESC) provides all the EISA system compatible functions. The ESC, with the PCEB, provides all the functions to implement an EISA to PCI bridge and EISA I/O subsystem. The ESC integrates the common I/O functions found in today's EISA based PC systems. The ESC incorporates the logic for an EISA (master and slave) interface, EISA Bus Controller, enhanced seven channel DMA controller with Scatter-Gather support, EISA arbitration, 14 channel interrupt controller, five programmable timer/counters, and non-maskable interrupt (NMI) control logic. The ESC also integrates support logic to decode peripheral devices such as the Flash BIOS, Real Time Clock, Keyboard/Mouse Controller, Floppy Controller, two Serial Ports, one Parallel Port, and IDE Hard Disk Drive.

ESC Block Diagram

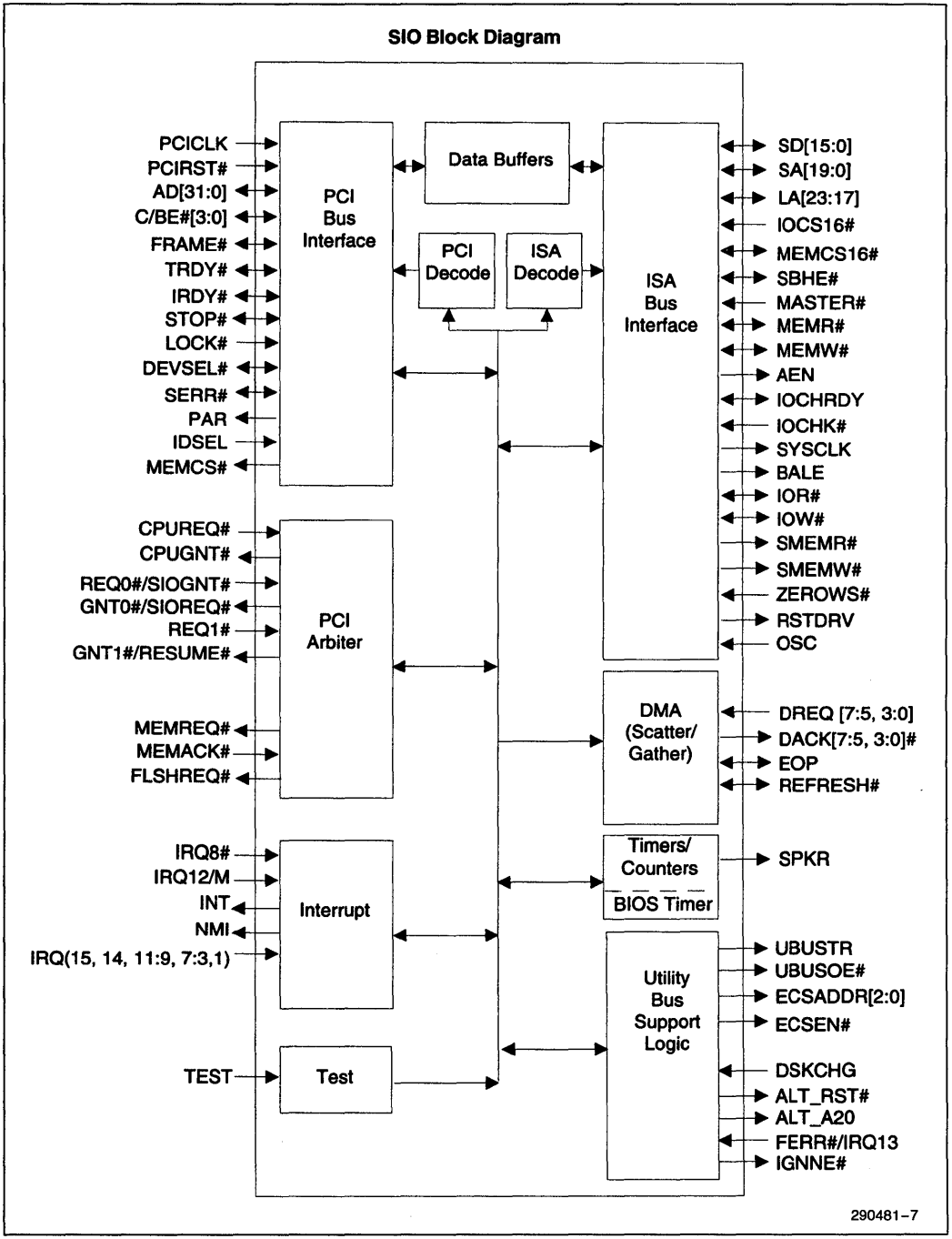


290481-6

82378IB
SYSTEM I/O (SIO)

- **Provides the Bridge between the PCI Bus and ISA Bus**
- **100% PCI and ISA Compatible**
  - **— PCI and ISA Master/Slave Interface**
  - **— Directly Drives 10 PCI Loads and 6 ISA Slots**
  - **— Supports PCI at 25 MHz and 33.33 MHz**
  - **— Supports ISA from 6 MHz to 8.33 MHz**
- **Enhanced DMA Functions**
  - **— Scatter/Gather**
  - **— Fast DMA Type A, B, and F**
  - **— Compatible DMA Transfers**
  - **— 32-Bit Addressability**
  - **— Seven Independently Programmable Channels**
  - **— Functionality of Two 82C37A DMA Controllers**
- **Integrated Data Buffers to Improve Performance**
  - **— 8-Byte DMA/ISA Master Line Buffer**
  - **— 32-Bit Posted Memory Write Buffer to ISA**
- **Integrated 16-Bit BIOS Timer**
- **Arbitration for PCI Devices**
  - **— Four PCI Masters Are Supported**
  - **— Fixed, Rotating, or a Combination of the Two**
- **Arbitration for ISA Devices**
  - **— ISA Masters**
  - **— DMA and Refresh**
- **Utility Bus (X-Bus) Peripheral Support**
  - **— Provides Chip Select Decode**
  - **— Controls Lower X-Bus Data Byte Transceiver**
  - **— Integrates Port 92, Mouse Interrupt, Coprocessor Error Reporting**
- **Integrates the Functionality of One 82C54 Timer**
  - **— System Timer**
  - **— Refresh Request**
  - **— Speaker Tone Output**
- **Integrates the Functionality of Two 82C59 Interrupt Controllers**
  - **— 14 Interrupts Supported**
- **Non-Maskable Interrupts (NMI)**
  - **— PCI System Errors**
  - **— ISA Parity Errors**
- **208-Pin QFP Package**
- **5V CMOS Technology**

The 82378IB System I/O (SIO) component provides the bridge between the PCI local bus and the ISA expansion bus. The SIO also integrates many of the common I/O functions found in today's ISA based PC systems. The SIO incorporates the logic for a PCI interface (master and slave), ISA interface (master and slave), enhanced seven channel DMA controller that supports fast DMA transfers and Scatter/Gather, data buffers to isolate the PCI bus from the ISA bus and to enhance performance, PCI and ISA arbitration, 14 level interrupt controller, a 16-bit BIOS timer, three programmable timer/counters, and non-maskable-interrupt (NMI) control logic. The SIO also provides decode for peripheral devices such as the Flash BIOS, Real Time Clock, Keyboard/Mouse Controller, Floppy Controller, two Serial Ports, one Parallel Port, and IDE Hard Disk Drive.


SIO Block Diagram
PCICLK
PCIRST#
AD[31:0]
C/BE#[3:0]
FRAME#
TRDY#
IRDY#
STOP#
LOCK#
DEVSEL#
SERR#
PAR
IDSEL
MEMCS#
PCI Bus Interface
Data Buffers
PCI Decode
ISA Decode
ISA Bus Interface
SD[15:0]
SA[19:0]
LA[23:17]
IOCS16#
MEMCS16#
SBHE#
MASTER#
MEMR#
MEMW#
AEN
IOCHRDY
IOCHK#
SYSCLK
BALE
IOR#
IOW#
SMEMR#
SMEMW#
ZEROWS#
RSTDRV
OSC
CPUREQ#
CPUGNT#
REQ0#/SIOGNT#
GNT0#/SIOREQ#
REQ1#
GNT1#/RESUME#
PCI Arbiter
MEMREQ#
MEMACK#
FLSHREQ#
DMA (Scatter/ Gather)
DREQ [7:5, 3:0]
DACK[7:5, 3:0]#
EOP
REFRESH#
IRQ8#
IRQ12/M
INT
NMI
IRQ(15, 14, 11:9, 7:3,1)
Interrupt
Timers/ Counters
BIOS Timer
SPKR
TEST
Test
Utility Bus Support Logic
UBUSTR
UBUSOE#
ECSADDR[2:0]
ECSEN#
DSKCHG
ALT_RST#
ALT_A20
FERR#/IRQ13
IGNNE#
290481-7

82374EB
EISA SYSTEM COMPONENT (ESC)

- **Integrates EISA Compatible Bus Controller**
  - **Translates Cycles between EISA and ISA Bus**
  - **Supports EISA Burst and Standard Cycles**
  - **Supports ISA No Wait State Cycles**
  - **Supports Byte Assembly/ Disassembly for 8-, 16- and 32-Bit Transfers**
  - **Supports Bus Frequency up to 8.33 MHz**
- **Supports Eight EISA slots**
  - **Directly Drives Address, Data and Control Signals for Eight Slots**
  - **Decodes Address for Eight Slot Specific AENs**
- **Provides Enhanced DMA Controller**
  - **Provides Scatter-Gather Function**
  - **Supports Type A, Type B, Type C (Burst), and Compatible DMA Transfers**
  - **Provides Seven Independently Programmable Channels**
  - **Integrates Two 82C37A Compatible DMA Controllers**
- **Provides High Performance Arbitration**
  - **Supports Eight EISA Masters and PCEB**
  - **Supports ISA Masters, DMA Channels, and Refresh**
  - **Provides Programmable Arbitration Scheme for Fixed, Rotating, or Combination Priority**
- **Integrates Support Logic for X-Bus Peripheral and More**
  - **Generates Chip Selects/Encoded Chip Selects for Floppy and Keyboard Controller, IDE, Parallel/ Serial Ports, and General Purpose Peripherals**
  - **Provides Interface for Real Time Clock**
  - **Generates Control Signals for X-Bus Data Transceiver**
  - **Integrates Port 92, Mouse Interrupt, and Coprocessor Error Reporting**
- **Integrates the Functionality of two 82C59 Interrupt Controllers and two 82C54 Timers**
  - **Provides 14 Programmable Channels for Edge or Level Interrupts**
  - **Provides 4 PCI Interrupts Routible to Any of 11 Interrupt Channels**
  - **Supports Timer Function for Refresh Request, System Timer, Speaker Tone, Fail Safe Timer, and Periodic CPU Speed Control**
- **Generates Non-Maskable Interrupts (NMI)**
  - **PCI System Errors**
  - **PCI Parity Errors**
  - **EISA Bus Parity Errors**
  - **Fail Safe Timer**
  - **Bus Timeout**
  - **Via Software Control**
- **Provides BIOS Interface**
  - **Supports 512 Kbytes of Flash or EPROM BIOS on the X-Bus**
  - **Allows BIOS on PCI**
  - **Supports Integrated VGA BIOS**
- **208-Pin QFP Package**

The 82374EB EISA System Component (ESC) provides all the EISA system compatible functions. The ESC with the PCEB provide all the functions to implement an EISA to PCI bridge and EISA I/O subsystem. The ESC integrates the common I/O functions found in today's EISA based PC systems. The ESC incorporates the logic for a EISA (master and slave) interface, EISA Bus Controller, enhanced seven channel DMA controller with Scatter-Gather support, EISA arbitration, 14 channel interrupt controller, five programmable timer/counters, and non-maskable-interrupt (NMI) control logic. The ESC also integrates support logic to decode peripheral devices such as the Flash BIOS, Real Time Clock, Keyboard/Mouse Controller, Floppy Controller, two Serial Ports, one Parallel Port, and IDE Hard Disk Drive.

Manufactured and tested for Intel Corporation by LSI Logic in accordance with LSI's internal standards.
Windows® is a registered trademark of Microsoft Corporation.
Win-NT™ is trademarked by Microsoft Corporation.

April 1993
Order Number: 290476-001

82374EB EISA System Component

CONTENTS

CONTENTS

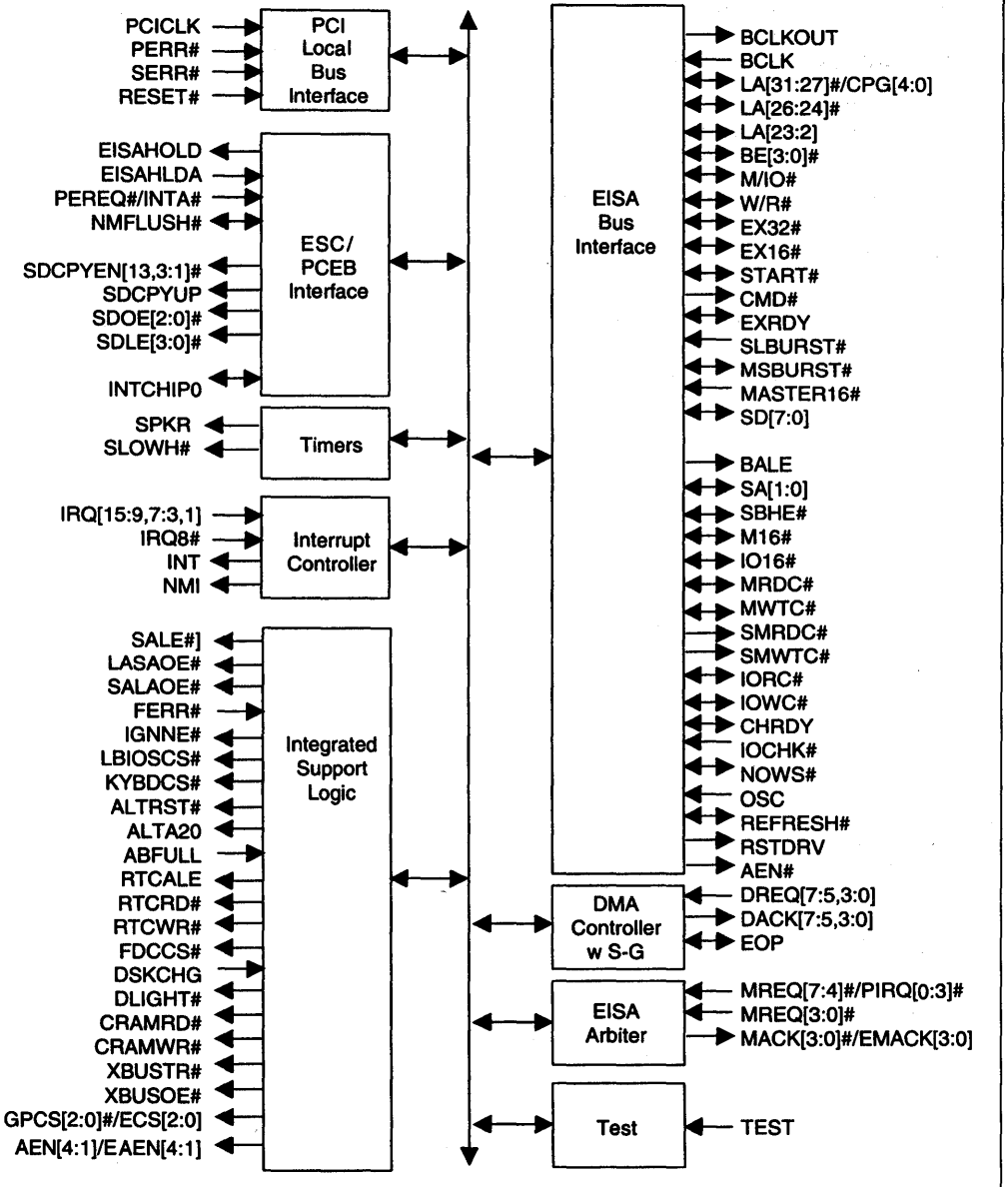
CONTENTS

CONTENTS

CONTENTS

CONTENTS

ESC Block Diagram



290476-1

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™ CPU). System designers can take advantage of the power of the PCI (Peripheral Component Interconnect) for the local I/O 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 1-0 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 Local Bus as a primary I/O bus

— EISA Bus as a secondary I/O bus

This bus hierarchy allows concurrency for simultaneous operations on all three bus environments. Data buffering permits concurrency for operations that crossover into another bus environment. For example, a PCI device could post data into the PCEB, permitting the PCI Local Bus transaction to complete in a minimum time and freeing up the PCI Local Bus for further transactions. The PCI device does not have to wait for the transfer to complete to its final destination. Meanwhile, any ongoing EISA Bus transactions are permitted to complete. The posted data is then transferred to its EISA Bus destination when the EISA Bus is available. The PCI-EISA Bridge chip set implements extensive buffering for PCI-to-EISA and EISA-to-PCI bus transactions. In addition to concurrency for the operations that cross bus environments, data buffering allows the fastest operations within a particular bus environment (via PCI burst transfers and EISA burst transfers).

The PCI local bus with 132 MByte/sec and EISA with 33 MByte/sec peak data transfer rate represent bus environments with significantly different bandwidths. Without buffering, transfers that cross the single bus environment are performed at the speed of the slower bus. Data buffers provide a mechanism for data rate adoption so that the operation of the fast bus environment (PCI), i.e., usable bandwidth, is not significantly impacted by the slower bus environment (EISA).

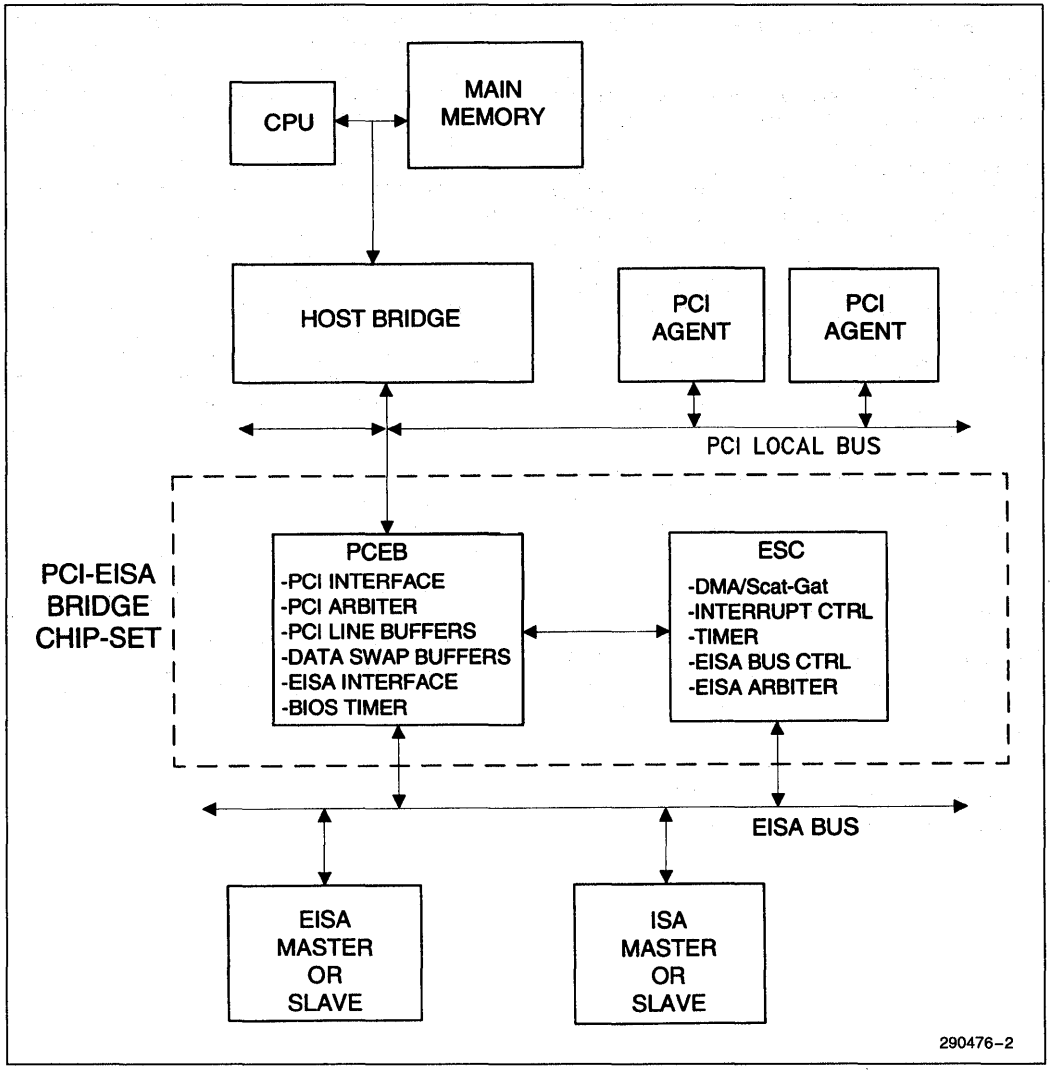


Figure 1-1. PCI-EISA Chip Set System Block Diagram

PCI Local Bus

The PCI Local 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 is 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 at frequencies from 20 MHz–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 slave registers 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 (address/data multiplexed)
- 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

System partitioning shown in Figure 1-0 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 L2 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 i486 or the new generation processors such as Pentium). 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 which 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 1-0 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 with 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 the PCI Local Bus, the functions include PCI Local Bus arbitration and default bus driver. For the EISA Bus, central functions include the EISA Bus controller and EISA arbiter are integrated in the ESC component and EISA Data Swap Logic is integrated in the PCEB.

Integrated System Functions

The PCI-EISA Bridge chip set integrates system functions including PCI parity and system errors reporting, buffer coherency management protocol, 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 provides the interface (bridge) between PCI and EISA buses by translating bus protocols in both directions. It uses extensive buffering on both the PCI and EISA interfaces to allow concurrent bus operations. The PCEB also implements the PCI central support functions (e.g., PCI arbitration, error signal support, and subtractive decoding). The major functions provided by the PCEB are described in this section.

PCI Local Bus Interface

The PCEB can be either a master or slave on the PCI Local Bus and supports bus frequencies from 25 MHz to 33 MHz. For PCI-initiated transfers, the PCEB can only be a slave. The PCEB becomes a slave when it positively decodes the cycle. The PCEB also becomes a slave for unclaimed cycles on the PCI Local Bus. These unclaimed cycles are either negatively or 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 and

burst cycles for memory operations. Note that, burst transfers cannot be performed to the PCEB's internal registers. Burst memory write cycles to the EISA Bus can transfer up to four Dwords, depending on available space in the PCEB's Posted Write Buffers. When space is no longer available in the buffers, the PCEB terminates the transaction. This supports the Incremental Latency Mechanism as defined in the Peripheral Component Interconnect (PCI) Specification. Note that, if the Posted Write Buffers are disabled, PCI burst operations are not performed and all transfers are single cycle.

For EISA-inititated transfers to the PCI Local 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 Local Bus, after which it surrenders the bus.

As a master on the PCI Local Bus, the PCEB generates address and command signal (C/BE#) 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 Local 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 Local Bus when no other device is requesting service. The arbiter also contains an efficient PCI retry mechanism to minimize PCI Local Bus thrashing when the PCEB generates a retry. The arbiter can be disabled, if an external arbiter is used.

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 Local Bus. For transfers contained to the EISA Bus, the PCEB is never a master or slave. However, the data swap logic contained in the PCEB is involved in these transfers, if data size translation is needed. The PCEB also provide 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 Local 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 Local 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. For memory write operations through the Posted Write Buffers, the PCEB uses EISA burst transfers, if supported by the slave, to flush the buffers. Otherwise, single cycle transfers are used. Single cycle transfers are used for all I/O cycles and memory reads.

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 MEMCS# is not used, this 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 Local Bus.

Data Buffering

To isolate the slower EISA Bus from the PCI Local Bus, the PCEB provides two types of data buffers. Buffer management control guarantees data coherency.

Four Dword wide Posted Write Buffers permit posting of PCI-initiated memory write cycles to the EISA Bus. For EISA-initiated cycles to the PCI Bus, there are four 16-byte wide Line Buffers. These buffers permit prefetching of PCI memory read data and posting of PCI memory write data.

By using burst transactions to fill or flush these buffers, if 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 ± 1 μs.

1.2 ESC Overview

The ESC implements system functions (e.g., timer/counter, DMA, and interrupt controller) and EISA subsystem control functions (e.g., EISA bus controller and EISA bus arbiter). 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 logic that is 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-bit 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 is also responsible for generating 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.

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 timeout functions and the CPU speed control.

Integrated Support Logic

To minimize the chip count for board designs, the ESC incorporates a number 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.

2.0 SIGNAL DESCRIPTION

This section provides a detail description of each signal. The signals (Figure 2-1) are arranged in functional group according to their associated interface.

The "#" symbol at the end of a signal indicates that the active, or asserted state occurs when the signal is at a low voltage level. When "#" is not presented 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. A pull-up sustains the inactive state until another agent drives it and is provided by the central resource.

2.1 PCI Local Bus Interface Signals

Pin Name	Type	Description
PCICLK	in	**PCI CLOCK:** PCICLK provides timing for all transactions on the PCI Local Bus. The ESC uses the PCI Clock (PCICLK) to generate EISA Bus Clock (BCLK). The PCICLK is divided by 3 or 4 to generate the BCLK. The EISA Bridge supports PCI Clock frequencies of 25 MHz through 33 MHz.
PERR#	in	**PARITY ERROR:** PERR# indicates a data parity error. PERR# may be pulsed active by any agent that detects an error condition. Upon sampling PERR# active, the ESC generates an NMI interrupt to the CPU.
SERR#	in	**SYSTEM ERROR:** SERR# may be pulsed active by any agent that detects an error condition. Upon sampling SERR# active, the ESC generates an NMI interrupt to the CPU.
RESET#	in	**SYSTEM RESET:** RESET# forces the entire ESC Chip into a known state. All internal ESC state machines are reset and all registers are set to their default values. RESET# may be asynchronous to PCICLK when asserted or negated. Although asynchronous, negation must be a clean, bounce-free edge. The ESC uses RESET to generate RSTDRV signal.

2.2 EISA Bus Interface Signals

Pin Name	Type	Description
BCLKOUT	out	**EISA BUS CLOCK OUTPUT:** BCLKOUT is typically buffered to create EISA Bus Clock (BCLK). The BCLK is the system clock used to synchronize events on the EISA/ISA bus. The BCLKOUT is generated by dividing the PCICLK. The ESC uses a divide by 3 or divide by 4 to generate the BCLKOUT.
BCLK	in	**EISA BUS CLOCK:** BCLK is an input to the ESC device. This ESC uses the BCLK to synchronize events on the EISA bus. The ESC generates or samples all the EISA/ISA bus signals on either the rising or the falling edge of BCLK.
LA[31:27]#/ CPG[4:0]	t/s	**EISA ADDRESS BUS/CONFIGURATION RAM PAGE ADDRESS:** These are multiplexed signals. These signals behave as the EISA address bus under all conditions except during access cycle to the Configuration RAM. **EISA ADDRESS BUS:** LA[31:27]# are directly connected to the EISA address bus. The ESC uses the address bus in conjunction with the BE[3:0]# signals as inputs to decode accesses to its internal resources except in DMA and Refresh modes. During DMA and Refresh modes, these are outputs, and the ESC uses these signals in conjunction with BE[3:0]# to drive Memory address. **CONFIGURATION RAM PAGE ADDRESS:** CPG[4:0] are connected to Configuration SRAM address lines. During I/O access to 0800h–08FFh, the ESC drives these signals with the configuration page address (the value contained in register 0C00h). The Configuration RAM Page Address function can be disabled by setting Mode Select register bit 5 = 0.
LA[26:24]# and LA[23:2]	t/s	**EISA ADDRESS BUS:** These signals are directly connected to the EISA address bus. The ESC uses the address bus in conjunction with the BE[3:0]# signals as inputs to decode accesses to its internal resources except in DMA and Refresh modes. During DMA and Refresh modes, these are outputs, and the ESC uses these signals in conjunction with BE[3:0]# to drive Memory address.
BE[3:0]#	t/s	**BYTE ENABLES:** BE[3:0]# signals are directly connected to the EISA address bus. These signals indicate which byte on the 32-bit EISA data bus are involved in the current cycle. BE[3:0]# are inputs during EISA master cycles which do not require assembly/disassembly operation. For EISA master assembly/ disassembly cycles, ISA master cycles, DMA, and Refresh cycles BE[3:0]# are outputs. BE0#: Corresponds to byte lane 0-SD[7:0] BE1#: Corresponds to byte lane 0-SD[15:8] BE2#: Corresponds to byte lane 0-SD[23:16] BE3#: Corresponds to byte lane 0-SD[31:24]
M/IO#	t/s	**MEMORY OR I/O CYCLE:** M/IO# signal is used to differentiate between memory cycles and I/O cycles on the EISA bus. A High value on this signal indicates a memory cycle, and a Low value indicates an I/O cycle. M/IO# is an input to the ESC during EISA master cycles, and M/IO# is an output during ISA, DMA, and ESC initiated Refresh cycles. M/IO# is floated during ISA master initiated Refresh cycles.
W/R#	t/s	**WRITE OR READ CYCLE:** W/R# signal is used to differentiate between write and read cycles on the EISA bus. A High value on this signal indicates a Write cycle, and a Low value indicates a Read cycle. W/R# is an input to the ESC during EISA master cycles, and W/R# is an output during ISA, DMA, and Refresh cycles.

2.2 EISA Bus Interface Signals (Continued)

Pin Name	Type	Description
EX32#	o/d	**EISA 32-BIT DEVICE DECODE:** EX32# signal is asserted by a 32-bit EISA slave device. EX32# assertion indicates that an EISA device has been selected as a slave, and the device has a 32-bit data bus size. The ESC uses this signal as an input as part of its slave decode to determine if data size translation and/or cycle translation is required. EX32# is an output of the ESC during the last portion of the mis-matched cycle. This is an indication to the backed-off EISA master that the data translation has been completed. The backed-off EISA master uses this signal to start driving the EISA bus again.
EX16#	o/d	**EISA 16-BIT DEVICE DECODE:** EX16# signal is asserted by a 16-bit EISA slave device. EX16# assertion indicates that an EISA device has been selected as a slave, and the device has a 16-bit data bus size. The ESC uses this signal as an input as part of its slave decode to determine if data size translation and/or cycle translation is required. EX16# is an output of the ESC during the last portion of the mis-matched cycle. This is an indication to the backed-off EISA master that the data translation has been completed. The backed-off EISA master uses this signal to start driving the EISA bus again.
START#	t/s	**START CYCLE:** START# signal provides timing control at the start of an EISA cycle. START# is asserted for one BCLK. START# is an input to the ESC during EISA master cycles except portions of the EISA master to mis-matched slave cycles where it becomes an output. During ISA, DMA, and Refresh cycles START# is an output.
CMD#	out	**COMMAND:** CMD# signal provides timing control within an EISA cycle. The ESC is a central resource of the CMD# signal, and the ESC generates CMD# during all EISA cycles. CMD# is asserted from the rising edge of BCLK simultaneously with the negation of START#, and remains asserted until the end of the cycle.
EXRDY	o/d	**EISA READY:** EXRDY signal is deasserted by EISA slave devices to add wait states to a cycle. EXRDY is an input to the ESC for EISA master cycles, ISA master cycles, and DMA cycles where an EISA slave has responded with EX32# or EX16# asserted. The ESC samples EXRDY on the falling edge of BCLK after CMD# is asserted (except during DMA compatible cycles). During DMA compatible cycles, EXRDY is sampled on the second falling edge of BCLK after CMD# is driven active. For all types of cycles if EXRDY is sampled inactive, the ESC keeps sampling it on every falling edge of BCLK#. EXRDY is an output for EISA master cycles decoded as accesses to the ESC internal registers. ESC forces EXRDY low for one BCLK at the start of a potential DMA burst write cycle to insure that the initial write data is held long enough to be sampled by the memory slave.
SLBURST#	in	**SLAVE BURST:** SLBURST# signal is asserted by an EISA slave to indicate that the device is capable of accepting EISA burst cycles. The ESC samples SLBURST# on the rising edge of BCLK at the end of START# for all EISA cycles. During DMA cycles, the ESC samples SLBURST# twice; once on the rising edge of BCLK at the beginning of START# and again on the rising edge of BCLK at the end of START#.
MSBURST#	t/s	**MASTER BURST:** MSBURST# signal is asserted by an EISA master to indicate EISA burst cycles. MSBURST# is asserted by an EISA master in response to an asserted SLBURST# signal. The ESC samples SLBURST# on the rising edge of BCLK that CMD# is asserted. If asserted, the ESC samples SLBURST# on all subsequent rising edges of BCLK until sampled negated. The ESC keeps CMD# asserted during Burst cycles. MSBURST# is an output during DMA burst cycles. The ESC drives MSBURST# active on the falling edge of BCLK, one half BCLK after SLBURST# is sampled active at the end of START#.

2.2 EISA Bus Interface Signals (Continued)

Pin Name	Type	Description
MASTER16#	in	**MASTER 16-BIT:** MASTER16# is asserted by a 16-bit EISA Bus master or an ISA Bus master device to indicate that it has control of the EISA Bus or ISA Bus. The ESC samples MASTER16# on the rising edge of BCLK that START# is asserted. If MASTER16# is sampled asserted, the ESC determines that a 16-bit EISA Bus master or an ISA Bus master owns the Bus. If MASTER16# is sampled negated at the first sampling point, the ESC will sample MASTER16# a second time on the rising edge of BCLK at the end of START#. If MASTER16# is sampled asserted here, the ESC determines that a 32-bit EISA Bus master has downshifted to a 16-bit Bus master, and thus, the ESC will disable the data size translation function.
SD[7:0]	t/s	**SYSTEM DATA:** SD[7:0] signals are directly connected to the System Data bus. The SD[7:0] pins are outputs during I/O reads when the ESC internal registers are being accessed and during interrupt acknowledge cycles. The SD[7:0] pins are input during I/O write cycles when the ESC internal registers are being accessed.

2.3 ISA Bus Signals

Pin Name	Type	Description
BALE	out	**BUS ADDRESS LATCH ENABLE:** BALE signal is asserted by the ESC to indicate that an address (SA[19:0], LA[23:17]), AEN and SBHE# signal lines are valid. The LA[23:17] address lines are latched on the trailing edge of BALE. BALE remains active throughout DMA and ISA Master cycles and Refresh cycles.
SA[1:0]	t/s	**ISA ADDRESS BITS 0 AND 1:** SA[1:0] are the least significant bits of the ISA address bus. SA[1:0] are inputs to the ESC during ISA master cycles except during ISA master initiated Refresh cycles. The ESC uses the SA[1:0] in conjunction with SBHE# to generate BE[3:0]# on the EISA bus. The SA[1:0] are outputs of the ESC during EISA master cycles and DMA cycles. The ESC generates these from BE[3:0]#.
SBHE#	t/s	**ISA BYTE HIGH ENABLE:** SBHE# signal indicates that the high byte on the ISA data bus (SD[15:8]) is valid. SBHE# is an input to the ESC during ISA master cycles, except during ISA master initiated Refresh cycles. The ESC uses the SBHE# in conjunction with SA[1:0] to generate BE[3:0]# on the EISA bus. SBHE# is an output during EISA master and DMA cycles.
M16#	o/d	**MEMORY CHIP SELECT 16:** M16# is an input when the ESC component owns the ISA bus. M16# is an output when an external ISA bus Master owns the ISA bus. The ISA slave memory drives this signal Low if it is a 16-bit memory device. For ISA to EISA translation cycles, the ESC combinatorially asserts M16# if either EX32# or EX16# are asserted. This signal has an external pullup resistor.
IO16#	o/d	**16-BIT I/O CHIP SELECT:** IO16# signal is used to indicate a 16-bit I/O bus cycle. This signal is asserted by the I/O devices to indicate that they support 16-bit I/O bus cycles. All I/O accesses to the ESC registers are run as 8-bit I/O bus cycles. This signal has an external pullup resistor.
MRDC#	t/s	**MEMORY READ:** MRDC# signal indicates a read cycle to the ISA memory devices. MRDC# is the command to a memory slave that it may drive data onto the ISA data bus. MRDC# is an output when the ESC owns the ISA bus. MRDC# is an input when an external ISA Bus master owns the ISA Bus. This signal is driven by the ESC during refresh cycles.

2.3 ISA Bus Signals (Continued)

Pin Name	Type	Description
MWTC#	t/s	**MEMORY WRITE:** MWTC# signal indicates a write cycle to the ISA memory devices. MWTC# is the command to a memory slave that it may latch data from the ISA data bus. MWTC# is an output when the ESC owns the ISA bus. MWTC# is an input when an ISA Bus master owns the ISA Bus.
SMRDC#	out	**SYSTEM MEMORY READ:** SMRDC# signal is asserted by the ESC to request a memory slave to drive data onto the data lines. SMRDC# indicates that the memory read cycle is for an address below the 1 Mbyte range on the ISA bus. This signal is also asserted during refresh cycles.
SMWTC#	out	**SYSTEM MEMORY WRITE:** SMWTC# signal is asserted by the ESC to request a memory slave to accept data from the data lines. SMWTC# indicates that the memory write cycle is for an address below the 1 Mbyte range.
IORC#	t/s	**I/O READ:** IORC# is the command to an ISA I/O slave device that it may drive data on to the data bus (SD[15:0]). The device must hold the data valid until after IORC# is negated. IORC# is an output when the ESC component owns the ISA bus. IORC# is an input when an ISA Bus master owns the ISA Bus.
IOWC#	t/s	**I/O WRITE:** IOWC# is the command to an ISA I/O slave device that it may latch data from the ISA data bus (SD[15:0]). IOWC# is an output when the ESC component owns the ISA Bus. IOWC# is an input when an ISA Bus master owns the ISA Bus.
CHRDY	o/d	**I/O CHANNEL READY:** CHRDY when asserted allows ISA Bus resources request additional time (wait states) to complete the cycle. CHRDY is an input when the ESC owns the ISA Bus. CHRDY is an input to the ESC during compatible DMA cycles. CHRDY is an output during ISA Bus master cycles to PCI slave or ESC internal register. The ESC will ignore CHRDY for ISA-Bus master accessing an ISA-Bus slave.
IOCHK#	in	**IO CHANNEL CHECK:** IOCHK# can be asserted by any resource on the ISA Bus. When asserted, it indicates that a parity or an uncorrectable error has occurred for a device or memory on the ISA Bus. A NMI will be generated to the CPU if enabled.
NOWS#	o/d	**ZERO WAIT STATES:** NOWS# indicates that a peripheral device wishes to execute a zero wait states bus cycle (the normal default 16-bit ISA bus memory or I/O cycle is 3 BCLKS). When NOWS# is asserted, a 16-bit memory cycle will occur in two BCLKs and a 16-bit I/O cycle will occur in three BCLKs. When NOWS# is asserted by an 8-bit device the default 6 BCLKs cycle is shortened to 4 or 5 BCLKs. NOWS# is an input when the ESC is performing bus translation cycles. NOWS# is an output when the ESC internal registers are accessed. If CHRDY and NOWS# are both asserted during the same clock then NOWS# will be ignored and wait states will be added as a function of CHRDY (CHRDY has precedence over NOWS#).
OSC	in	**OSCILLIATOR:** OSC is the 14.31818 MHz signal with 50% duty cycle. OSC is used by the ESC Timers.
RSTDRV	out	**RESET DRIVE:** RSTDRV is asserted by the ESC. An asserted RSTDRV cause a hardware reset of the devices on the ISA Bus. RSTDRV is asserted whenever the RESET# input to the ESC is asserted.

2.3 ISA Bus Signals (Continued)

Pin Name	Type	Description
REFRESH#	t/s	**REFRESH:** REFRESH# is used by the ESC as an output to indicate when a refresh cycle is in progress. It should be used to enable the SA[15:0] address to the row address inputs of all banks of dynamic memory on the ISA bus so that when MRDC# goes active, the entire expansion bus dynamic memory is refreshed. Memory slaves must not drive any data onto the bus during refresh and should not add wait states since this will affect the entire system throughput. As an output, this signal is driven directly onto the ISA bus. This signal is an output only when the ESC DMA Refresh is a master on the bus responding to an internally generated request for Refresh. Upon RESET this pin will tri-state. Note that address lines [15:8] are driven during refresh, but the value is meaningless and is not used to refresh ISA bus memory. REFRESH# may be asserted by an expansion bus adapter acting as a 16-bit ISA bus master.
AEN#	out	**ADDRESS ENABLE:** AEN# is driven high for Bus master cycles. AEN# is driven low for DMA cycles and Refresh cycles. AEN# is used to disable I/O devices from responding to DMA and Refresh cycles. System designs which do not use the slots specific AENs (AEN[4:1]/EAEN[4:1]) provided by the ESC can use the AEN# signal to generate their own slot specific AENs.
AEN[4:1]/ EAEN[4:1]	out	These pins have slightly different function depending on the ESC configuration (Mode Select register bit 1 and bit 0). **SLOT SPECIFIC ADDRESS ENABLE:** If the ESC is programmed to support 4 EISA slot, these signals function as Slot Specific Address Enables (AEN[4:1]). **ENCODED SLOT SPECIFIC ADDRESS ENABLE:** If the ESC has been programmed to support more than 4 EISA slots, then these signals behave as Encoded Address Enables (EAEN[4:1]). A discrete decoder is required to generate slot specific AENs. Refer to Section 5.8.1 AEN GENERATION for a detailed description of these signals.

2.4 DMA Signal Description

Pin Name	Type	Description
DREQ[7:5,3:0]	in	**DMA REQUEST:** DREQ signals are either used to request DMA service from the ESC or used to gain control of the ISA Bus by a ISA Bus master. The active level (high or low) is programmed in the Command registers. When the Command register Bit 6 is programmed to 0, DREQ are asserted high, otherwise the DREQ are asserted low. All inactive to active edges of DREQ are assumed to be asynchronous. The request must remain asserted until the appropriate DACK is negated. At power-up and after RESET, these lines should be low (negated).
DACK# [7:5,3:0]	out	**DMA ACKNOWLEDGE:** DACK# indicates that a request for DMA service from the DMA subsystem has been recognized or that a ISA Bus master has been granted the bus. The level of the DACK lines when asserted may be programmed to be either high or low. This is accomplished by programming the DMA Command register. These lines should be used to decode the DMA slave device with the IORC# or IOWC# line to indicate selection. If used to signal acceptance of a bus master request, this signal indicates when it is legal to assert MASTER16#. If the DMA controller has been programmed for a timing mode other than compatible mode, and another device has requested the bus, and a 4 μs time has elapsed, DACK# will be negated and the transfer stopped before the transfer is complete. In this case, the transfer will be restarted at the next arbitration period in which the channel wins the bus. Upon reset these lines are negated.
EOP	t/s	**END OF PROCESS:** EOP pin acts in one of two modes, and it is directly connected to the TC line of the ISA Bus. In the first mode, EOP-In, the pin is an input and can be used by a DMA slave to stop a DMA transfer. In the second mode, TC-Out, it is used as a terminal count output by DMA slaves. An active pulse is generated when the byte counter reaches its last value. **EOP-IN MODE:** During DMA, for all transfer types, the EOP pin is sampled by the ESC. If it is sampled asserted, the address bus is tri-stated and the transfer is terminated. **TC-OUT MODE:** The EOP output will be asserted after a new address has been output if the byte count expires with that transfer. The EOP (TC) will stay asserted until AEN# is negated unless AEN is negated during an Autoinitialization. EOP (TC) will be negated before AEN is negated during an Autoinitialization. **INTOUT MODE:** In this mode the EOP signal has the same behavior as the Chaining Interrupt or the Scatter-Gather interrupt to the host processor (IRQ13). If a scatter-gather or chaining buffer is expired, EOP will go active on the falling edge of BCLK. Only the currently active channel's interrupt will be reflected on this pin. Other channel's with active interrupts pending will not affect the EOP pin. Whenever all the DMA channels are not in use, the EOP pin is kept in output mode and negated. After reset, the EOP pin is kept in output mode and negated.

2.5 EISA Arbitration Signals

<table>
<tr><th>Pin Name</th><th>Type</th><th>Description</th></tr>
<tr><td>MREQ[3:0]#</td><td>in</td><td>MASTER REQUEST: MREQ[3:0]# are slot specific signals used by EISA bus masters to request bus access. MREQ# once asserted, must remain asserted until the corresponding MACK# is asserted. The MREQ# is negated on the falling edge of BCLK slightly before the end of a master transfer. The LA[ ], BE[ ]#, M-IO#, and W/R# lines should be floated on or before the rising edge of BCLK after MREQ# is negated. The end of the last bus cycle is derived from CMD# in this case. The MREQ# signals are asserted on the falling edge of BCLK. MREQ# is always sampled on the rising edge of BCLK. MREQ# is synchronous with respect to BCLK. After asserting MREQ#, the corresponding master must not assert MREQ# until 1.5 BCLKs after CMD# is negated.</td></tr>
<tr><td>MREQ[7:4]# / PIRQ[0:3]#</td><td>in</td><td>These pins behave in one of two modes depending on the state of the Mode Select register bit 1 and bit 0.
MASTER REQUEST: MREQ# lines are slot specific signals used by EISA bus masters to request bus access. This signal behaves in the same manner as MREQ[3:0]# signals.
PCI INTERRUPT REQUEST: PIRQ# are used generate asynchronous interrupts to the CPU via the Programmable Interrupt Controller (82C59) integrated in the ESC. These signals are defined as level sensitive and are asserted low. The PIRQx# can be shared with PC compatible interrupts IRQ3:IRQ7, IRQ9:IRQ15. The PIRQx# Route Control Register determines which PCI interrupt is shared with which PC compatible interrupt.
<table>
<tr><th rowspan="2">Register Bit[1:0]</th><th colspan="4">Pins</th></tr>
<tr><th>MREQ7#/ PIRQ0#</th><th>MREQ6#/ PIRQ1#</th><th>MREQ5#/ PIRQ2#</th><th>MREQ4#/ PIRQ3#</th></tr>
<tr><td>00</td><td>PIRQ0#</td><td>PIRQ1#</td><td>PIRQ2#</td><td>PIRQ3#</td></tr>
<tr><td>01</td><td>PIRQ0#</td><td>PIRQ1#</td><td>MREQ5#</td><td>MREQ4#</td></tr>
<tr><td>10</td><td>PIRQ0#</td><td>MREQ6#</td><td>MREQ5#</td><td>MREQ4#</td></tr>
<tr><td>11</td><td>MREQ7#</td><td>MREQ6#</td><td>MREQ5#</td><td>MREQ4#</td></tr>
</table></td></tr>
<tr><td>MACK[3:0]# / EMACK[3:0]</td><td>out</td><td>These pins behave in one of two modes depending on the state of the Mode Select register bit 1 and bit 0. If the ESC is programmed to support 4 EISA slots then these pins are used as MACK#. If the ESC is programmed to support more than 4 EISA slots then these pins are used as EMACK#
MASTER ACKNOWLEDGE: The MACK[3:0]# signals are asserted from the rising edge of BCLK at which time the bus master may begin driving the LA[ ], BE[ ]#, M/IO#, and W/R# lines on the next falling edge of BCLK. MACK# will stay asserted until the rising edge of BCLK when MREQ# is sampled negated. MACK# is sampled by EISA Bus masters on the falling edge of BCLK. If another device has requested the bus, MACK# will be negated before MREQ# is negated. When MACK# is negated, the granted device has a maximum of 8 μs to negate MREQ# and begin a final bus cycle. The ESC may negate the MACK# signal a minimum of one BCLK after asserting it if another device (or refresh) is requesting the bus. Upon reset MACK# is negated.
ENCODED MASTER ACKNOWLEDGE: EMACK# behaves like MACK#. The difference is that a discrete decoder is required to generate MACK# for the EISA Bus masters.
Refer to Section 5.8.2 MACK Generation for details.</td></tr>
</table>

2.6 Timer Unit Signal

Pin Name	Type	Description
SPKR	out	**SPEAKER DRIVE:** SPKR is the output of Timer 1, Counter 2 and is "ANDed" with Port 061h Bit 1 to provide Speaker Data Enable. This signal drives an external speaker driver device, which in turn drives the ISA system speaker. SPKR has a 24 mA drive capability. Upon reset, its output state is 0.
SLOWH#	out	**SLOW DOWN CPU:** SLOWH# is the output of Timer 2, Counter 2. This counter is used to slow down the main CPU of its execution via the CPU's HOLD pin by pulse width modulation. The first read of I/O register in the 048h-04Bh range will enable SLOWH# signal to follow the output of the Timer 2, Counter 2. Upon reset, SLOWH# is negated.

2.7 Interrupt Controller Signals

Pin Name	Type	Description
IRQ[15:9], IRQ8#, IRQ[7:3,1]	in	**INTERRUPT REQUEST:** IRQ These signals provide both system board components and EISA bus I/O devices with a mechanism for asynchronously interrupting the CPU. The assertion mode of each interrupt can be programmed to be edge or level triggered. An asserted IRQ input must remain asserted until after the falling edge of INTA#. If the input is negated before this time, a DEFAULT IRQ7 will occur when the CPU acknowledges the interrupt. (Refer to Section 9.12.7.)
INT	out	**CPU INTERRUPT:** INT is driven by the ESC to signal the CPU that an Interrupt request is pending and needs to be serviced. It is asynchronous with respect to BCLK or PCICLK and it is always an output. The interrupt controllers must be programmed following a reset to ensure that this pin takes on a known state. Upon reset the state of this pin is undefined.
NMI	out	**NON-MASKABLE INTERRUPT:** NMI is used to force a non-maskable interrupt to the CPU. The CPU registers an NMI when it detects a rising edge on NMI. NMI will remain active until a read from the CPU to the NMI register at port 061h is detected by the ESC. This signal is set to low upon reset.

2.8 ESC/PCEB Interface Signals

2.8.1 ARBITRATION AND INTERRUPT ACKNOWLEDGE CONTROL

Pin Name	Type	Description
EISAHOLD	out	**EISA HOLD:** EISAHOLD is used to request control of the EISA bus from its default owner, PCEB. This signal is synchronous to PCICLK and is asserted when RESET# is asserted.
EISAHLDA	in	**EISA HOLD ACKNOWLEDGE:** EISAHLDA is used by the PCEB to inform the ESC that it has been granted ownership of EISA bus. This signal is synchronous to PCICLK.
PEREQ#/ INTA#	in	**PCI TO EISA REQUEST OR INTERRUPT ACKNOWLEDGE:** PEREQ#/INTA# is a dual function signal. The context of the signal pin is determined by the state of EISAHLDA signal. When EISAHLDA is deasserted (0) this signal has the context of Interrupt Acknowledge i.e., if PEREQ#/INTA# is asserted it indicates to the ESC that current cycle on the EISA is an interrupt acknowledge. When EISAHLDA is asserted (1) this signal has the context of PCI-to-EISA Request i.e., if PEREQ#/INTA# is asserted it indicates to the ESC that 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.

2.8.2 PCEB BUFFER COHERENCY CONTROL

Pin Name	Type	Description
NMFLUSH#	t/s	**NEW MASTER FLUSH:** NMFLUSH# is a bi-directional signal which is used to provide handshake between PCEB and ESC to control flushing of system buffers on behalf of EISA masters. During an EISA bus ownership change, before ESC can grant the bus to the EISA master (or DMA) it must ensure that system buffers are flushed and buffers pointing (potentially) towards EISA subsystem are disabled. The ESC asserts NMFLUSH# signal for one PCI clock indicating the request for system buffer flushing. (After driving NMFLUSH# asserted for 1 PCI clock the ESC tri-states NMFLUSH# signal.) When PCEB samples NMFLUSH# asserted it starts immediately to drive NMFLUSH# asserted and initiates internal and external requests for buffer flushing. After all buffers have been flushed (indicated by the proper handshake signals) the PCEB negates NMFLUSH# for 1 PCI clock and stops driving it. When ESC samples signal deasserted that indicates that all system buffers are flushed, it grants EISA bus to an EISA master (or DMA). The ESC resumes responsibility of default NMFLUSH# driver and starts driving NMFLUSH# deasserted until the next time a new EISA master (or DMA) wins arbitration. This signal is synchronous with PCICLK and will be driven negated by the ESC at reset.
INTCHIP0	t/s	**INTER CHIP 0:** INTCHIP0 is a reserved signal. The INTCHIP0 output of the ESC should be connected to the INTCHIP0 input of the PCEB for proper device operation.

2.8.3 DATA SWAP BUFFER CONTROL

Pin Name	Type	Description
SDCPYEN01# SDCPYEN02# SDCPYEN03# SDCPYEN13#	out	**COPY ENABLE:** These active low signals perform byte copy operation on the EISA data bus (SD). These signal are active during mis-matched cycle, and they are used by the PCEB to enable byte copy operation between 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])
SDCPYUP	out	**SYSTEM (DATA) COPY UP:** SDCPYUP is used to control the direction of the byte copy operation. A High on the signal indicates a COPY UP operation where the lower byte lower word of the SD data bus is copied on to the higher byte or higher word 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 the signal to perform the actual data byte copy operation during mis-matched cycles.
SDOE[2:0]#	out	**SYSTEM DATA OUTPUT ENABLES:** SDOE# enables the SD data output of the PCEB Data Swap Buffers on to EISA bus. The ESC activates these signals only during mis-matched cycles. The PCEB uses these signals 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 2 SD[23:16] and byte lane 3 SD[31:24]
SDLE[3:0]#	out	**SYSTEM DATA LATCH ENABLES:** SDLE[3:0]# enable the latching of EISA data bus. These signals are activated only during mis-matched cycles except PCEB initiated write cycle. 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 0 SD[15:8] SDLE2# Latch byte lane 0 SD[23:16] SDLE3# Latch byte lane 0 SD[31:24]

2.9 Integrated Logic Signals

2.9.1 EISA ADDRESS BUFFER CONTROL

Pin Name	Type	Description
SALE#	out	**SA LATCH ENABLE:** SALE# is directly connected to F543s which buffer the LA addresses from the SA addresses. The rising edge of SALE# latches the LA address Bit LA[19:2] to the SA address Bit SA[19:2].
LASAOE#	out	**LA TO SA ADDRESS OUTPUT ENABLE:** LASAOE# is directly connected to the SA output buffer enables of the F543s. The ESC asserts LASAOE# during EISA master cycles. When LASAOE# is asserted, the LA to SA output buffers of the F543s are enabled.
SALAOE#	out	**SA TO LA ADDRESS OUTPUT ENABLE:** SALAOE# is connected to the LA output buffer enables of the F543s. This signal functionally is the exact opposite of LASAOE# signals. The ESC asserts SALAOE# during ISA master cycles. When LASAOE# is asserted, the SA to LA output buffers of the F543s are enabled.

2.9.2 COPROCESSOR INTERFACE

Pin Name	Type	Description
FERR#	in	**NUMERIC CO-PROCESSOR ERROR:** FERR# signal is tied to the Co-processor error signal of the CPU. If FERR# is asserted (Co-processor error detected by the CPU), an internal IRQ13 is be generated and the INT from the ESC will be asserted.
IGNNE#	out	**IGNORE ERROR:** IGNNE# is tied to the ignore numeric error pin of the CPU. IGNNE# is asserted and internal IRQ13 is negated from the falling edge of IOWC# during an I/O write to location 00F0h. IGNNE# will remain asserted until FERR# is negated. Upon reset, this signal is driven negated (high).

2.9.3 BIOS INTERFACE

Pin Name	Type	Description
LBIOSCS#	out	**LATCHED BIOS CHIP-SELECT:** LBIOSCS# indicates that the current address is for the system BIOS. The ESC generates this signal by decoding the EISA LA addresses. The ESC uses a transparent latch to latch the decoded signal. The LBIOSCS# is latched on the falling edge of BALE and qualified with REFRESH#.

2.9.4 KEYBOARD CONTROLLER INTERFACE

Pin Name	Type	Description
KYBDCS#	out	**KEYBOARD CHIP SELECT:** KYBDCS# is connected to the chip select of the 82C42. KYBDCS# is active for I/O addresses 0060h–0064h.
ALTRST#	out	**ALTERNATE RESET:** ALTRST# is used to reset the CPU under program control. This signal is AND'ed together externally with the reset signal (RSTAR#) from the keyboard controller to provide a software means of resetting the CPU. This provides a faster means of reset than is provided by the Keyboard controller. Writing a 1 to Bit 0 in the Port 92 register will cause this signal to pulse active (low) for approximately 4 BCLK's. Before another ALTRST# pulse can be generated, Bit 0 must be written back to a 0. Upon RESET, this signal is driven high (Bit 2 in the Port 92 register is reset low).
ALTA20	out	**ALTERNATE A20:** ALTA20 is used to force A20M# to the CPU low for support of real mode compatible software. This signal is externally OR'ed with the ALTA20 signal from the Keyboard controller and CPURST to control the A20M# input of the CPU. Writing a "0" to Bit 1 of Port 92h register will force ALTA20 inactive (low). This in turn will drive A20M# to the CPU low, if A20GATE from the keyboard controller is also low. Writing a "1" to Bit 1 of the Port 92h register will force ALTA20 active (high), which in turn will drive A20M# to the CPU high, regardless of the state of ALTA20 from the keyboard controller. Upon reset, this signal is driven low.
ABFULL	in	**AUXILIARY BUFFER FULL:** ABFULL is tied directly to the ABFULL signal on the keyboard controller on the system board. This signal indicates that the keyboard controller auxiliary buffer for the mouse interface is full. If the Mouse Interrupt Function bit (offset 4Dh bit 4) is enabled, then the ABFULL signal is connected to the internal IRQ12 (IRQ12 is also available for external use). On a low to high transition on ABFULL the internal IRQ12 is asserted (the internal IRQ12 transitions from low to high if the IRQ12 in the Interrupt controller is programmed for edge triggered mode, the internal IRQ12 is asserted low if the IRQ12 in the interrupt controller is programmed for level triggered mode. A low to high transition on ABFULL will be latched by the ESC. This high level will remain latched internally until an I/O read to port 60h (falling edge of IORC#) or reset (RESET#) has been detected. If this function is not used, ABFULL should be tied low through a 1k resistor.

2.9.5 REAL TIME CLOCK INTERFACE

Pin Name	Type	Description
RTCALE	out	**REAL TIME CLOCK ADDRESS LATCH ENABLE:** RTCALE is directly connected to the system Real Time Clock. The RTC uses this signal to latch the appropriate memory address. A write to port 070h with the appropriate Real Time Clock memory address that will be written to or read from will cause RTCALE to go active.
RTCRD#	out	**REAL TIME CLOCK READ COMMAND:** RTCRD# is asserted for I/O reads from address 0071h. If the Power On Password protection is enabled (I/O Port 92h bit 3 = 1) then for accesses to RTC addresses 36h-3Fh (Port 70h) RTCRD# will not be asserted.
RTCWR#	out	**REAL TIME CLOCK READ COMMAND:** RTCWR# is asserted for I/O writes to address 0071h. If the Power On Password protection is enabled (I/O Port 92h bit 3 = 1) then for accesses to RTC addresses 36h-3Fh (Port 70h) RTCWR# will not be generated.

2.9.6 FLOPPY DISK CONTROLLER INTERFACE

<table>
<tr><th>Pin Name</th><th>Type</th><th>Description</th></tr>
<tr><td>FDCCS#</td><td>out</td><td>FLOPPY DISK CONTROLLER CHIP-SELECT: FDCCS# is asserted for I/O cycles to the floppy drive controller. See Section 11.6 for details of the floppy drive controller decode.</td></tr>
<tr><td>DSKCHG</td><td>in</td><td>DISK CHANGE: DSKCHG signal is tied directly to the DSKCHG signal of the floppy controller. This signal is inverted and driven onto system data line 7 (SD7) during I/O read cycles to floppy address locations 3F7h (primary) or 377h (secondary) as indicated by the table below.
NOTE:
The primary and secondary locations are programmed in the X-Bus Address Decode Enable/Disable Register "A".
<table>
<tr><th>FDCCS#</th><th>IDECSx#</th><th>State of SD7 (output)</th><th>State of XBUSOE#</th></tr>
<tr><td>Decode Enabled</td><td>Decode Enabled</td><td>Tri-stated</td><td>Enabled</td></tr>
<tr><td>Enabled</td><td>Disabled</td><td>Driven via DSKCHG</td><td>Disabled</td></tr>
<tr><td>Disabled</td><td>Enabled</td><td>Tri-stated</td><td>Disabled (note)</td></tr>
<tr><td>Disabled</td><td>Disabled</td><td>Tri-stated</td><td>Disabled</td></tr>
</table>
NOTE:
This mode is not supported because of potential contention between the X-Bus buffer and a floppy on the ISA bus driving the system bus at the same time during shared I/O accesses.
This signal is also used to determine if the floppy controller is present on the X-Bus. It is sampled on the trailing edge of RESET, and if high, the floppy is present. For systems that do not support a floppy via the ESC, this pin should strapped low. If sampled low, the SD7 function, and XBUSOE# will not be enabled for accesses to the floppy disk controller.</td></tr>
<tr><td>DLIGHT#</td><td>out</td><td>FIXED DISK ACTIVITY LIGHT: DLIGHT# is used to control the fixed disk X light. When low, the light is on. When high, the light is off. If either Bit 6 or Bit 7 of the Port 92 register is set to a 1 (Bit 6 and 7 are internally NOR'ed together), DLIGHT# is driven active (low). Setting both Bits 6 and 7 low will cause DLIGHT# to be driven high.</td></tr>
</table>

2.9.7 CONFIGURATION RAM INTERFACE

Pin Name	Type	Description
CRAMRD#	out	**CONFIGURATION RAM READ COMMAND:** CRAMRD# is connected directly to the system Configuration RAM. The ESC asserts CRAMRD# for I/O reads from the address range programmed into the low and high bytes of the configuration RAM command registers.
CRAMWR#	out	**CONFIGURATION RAM WRITE COMMAND:** This is an active Low output. CRAMWR# is connected directly to the system Configuration RAM. The ESC activates CRAMWR# for I/O writes to the address range programmed into the low and high bytes of the configuration RAM command registers.

2.9.8 X-BUS CONTROL AND GENERAL PURPOSE DECODE

Pin Name	Type	Description
XBUSTR#	out	**X-BUS DATA TRANSMIT/RECEIVE:** XBUSTR# is tied directly to the direction control of a 74F245 that buffers the X-Bus data, XD(7:0), from the system data bus, SD(7:0). XBUSTR# is driven high (transmit) for I/O reads from EISA master, ISA master, and DMA cycles. XBUSTR# is also driven high if BIOS space has been decoded for memory cycles. XBUST/R# signal will also be driven high during DMA I/O write cycles for which the X-Bus BIOS has been selected. XBUSTR# is driven low (receive) at all other times. This signal is generated independent of X-Bus peripheral decode. Upon reset, this signal is driven low (receive).
XBUSOE#	out	**X-BUS DATA OUTPUT ENABLE:** XBUSOE# is tied directly to the output enable of a 74F245 that buffers the X-Bus data, XD(7:0), from the system data bus, SD(7:0). XBUSOE# is driven active (low) anytime a ESC supported X-Bus device has been decoded, and the device decode has been enabled in the corresponding configuration registers with the exception of the following conditions. XBUSOE# will not be driven active under the following conditions: (1) during an I/O access to the floppy controller if DSKCHG is sampled low at reset, (2) if the Digital Output Register is programmed to ignore DACK2#, (3) during an I/O read access to floppy location 3F7h (primary) or 377h (secondary) if the IDE decode space is disabled (i.e., IDE is not resident on the X-Bus). The XBUSOE# signal is driven high during any access to a X-Bus peripheral in which its decode space has been disabled through the configuration registers. Upon reset, this signal is negated (high).
GPCS[2:0]# / ECS[2:0]	out	These are dual function signals. The function of these pins is selected through the Mode Select Register bit 4. **GENERAL PURPOSE CHIP SELECT:** GPCS[2:0]# are Chip Selects for peripheral devices. The peripheral devices can be mapped in the I/O range by programming the General Purpose Chip Select Base Address registers and General Purpose Mask registers (offset 64h–6Eh). **ENCODED CHIP SELECT:** ECS[2:0] provide encoded chip select decoding for serial ports, parallel port, IDE and general purpose devices. The device chip selects for the peripheral devices are generated by using a F138 with ECS[2:0] as inputs. Refer to Section 11.9 for details.

2.10 Testing

Pin Name	Type	Description
TEST#	in	**TEST:** TEST# is used to tristate all of the outputs. During normal operation this pin should be tied to ground.

3.0 ESC REGISTER DESCRIPTION

The ESC contains ESC configuration registers, DMA registers, Timer Unit registers, Interrupt Unit registers, and EISA configuration registers. All of the registers are accessable from the EISA bus. During a reset the ESC sets its internal registers to predetermined default states. The default values are indicated in the individual register descriptions.

3.1 ESC Configuration Registers

ESC configuration registers are accessed through an indexing scheme. The index address register is located at I/O address 0022h, and the index data register is located at I/O address 0023h. The offset (data) written into the index address register selects the desired configuration register. Data for the selected configuration register can be read from or written to by performing a read or a write to the index data register. See Table 4-3 for a summary of configuration register index addresses.

3.1.1 ESCID—ESC ID REGISTER

Register Name: ESC ID
Register Location: 02h
Default Value: 00h
Attribute: Read/Write
Size: 8 bits

Since the ESC configuration registers are accessed by the index addressing mechanism using I/O Ports 22h, and 23h, it is possible that another device in the system might use the same approach for configuration. In order to avoid contention with similar index register devices, the ID register must be written with 0Fh. The ESC will not respond to accesses to any other configuration register until the ID byte has been written in the ESC ID Register.

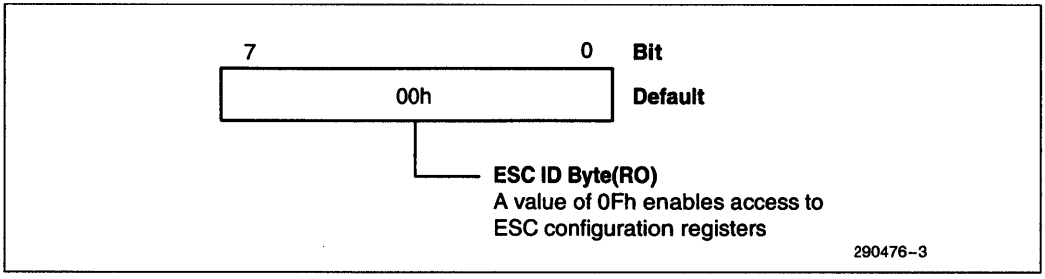


Figure 3-1. ESC ID Register

Table 3-1. ESC ID Register

Bit #	Description
7:0	**ESC ID BYTE:** These bits must be written to a value of 0Fh before the ESC will respond to any other configuration register access. After a reset has occurred all of the configuration registers, except this register, are disabled.

3.1.2 RID—REVISION ID REGISTER

Register Name: Revision ID
Register Offset: 08h
Default Value: 00h
Attribute: Read only
Size: 8 bits

This 8-bit register contains device stepping information. Writes to this register have no effect.

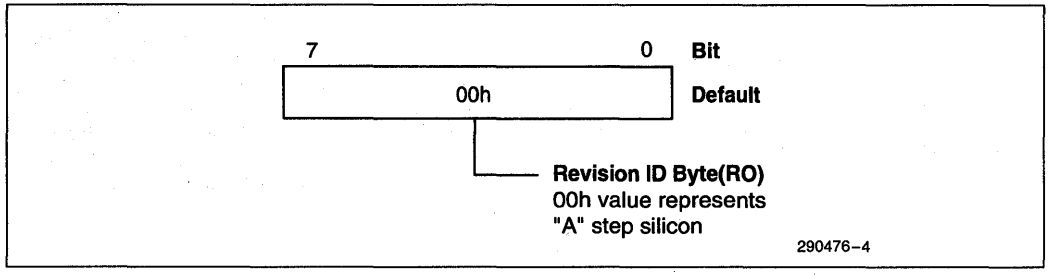


Figure 3-2. Revision ID Register

Table 3-2. Revision ID Register

Bit #	Description
7:0	**REVISION ID BYTE:** These bits contain the stepping information about the device. The register is hardwired during manufacturing. The register is read only. Writes have no affect on the register value.

3.1.3 MS—MODE SELECT REGISTER

Register Name: Mode Select
Register Location: 40h
Default Value: 20h
Attribute: Read/Write
Size: 8 bits

This register selects the various functional modes of the ESC.

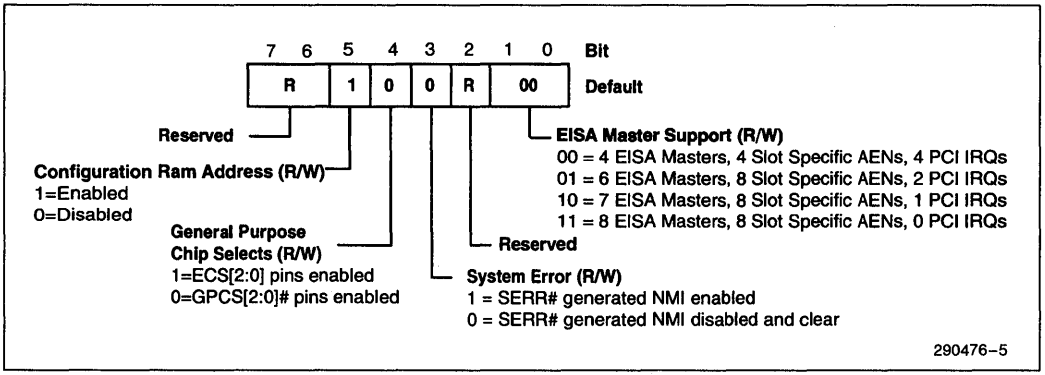


Figure 3-3. Mode Select Register

Table 3-3. Mode Select Register

Bit #	Description
7:6	**RESERVED.**
5	**CONFIGURATION RAM ADDRESS:** This bit is used to enable or disable the Configuration RAM Page Address (CPG[4:0]) generation. If this bit is set to 1, accesses to the Configuration RAM space will generate the RAM page address on the LA[31:27] # pins. If this bit is set to 0, the CPG[4:0] signals will not be activated. The default for this is bit is "1".
4	**GENERAL PURPOSE CHIP SELECTS:** This bit is used to select the functionality of the GPCS[2:0] # /ECS[2:0] pins. If the bit is set to 0, the GPCS[2:0] # functionality is selected. If the bit is set to 1, the ESC[2:0] functionality is selected.
3	**SYSTEM ERROR:** This bit is used to disable or enable the generation of NMI based on SERR # signal pulsing active.
2	**RESERVED.**
1:0	**EISA MASTER SUPPORT:** These bit combinations determine the functionality of pins AEN[4:1]/ EAEN[4:1], MACK[3:0] # /EMACK[3:0], and MREQ[7:4] # /PIRQ[0:3] #. Bit[1,0] = 00 selects 4 EISA Masters, 4 Slot Specific AENs, and 4 PCI IRQs. For this combination the pin functionality is AEN[4:1], MACK[3:0] #, and PIRQ[0:3] #. For any other combination of Bit[1,0] the encoded pins are selected i.e., EAEN[4:1], and EMACK[3:0].

3.1.4 BIOSCSA—BIOS CHIP SELECT REGISTER

Register Name: BIOS Chip Select A
Register Location: 42h, 43h
Default Value: 10h, 00h
Attribute: Read/Write
Size: 8 bits

The LBIOSCS# signal is used to decode access to the motherboard BIOS. The ESC decodes memory access to the following address ranges, and if the range has been enabled the LBIOSCS# signal is always asserted for memory reads in the enabled BIOS range. If the BIOS Write Enable bit is set in the configuration register BIOSCSB, the LBIOSCS# is also asserted for memory write cycles.

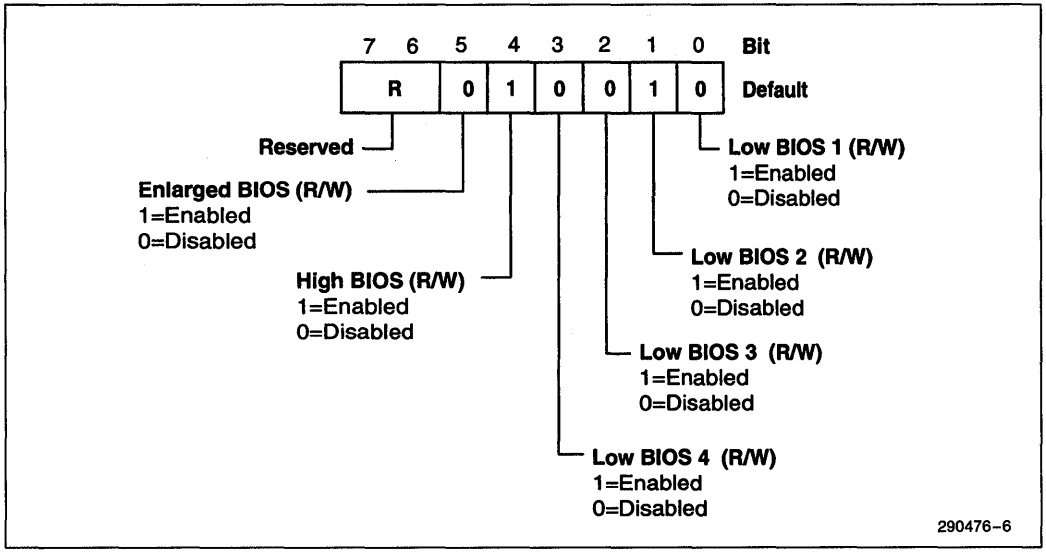


Figure 3-4. BIOSCSA Register

Table 3-4. BIOSCSA Register

Bit #	Description
7:6	**RESERVED.**
5	**ENLARGED BIOS:** During Memory access to locations FFF80000h–FFFDFFFFh with this bit set, LBIOSCS# will be asserted for memory read cycles. If bit 3 of BIOSCSB is set then LBIOSCS# will be asserted for write cycles as well.
4	**HIGH BIOS:** During Memory access to locations 0F0000h–0FFFFFh, FF0000h–FFFFFFh, FFFF0000h–FFFFFFFFh with this bit set, LBIOSCS# will be asserted for memory read cycles. If bit 3 of BIOSCSB is set then LBIOSCS# will be asserted for write cycles as well.
3	**LOW BIOS 4:** During Memory access to locations 0EC000h–0EFFFFh, FFEEC000h–FFEEFFFFh, FFFEC000h–FFFEFFFFh with this bit set, LBIOSCS# will be asserted for memory read cycles. If bit 3 of BIOSCSB is set then LBIOSCS# will be asserted for write cycles as well.
2	**LOW BIOS 3:** During Memory access to locations 0E8000h–0EBFFFh, FFEE8000h–FFEEBFFFh, FFFE8000h–FFFEBFFFh with this bit set, LBIOSCS# will be asserted for memory read cycles. If bit 3 of BIOSCSB is set then LBIOSCS# will be asserted for write cycles as well.

Table 3-4. BIOSCSA Register (Continued)

Bit #	Description
1	**LOW BIOS 2:** During Memory access to locations 0E4000h–0E7FFFh, FFEE4000h–FFEE7FFFh, FFFE4000h–FFFE7FFFh with this bit set, LBIOSCS# will be asserted for memory read cycles. If bit 3 of BIOSCSB is set then LBIOSCS# will be asserted for write cycles as well.
0	**LOW BIOS 1:** During Memory access to locations 0E0000h–0E3FFFh, FFEE0000h–FFEE3FFFh, FFFE0000h–FFFE3FFFh with this bit set, LBIOSCS# will be asserted for memory read cycles. If bit 3 of BIOSCSB is set then LBIOSCS# will be asserted for write cycles as well.

3.1.5 BIOSCSB—BIOS CHIP SELECT REGISTER

Register Name: BIOS Chip Select B
Register Location: 42h, 43h
Default Value: 10h, 00h
Attribute: Read/Write
Size: 8 bits

The LBIOSCS# signal is used to decode access to the motherboard BIOS. The ESC decodes memory access to the following address ranges, and if the range has been enabled the LBIOSCS# signal is always asserted for memory reads in the enabled BIOS range. If the BIOS Write Enable bit is set in the configuration register BIOSCSB, the LBIOSCS# is also asserted for memory write cycles.

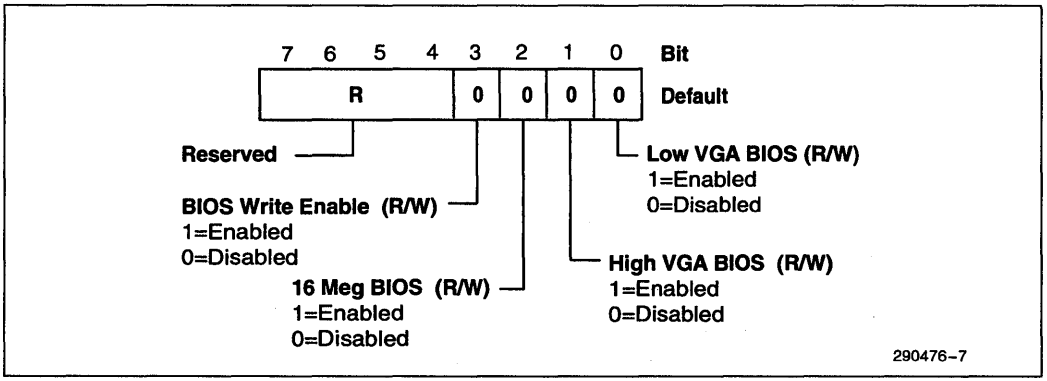


Figure 3-5. BIOSCSB Register

Table 3-5. BIOSCSB Register

Bit #	Description
7:4	**RESERVED.**
3	**BIOS WRITE ENABLE:** When enabled LBIOSCS# is asserted for memory read AND write cycles for addresses in the decoded and enabled BIOS range, otherwise LBIOSCS# is asserted for memory read cycles ONLY.
2	**16 MEG BIOS:** During Memory access to locations FF0000h–FFFFFFh with this bit set, LBIOSCS# will be asserted for memory read cycles. If bit 3 of BIOSCSB is set then LBIOSCS# will be asserted for write cycles as well.
1	**HIGH VGA BIOS:** During Memory access to locations 0C4000h–0C7FFFh with this bit set, LBIOSCS# will be asserted for memory read cycles. If bit 3 of BIOSCSB is set then LBIOSCS# will be asserted for write cycles as well.
0	**LOW VGA BIOS:** During Memory access to locations 0C0000h–0C3FFFh with this bit set, LBIOSCS# will be asserted for memory read cycles. If bit 3 of BIOSCSB is set then LBIOSCS# will be asserted for write cycles as well.

3.1.6 CLKDIV—EISA CLOCK DIVISOR REGISTER

Register Name: EISA Clock Divisor
Register Location: 4Dh
Default Value: xx00x000b
Attribute: Read/Write
Size: 8 bits

This register is used to select the integer value used to divide the PCI clock (PCICLK) to generate the EISA Bus Clock (BCLK). In addition, the register provides a bit to enable/disable the ABFULL function, and a bit to enable/disable the co-processor error support.

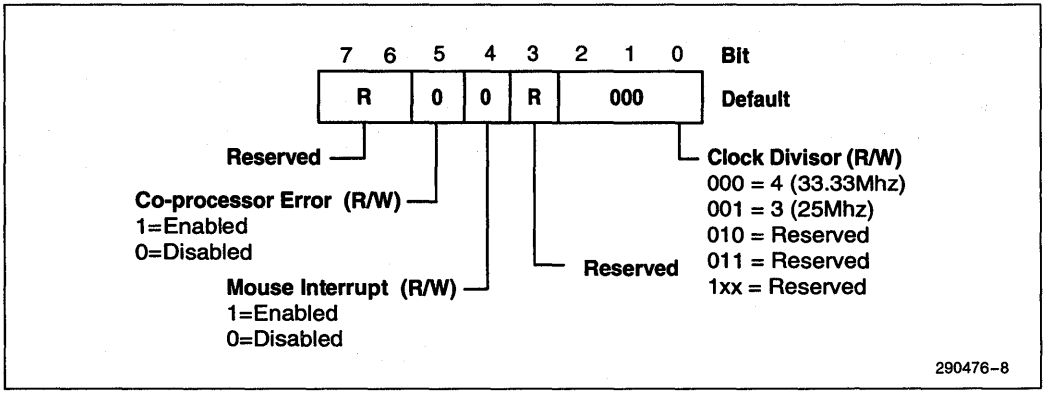


Figure 3-6. EISA Clock Divisor Register

Table 3-6. EISA Clock Divisor Register

Bit #	Description
7:6	**RESERVED.**
5	**CO-PROCESSOR ERROR:** The state of this bit determines if the FERR# signal is connected to the ESC internal IRQ13 interrupt signal. If this bit is set to "1", the ESC will assert IRQ13 to the interrupt controller if FERR# signal is asserted. If this bit is set to "0", then the FERR# signal is ignored by the ESC (i.e., this signal is not connected to any logic in the ESC).
4	**MOUSE INTERRUPT:** The state of this bit determines if the ABFULL signal is connected to the ESC internal IRQ12 interrupt signal. If this bit is set to "1", a low to high transition on the ABFULL signal will generate interrupt on the IRQ12 signal. If this bit is set to "0", then the ABFULL signal is ignored by the ESC (i.e., this signal is not connected to any logic in the ESC).
3	**RESERVED.**
2:0	**CLOCK DIVISOR:** These bits are used to select the integer that is used to divide the PCICLK down to generate the BCLK. Upon reset, these bits are set to 000b (divisor or 4).

Bit 2	Bit 1	Bit 0	Divisor	BCLK
0	0	0	4 (33.33 MHz)	8.33 MHz
0	0	1	3 (25 MHz)	8.33 MHz
0	1	0	Reserved	
0	1	1	Reserved	
1	x	x	Reserved	

3.1.7 PCSA—PERIPHERAL CHIP SELECT A REGISTER

Register Name: Peripheral Chip Select A
Register Location: 4Eh
Default Value: xx000111b
Attribute: Read/Write
Size: 8 bits

This register is used to enable or disable accesses to the RTC, keyboard controller, Floppy Disk controller, and IDE. Disabling any of these bits will prevent the chip select and X-Bus transceiver control signal (XBUSOE#) for that device from being generated. This register is also used to select which address range (primary or secondary) will be decoded for the resident floppy controller and IDE. This insures that there is no contention with the X-Bus transceiver driving the system data bus during read accesses to these devices.

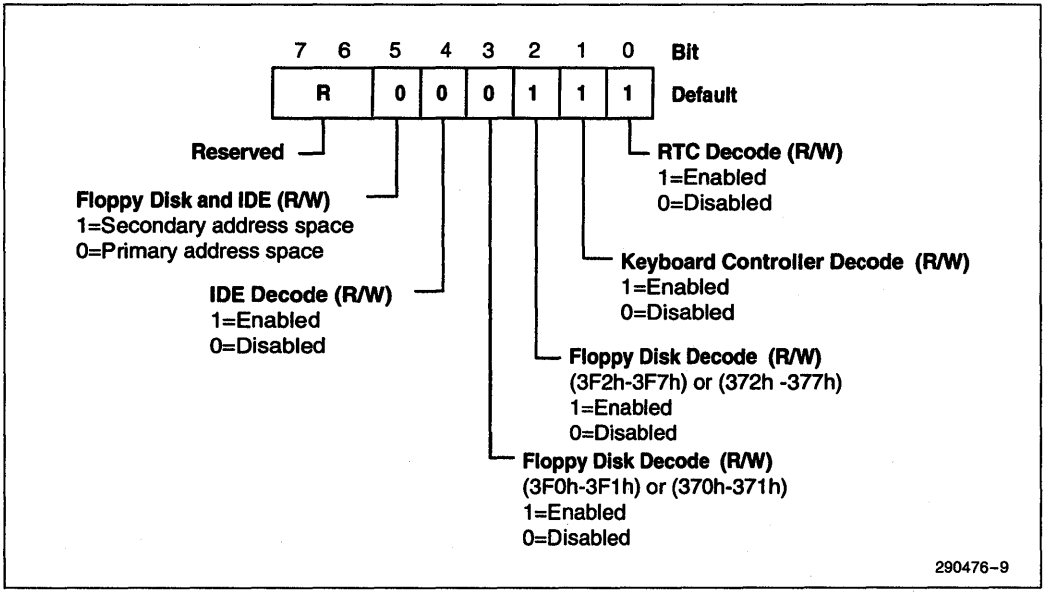


Figure 3-7. PCSA Register

Table 3-7. PCSA Register

<table>
<tr><th>Bit #</th><th>Description</th></tr>
<tr><td>7:6</td><td>RESERVED.</td></tr>
<tr><td>5, 3:2</td><td>FLOPPY DISK AND IDE, FLOPPY DISK DECODES: Bits 2 and 3 are used to enable or disable the floppy locations as indicated. Bit 2 defaults to enabled (1) and bit 3 defaults to disabled (0) when a reset occurs. Bit 5 is used to select between the primary and secondary address range used by the Floppy Controller and the IDE. Only primary or only secondary can be programmed at any one time. This bit defaults to primary (0).
The following table shows how these bits are used to select the floppy controller:
<table>
<tr><th>Address</th><th>Bit 2</th><th>Bit 3</th><th>Bit 5</th><th>DSKCHG</th><th>FDCCS#</th></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>0</td><td>1</td></tr>
<tr><td>3F0h, 3F1h</td><td>X</td><td>1</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>3F2h–3F7h</td><td>1</td><td>X</td><td>0</td><td>1</td><td>0 (note)</td></tr>
<tr><td>370h, 371h</td><td>X</td><td>1</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td>372h–37Fh</td><td>1</td><td>X</td><td>1</td><td>1</td><td>0 (note)</td></tr>
</table>
NOTE:
If IDE decode is enabled, all accesses to locations 03F6h and 03F7h (primary) or 0376h and 0377h (secondary) will result in decode for IDECS1# (FDCCS# will not be generated). An external AND gate can be used to tie IDECS1# and FDCCS# together to insure that the floppy is enabled for these accesses (refer to Figure 17-1).</td></tr>
<tr><td>4</td><td>IDE DECODE: Bit 4 is used to enable or disable IDE locations 1F0h–1F7h (primary) or 170h–177h (secondary) and 3F6h, 3F7h (primary) or 376h, 377h (secondary). When this bit is set to 0, the IDE encoded chip select signals and the X-Bus transceiver signal (XBUSOE#) are not generated for these addresses.</td></tr>
<tr><td>1</td><td>KEYBOARD CONTROLLER DECODE: Enables (1) or disables (0) the Keyboard Controller address locations 60h, 62h, 64h, and 66h. When this bit is set to 0, the Keyboard Controller encoded chip select signals and the X-Bus transceiver signal (XBUSOE#) are not generated for these locations.</td></tr>
<tr><td>0</td><td>REAL TIME CLOCK DECODE: Enables (1) or disables (0) the RTC address locations 70h - 77h. When this bit is set to 0, the RTC encoded chip select signals RTCALE, RTCRD, RTCWR#, and XBUSOE# signals are not generated for these addresses.</td></tr>
</table>

3.1.8 PCSB—PERIPHERAL CHIP SELECT B REGISTER

Register Name: Peripheral Chip Select B
Register Location: 4Fh
Default Value: CFh
Attribute: Read/Write
Size: 8 bits

This register is used to enable or disable generation of the X-Bus transceiver signal (XBUSOE#) for accesses to the serial ports and parallel port locations. When disabled, the XBUSOE# signal for that device will not be generated.

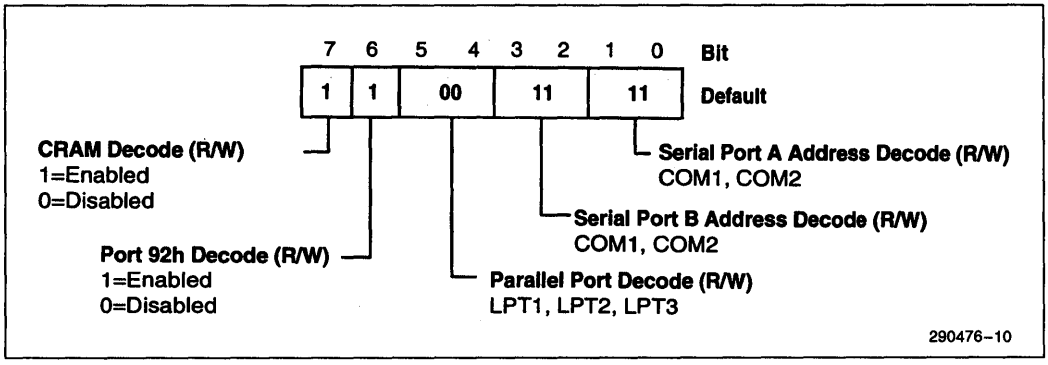


Figure 3-8. Peripheral Chip Select B Register

Table 3-8. Peripheral Chip Select B Register

Bit #	Description
7	**CRAM DECODE:** This bit is used to enable (1) or disable (0) I/O write accesses to location 0C00h and I/O read/write accesses to locations 0800h-08FFh. The configuration RAM read and write (CRAMRD#, CRAMWR#) strobes are valid for accesses to 0800h-08FFh.
6	**Port 92 DECODE:** This bit is used to disable (0) access to Port 92. This bit defaults to enable (1) at PCIRST.
5:4	**PARALLEL PORT DECODE:** These bits are used to select which Parallel Port address range (LPT1, 2, or 3) is decoded. **Bit 5 Bit 4** 0 0 LPT1 (3BCh–3BFh) 0 1 LPT2 (378h–37Fh) 1 0 LPT3 (278h–27Fh) 1 1 Disabled
3:2	**SERIAL PORT B ADDRESS DECODE:** If either COM1 or COM2 address ranges are selected. These bits default to disabled upon PCIRST. **Bit 3 Bit 2** 0 0 3F8h–3FFh (COM1) 0 1 2F8h–2FFh (COM2) 1 0 Reserved 1 1 Port A disabled
1:0	**SERIAL PORT A ADDRESS DECODE:** If either COM1 or COM2 address ranges are selected. These bits default to disabled upon PCIRST. **Bit 1 Bit 0** 0 0 3F8h–3FFh (COM1) 0 1 2F8h–2FFh (COM2) 1 0 Reserved 1 1 Port A disabled

3.1.9 EISAID[4:1]—EISA ID REGISTERS

Register Name: EISA ID Register (Byte 1, Byte 2, Byte 3, Byte 4)
Register Offset: 50h, 51h, 52h, 53h
Default Value: 00h, 00h, 00h, 00h
Attribute: Read/Write only
Size: 8 bits

These 8-bit registers contain the EISA motherboard ID. The data in the register is reflected on the data bus for I/O cycles addressed to 0C80h–0C83h respectively.

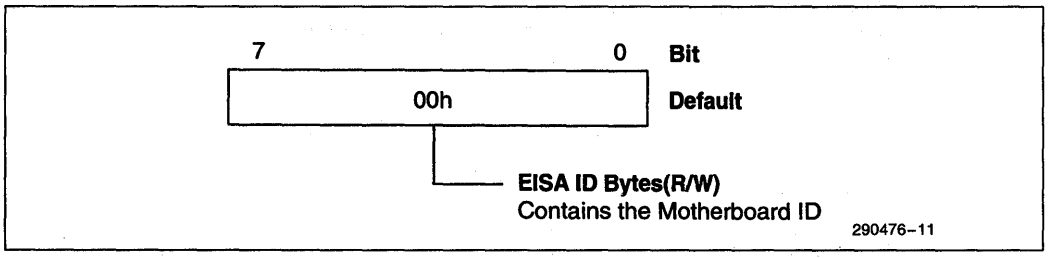


Figure 3-9. EISA ID Registers

Table 3-9. EISA ID Registers

Bit #	Description
7:0	**EISA ID BYTE:** These bits contain the EISA Motherboard ID information. On Power up these bits default to 00h. These bits are written with the ID value during configuration. The value of these bits are reflected in I/O registers 0C80h–0C83h.

3.1.10 SGRBA—SCATTER-GATHER RELOCATE BASE ADDRESS REGISTER

Register Name: S-G Relocate Base Address
Register Location: 57h
Default Value: 04h
Attribute: Read/Write
Size: 8 bits

The value programmed in this register determines the high order I/O address of the S-G registers. The default value is 04h.

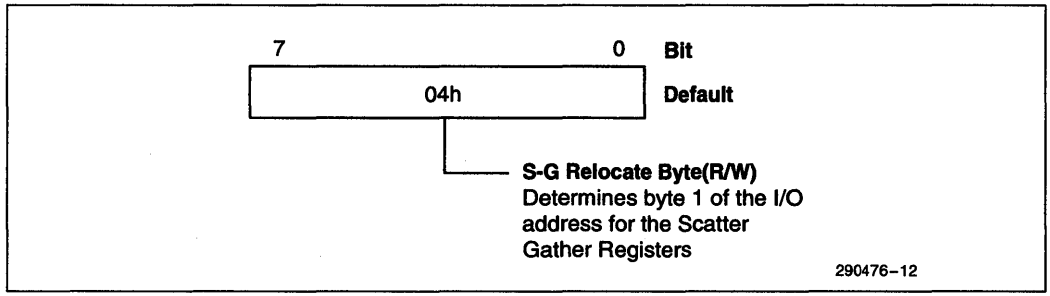


Figure 3-10. S-G Relocate Base Address Register

Table 3-10. S-G Relocate Base Address Register

Bit #	Description
7:0	**S-G RELOCATE BYTE:** These bits determine the I/O location of the Scatter-Gather Registers. The Scatter-Gather register relocation range is xx10h–xx3Fh (default 0410h - 043Fh). These bits determine the Byte 1 of the I/O address. Address signals LA[15:8] are compared against the contents of this register (Bit[7:0]) to determine I/O accesses to the Scatter-Gather registers. The default on Power up is 04h.

3.1.11 PIRQ[0:3]#—PIRQ ROUTE CONTROL REGISTERS

Register Name: PIRQ0#, PIRQ1#, PIRQ2#, PIRQ3# Route Control

Register Location: 60h, 61h, 62h, 63h

Default Value: 80h

Attribute: Read/Write

Size: 8 bits

These registers control the routing of PCI Interrupts (PIRQ[0:3]#) to the PC compatible Interrupts. Each PCI interrupt can be independently routed to 1 of 11 compatible interrupts.

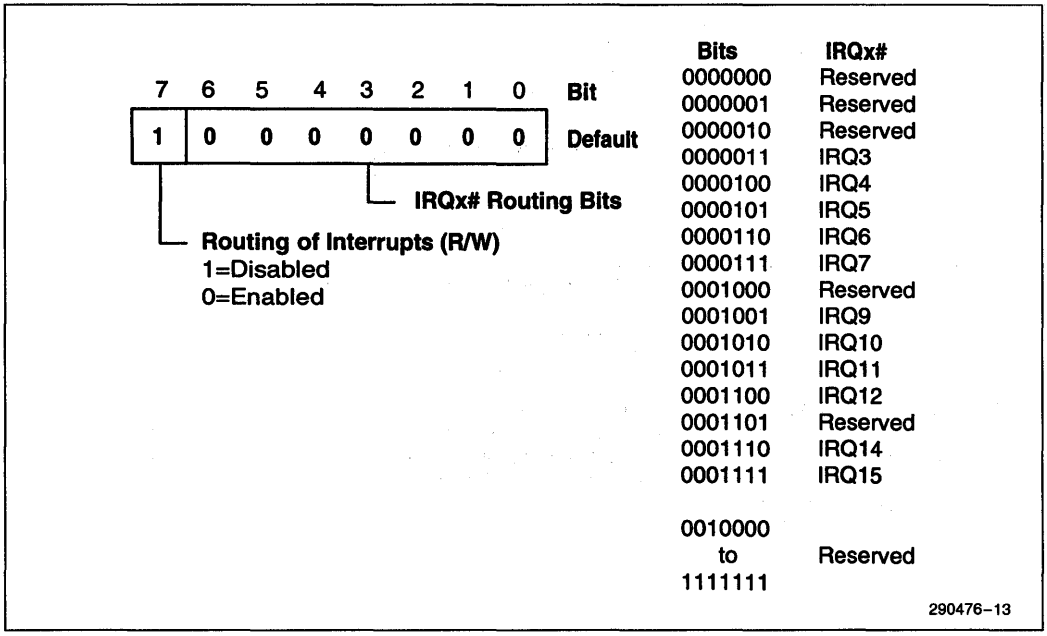


Figure 3-11. PIRQ Route Control Registers

Table 3-11. PIRQ Route Control Register

Bit #	Description
7	**ROUTING OF INTERRUPTS:** When enabled this bit routes the PCI Interrupt signal to the PC compatible interrupt signal specified in bits[6:0]. After a reset or a power-on this bit is disabled (set to 1).
6:0	**IRQx# ROUTING BITS:** These bits specify which IRQ signal to generate when the PCI Interrupt for this register has been triggered.

3.1.12 GPCSLA[2:0]—GENERAL PURPOSE CHIP SELECT LOW ADDRESS REGISTER

Register Name: General Purpose Chip Select Low Address

Register Location: 64h, 68h, 6Ch

Default Value: 00h

Attribute: Read/Write

Size: 8 bits

This register contains the low byte of the General Purpose Peripheral mapping address. The contents of this register are compared with the LA[7:0] address lines. The contents of this register, the GPCSHA Register and the GPCSM Register control the generation the GPCS[2:0]# signal or the ESC[2:0] signal (101, 110 combination). If Mode Select Register (offset 40h) bit 4 = 1, offset register 6Ch is ignored.

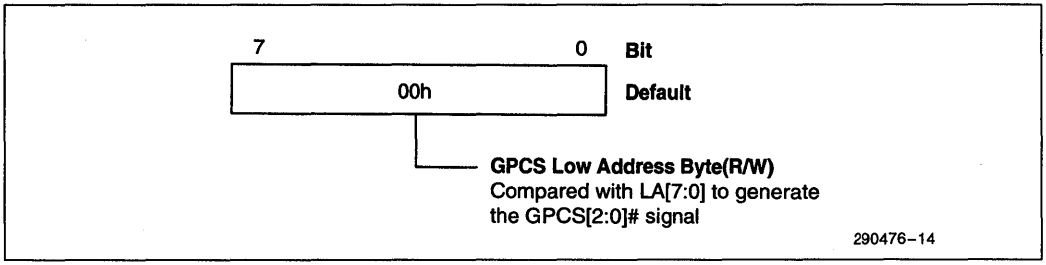


Figure 3-12. General Purpose Chip Select Low Address Byte Register

Table 3-12. General Purpose Chip Select Low Address Byte Register

Bit #	Description
7:0	**GPCS LOW ADDRESS BYTE:** The contents of these bits are compared with the address lines LA[7:0] to generate the GPCS[2:0]# signal or the ECS[2:0] combination for this register. The mask register (GPCSM[2:0]) determines which bits to use during the comparison.

3.1.13 GPCSHA[2:0]—GENERAL PURPOSE CHIP SELECT HIGH ADDRESS REGISTER

Register Name: General Purpose Chip Select High Address

Register Location: 65h, 69h, 6Dh

Default Value: C0h

Attribute: Read/Write

Size: 8 bits

This register contains the high byte of the General Purpose Peripheral mapping address. The contents of this register are compared with the LA[15:8] address lines. The contents of this register, the GPCSLA Register and the GPCSM Register control the generation the GPCS[2:0]# signal or the ESC[2:0] signal (101, 110 combination). If Mode Select Register (offset 40h) bit 4 = 1, offset register 6Dh is ignored.

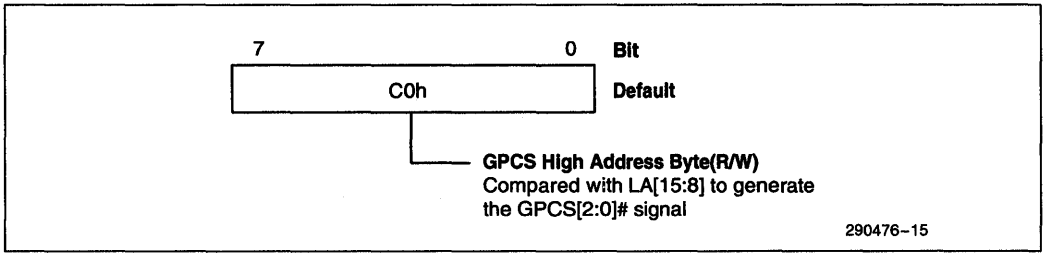


Figure 3-13. General Purpose Chip Select High Address Byte Register

Table 3-13. General Purpose Chip Select High Address Byte Register

Bit #	Description
7:0	**GPCS HIGH ADDRESS BYTE:** The contents of these bits are compared with the address lines LA[15:8] to generate the GPCS[2:0]# signal or the ECS[2:0] combination for this register.

3.1.14 GPCSM[2:0]—GENERAL PURPOSE CHIP SELECT MASK REGISTER

Register Name: General Purpose Chip Select Mask Register

Register Location: 66h, 6Ah, 6Eh

Default Value: 00h

Attribute: Read/Write

Size: 8 bits

This register contains the mask bits for determining the address range for which the GPCSx# signals are generated. If a register bit is set to a 1 then the corresponding bit in the GPCSL register is not compared with the address signal in the generation of the GPCSx# signals. If Mode Select Register (offset 40h) bit 4 = 1, offset register 6Eh is ignored.

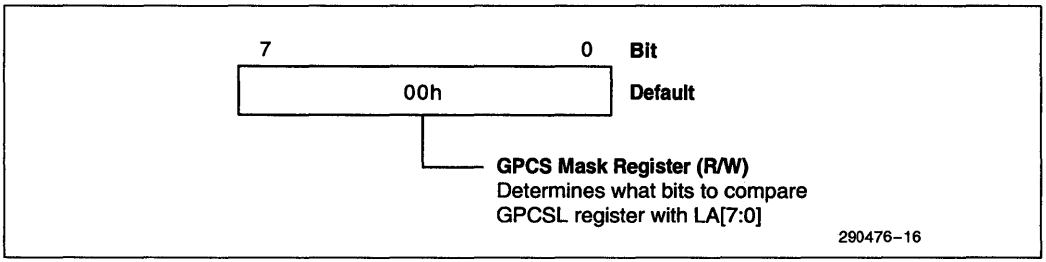


Figure 3-14. General Purpose Chip Select Mask Register

Table 3-14. General Purpose Chip Select Mask Register

Bit #	Description
7:0	**GPCS MASK REGISTER:** The contents of these bits are used to determine which bits to compare GPCSLA[2:0] with the address lines LA[7:0]. A 1 bit means the bit should not be compared.

3.1.15 GPXBC—GENERAL PURPOSE PERIPHERAL X-BUS CONTROL REGISTER

Register Name: General Purpose Peripheral X-Bus Control
Register Location: 6Fh
Default Value: xxxx x000b
Attribute: Read/Write
Size: 8 bits

The register controls the generation of the X-Bus buffer output enable (XBUSOE#) signal for I/O accesses to the peripherals mapped in the General Purpose Chip Select address decode range. This register determines if the General Purpose Peripheral is placed on the X-Bus or not. If the General Purpose Peripheral is on the X-Bus than the corresponding bit is set to "1". Otherwise the bit is set to "0".

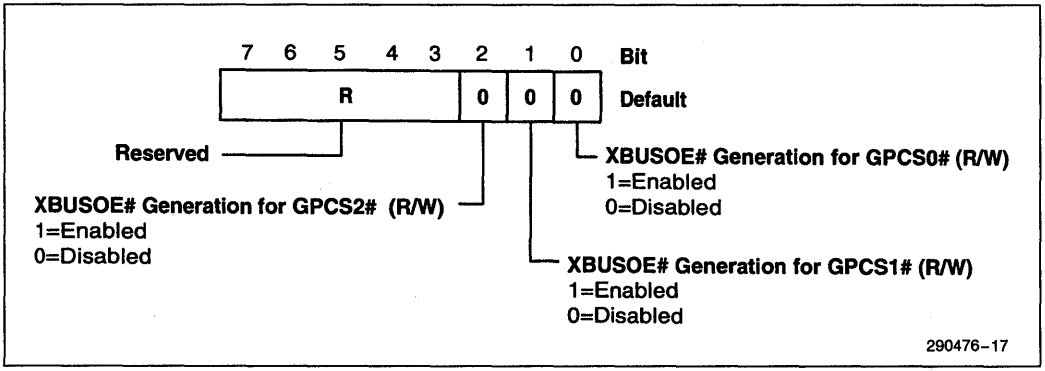


Figure 3-15. General Purpose Peripheral X-Bus Control Register

Table 3-15. General Purpose Peripheral X-Bus Control Register

Bit #	Description
7:3	**RESERVED.**
2	**XBUSOE# GENERATION FOR GPCS2#:** When this bit is enabled XBUSOE# will be generated when GPCS2# is generated.
1	**XBUSOE# GENERATION FOR GPCS1#:** When this bit is enabled XBUSOE# will be generated when GPCS1# is generated.
0	**XBUSOE# GENERATION FOR GPCS0#:** When this bit is enabled XBUSOE# will be generated when GPCS0# is generated.

3.1.16 TEST CONTROL REGISTER

Register Name: Test Control Register
Register Location: 88h
Default Value: 00h
Attribute: Read/Write
Size: 8 bits

This register provides control for ESC manufacturing test modes. The functionality of this register is reserved.

3.2 DMA Register Description

The ESC contains DMA circuitry that incorporates the functionality of two 82C37 DMA controllers (DMA1 and DMA2). The DMA registers control the operation of the DMA controllers and are all accessible from the EISA Bus. This section describes the DMA registers. Unless otherwise stated, a reset sets each register to its default value. The operation of the DMA is further described in Chapter 6.0, DMA Controller.

3.2.1 DCOM—COMMAND REGISTER

Register Name: DMA Command
Register Location: 08h—Channels 0–3
0D0h—Channels 4–7
Default Value: 00000000b
Attribute: Write Only
Size: 8 bits

This 8-bit register controls the configuration of the DMA. It is programmed by the microprocessor in the Program Condition and is cleared by reset or a Master Clear instruction. Note that disabling Channels 4–7 will also disable Channels 0–3, since Channels 0–3 are cascaded onto Channel 4. The DREQ and DACK# channel assertion sensitivity is assigned by channel group, not per individual Channel. For priority resolution the DMA consists of two logical channel groups—Channels 0–3 (Controller 1–DMA1) and Channels 4–7 (Controller 2–DMA2). Both groups may be assigned fixed priority, one group can be assigned fixed priority and the second rotating priority, or both groups may be assigned rotating priority. A detailed description of the channel priority scheme is found in the DMA functional description, Section 6.5. Following a reset or DMA Master Clear, both DMA-1 and DMA-2 are enabled in fixed priority, the DREQ sense level is active high, and the DACK# assertion level is active low.

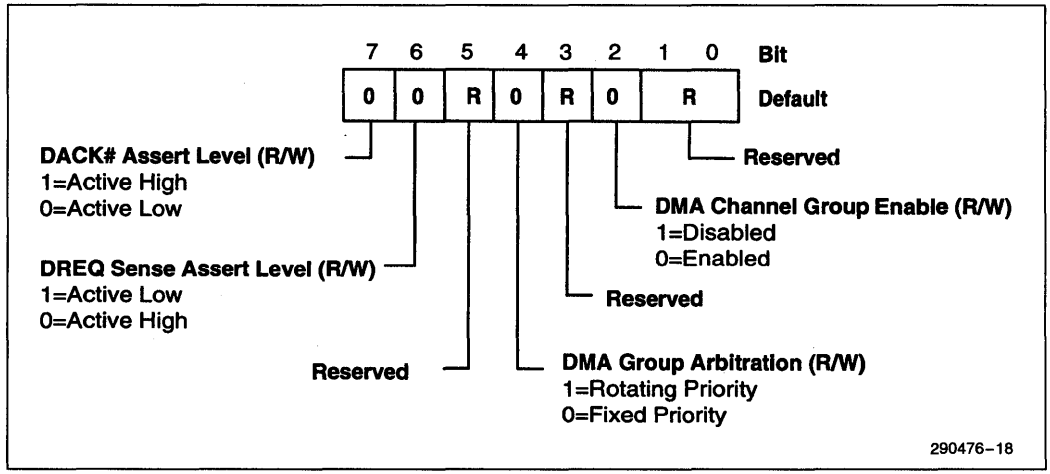


Figure 3-16. DMA Command Register

Table 3-16. DMA Command Register

Bit #	Description
7	**DACK# ASSERT LEVEL:** Bit 7 controls the DMA channel request acknowledge (DACK#) assertion level. Following reset, the DACK# assertion level is active low. The low level indicates recognition and acknowledgement of the DMA request to the DMA slave requesting service. Writing a 0 to Bit 7 assigns active low as the assertion level. When a 1 is written to this bit, a high level on the DACK# line indicates acknowledgement of the request for DMA service to the DMA slave.
6	**DREQ SENSE ASSERT LEVEL:** Bit 6 controls the DMA channel request (DREQ) assertion detect level. Following reset, the DREQ sense assert level is active high. In this condition, an active high level sampled on DREQ is decoded as an active DMA channel request. Writing a 0 to Bit 6 assigns active high as the sense assert level. When a 1 is written to this bit, a low level on the DREQ line is decoded as an active DMA channel request.
5, 3, 1:0	**RESERVED:** Must be 0.
4	**DMA GROUP ARBITRATION:** Each channel group is individually assigned either fixed or rotating arbitration priority. At reset, each group is initialized in fixed priority. Writing a 0 to Bit 4 assigns fixed priority to the channel group, while writing a 1 assigns rotating priority to the group.
2	**DMA GROUP ENABLE:** Writing a 1 to this bit disables the DMA channel group, while writing a 0 to this bit enables the DMA channel group. Both channel groups are enabled following reset. Disabling Channel group 4–7 also disables Channel group 0–3, which is cascaded through Channel 4.

3.2.2 DCM—DMA CHANNEL MODE REGISTER

Register Name: DMA Channel Mode

Register Location: 0Bh—Channels 0–3
0D6h—Channels 4–7

Default Value: 000000xxb

Attribute: Write Only

Size: 6 bits

Each channel has a 6-bit Mode register associated with it. The Mode registers provide control over DMA Transfer type, transfer mode, address increment/decrement, and autoinitialization. When writing to the register, Bits [1:0] determine which channel's Mode register will be written and are not stored. Only Bits [7:2] are stored in the mode register. This register is set to the default value upon reset and Master Clear. Its default value is Verify transfer, Autoinitialize disable, Address increment, and Demand mode. Channel 4 defaults to cascade mode and cannot be programmed for any mode other than cascade mode.

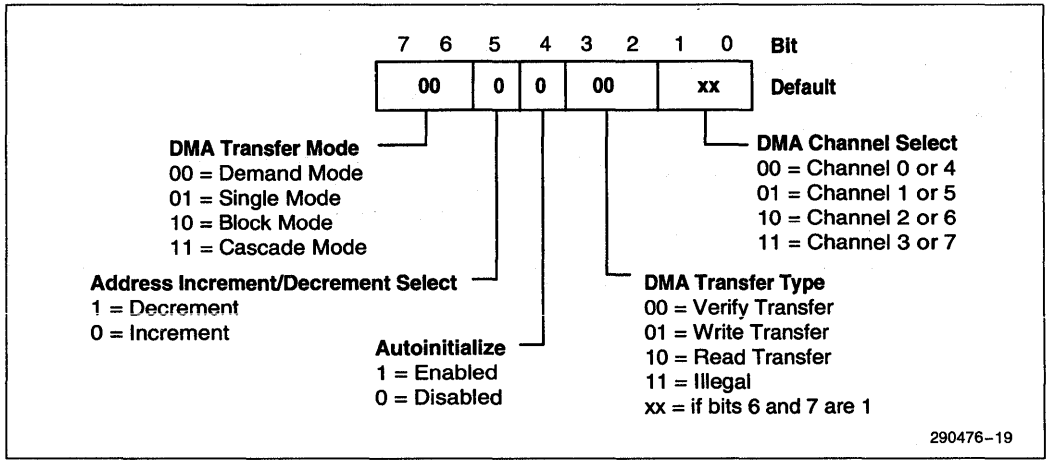


Figure 3-17. DMA Channel Mode Register

Table 3-17. DMA Channel Mode Register

Bit #	Description
7:6	**DMA TRANSFER MODE:** Each DMA channel can be programmed in one of four different modes: single transfer, block transfer, demand transfer and cascade. **Bit 7** **Bit 6** **Transfer Mode** 0 0 Demand mode 0 1 Single mode 1 0 Block mode 1 1 Cascade mode
5	**ADDRESS INCREMENT/DECREMENT SELECT:** Bit 5 controls address increment/decrement during multi-byte DMA transfers. When bit 5 = 0, address increment is selected. When bit 5 = 1, address decrement is selected. Address increment is the default after a PCIRST# cycle or Master Clear command.
4	**AUTOINITIALIZE ENABLE:** When bit 4 = 1, the DMA restores the Base Page, Address, and Word count information to their respective current registers following a terminal count (TC). When bit 4 = 0, the autoinitialize feature is disabled and the DMA does not restore the above mentioned registers. A PCIRST# or Master Clear disables autoinitialization (sets bit 4 to 0).
3:2	**DMA TRANSFER TYPE:** Verify, write and read transfer types are available. Verify transfer is the default transfer type upon PCIRST# or Master Clear. Write transfers move data from an I/O device to memory. Read transfers move data from memory to an I/O device. Verify transfers are pseudo transfers; addresses are generated as in a normal read or write transfer and the device responds to EOP etc. However, with Verify transfers, the ISA memory and I/O cycle lines are not driven. Bit combination 11 is illegal. When the channel is programmed for cascade ([7:6] = 11) the transfer type bits are irrelevant. **Bit 3** **Bit 2** **Transfer Type** 0 0 Verify transfer 0 1 Write transfer 1 0 Read transfer 1 1 Illegal
1:0	**DMA CHANNEL SELECT:** Bits [1:0] select the DMA Channel Mode Register that will be written by bits [7:2]. **Bit 1** **Bit 0** **Channel** 0 0 Channel 0 (4) 0 1 Channel 1 (5) 1 0 Channel 2 (6) 1 1 Channel 3 (7)

3.2.3 DCEM—DMA CHANNEL EXTENDED MODE REGISTER

Register Name: DMA Channel Extended Mode

Register Location: 040Bh—Channels 0–3
04D6h—Channels 4–7

Default Value: 000000xxb

Attribute: Write Only

Size: 6 bits

Each channel has a 6-bit Extended Mode register associated with it. The register is used to program the DMA device data size, timing mode, EOP input/output selection, and Stop register selection. When writing to the register, Bits [1:0] determine which channel's Extended Mode register will be written and are not stored. Only Bits [7:2] are stored in the extended mode register. Four timing modes are available: ISA-compatible, A, B, and Burst.

The default bit values for each DMA group are selected upon reset. A Master Clear or any other programming sequence will not set the default register settings. The default programmed values for DMA1 Channels 0–3 are 8-bit I/O Count by Bytes, Compatible timing, and EOP output. The default values for DMA2 Channels 4–7 are 16-bit I/O Count by Words with shifted address, Compatible timing, and EOP output. These default settings provide a rigorous ISA-compatible DMA implementation.

NOTE:
DMA1/DMA2 refer to the original PC-AT implementation which used two discrete 8237 DMA controllers. In this context DMA1 refers to DMA Channels 0–3 and DMA2 refers to DMA Channels 4–7. The PC-AT used Channel 4 (Channel 0 of DMA2) as a cascade channel for DMA1. Consequently, Channel 4 is not used in compatible DMA controllers although the compatible DMA registers are kept to maintain compatibility with the original PC-AT. Because Channel 4 is not used, the DMA controller does not support extended registers for Channel 4.

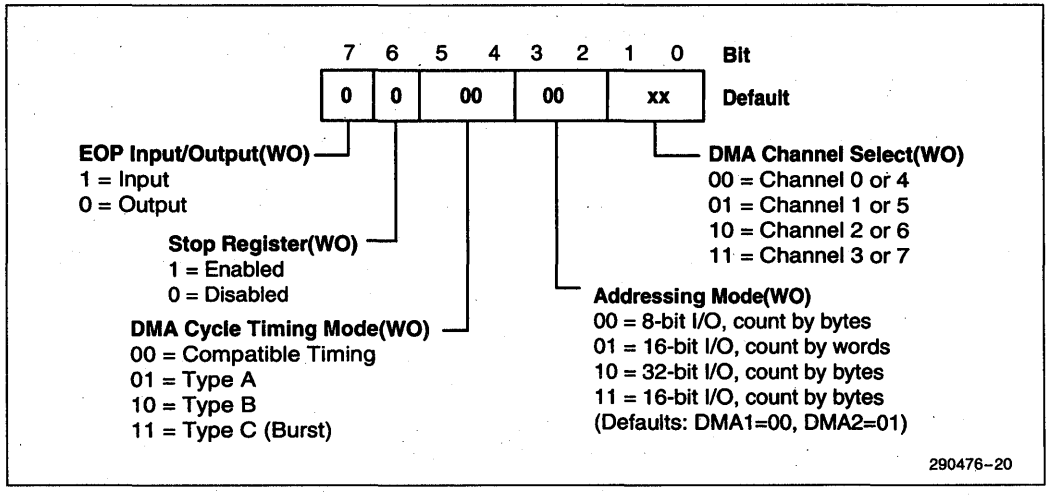


Figure 3-18. DMA Channel Extended Mode Register

Table 3-18. DMA Channel Extended Mode Register

Bit #	Description
7	**EOP INPUT/OUTPUT:** Bit 7 of this register selects whether or not the Stop registers associated with this channel are to be used. Normally the Stop Registers will not be used. This function was added to help support data communication or other devices that work from a ring buffer in memory. Upon reset, the Bit 7 is set to "0"-Stop register disabled. The detailed Stop register functional description discusses the use of the Stop registers.
6	**STOP REGISTER:** Bit 6 of the Extended Mode register selects whether the EOP signal is to be used as an output during DMA on this channel or an input. EOP will generally be used as an output, as was available on the PCAT. The input function was added to support Data Communication and other devices that would like to trigger an autoinitialize when a collision or some other event occurs. The direction of EOP is switched when DACK is changed (when a different channel wins the arbitration and is granted the bus). There may be some overlap of the ESC driving the EOP signal along with the DMA slave. However, during this overlap both devices will be driving the signal to a low level (negated). For example, assume Channel 2 is about to go inactive (DACK negated) and channel 1 is about to go active. If Channel 2 is programmed for "EOP OUT" and Channel 1 is programmed for "EOP IN", when Channel 2's DACK is negated and Channel 1's DACK is asserted, the ESC may be driving EOP to a low value on behalf of Channel 2 at the same time the device connected to Channel 1 is driving EOP in to the ESC, also at an inactive level. This overlap will only last until the ESC EOP output buffer is tri-stated, and will not effect the DMA operation. Upon reset, the value of Bit 6 is 0 (EOP output selected).
5:4	**DMA CYCLE TIMING MODE:** The ESC supports four DMA transfer timings: ISA-compatible, Type A, Type B, and Burst. Each timing and its corresponding code are described below. Upon reset, compatible timing is selected and the value of these bits is "00". The cycle timings noted below are for a BCLK (8.33 MHz) (maximum BCLK frequency). DMA cycles to ISA expansion bus memory will default to compatible timing if the channel is programmed in one of the performance timing modes (Type A, B, or Burst). 00 Compatible Timing DMA slaves on the ISA bus may run compatible DMA cycles. Bits [5:4] must be programmed to "00". Compatible timing is provided for DMA slave devices, which, due to some design limitation, cannot support one of the faster timings. Compatible timing runs at 9 BCLKs (1080 ns/single cycle) and 8 BCLKs (960 ns/cycle) during the repeated portion of a BLOCK or DEMAND mode transfers. 01 Type "A" Timing Type "A" timing is provided to allow shorter cycles to EISA memory. If ISA memory is decoded, the system automaticallly reverts to ISA DMA type compatible timing on a cycle-by-cycle basis. Type "A" timing runs at 7 BCLKs (840 ns/single cycle) and 6 BCLKs (720 ns/cycle) during the repeated portion of a BLOCK or DEMAND mode transfer. Type "A" timing varies from compatible timing primarily in shortening the memory operation to the minimum allowed by system memory. The I/O portion of the cycle (data setup on write, I/O read access time) is the same as with compatible cycles. The actual active command time is shorter, but it is expected that the DMA devices which provide the data access time or write data setup time should not require excess IOR# or IOW# command active time. Because of this, most ISA DMA devices should be able to use type "A" timing. 10 Type "B" Timing Type "B" timing is provided for 8-bit and 16-bit ISA or EISA DMA devices which can accept faster I/O timing. Type "B" only works with EISA memory. Type "B" timing runs at 6 BCLKs (720 ns/single cycle) and 4 BCLKs (480 ns/cycle) during the repeated portion of a BLOCK or DEMAND mode transfer. Type "B" timing requires faster DMA slave devices than compatible timing in that the cycles are shortened so that the data setup time on I/O write cycles is shortened and the I/O read access time is required to be faster. Some of the current ISA devices should be able to support type "B" timing, but these will probably be more recent designs using relatively fast technology.

Table 3-18. DMA Channel Extended Mode Register (Continued)

Bit #	Description
5:4	**DMA CYCLE TIMING MODE:** (Continued) 11 Type "C" Timing (Burst) Burst timing is provided for high performance EISA DMA devices. The DMA slave device needs to monitor the EXRDY and IORC# or IOWC# signals to determine when to change the data (on writes) or sample the data (on reads). This timing will allow up to 33 MBytes per second transfer rate with a 32-bit DMA device and 32-bit memory. Note that 8-bit or 16-bit DMA devices are supported (through the programmable Address size) and that they use the "byte lanes" natural to their size for the data transfer. As with all bursts, the system will revert to two BCLK cycles if the memory does not support burst. When a DMA burst cycle accesses non-burst memory and the DMA cycle crosses a page boundary into burstable memory, the ESC will continue performing non-burst cycles. This will not cause a problem since the data is still transferred correctly.
3:2	**ADDRESSING MODE:** The ESC supports 8-, 16-, and 32-bit DMA device data sizes. The four data size options are programmable with Bits [3:2]. Both the 8-bit I/O, "Count By Bytes" Mode and the 16-bit I/O, "Count By Words" (Address Shifted) Mode are ISA compatible. The 16-bit and 32-bit I/O, "Count By Bytes" Modes are EISA extensions. Byte assembly/disassembly is performed by the EISA Bus Controller. Each of the data transfer size modes is discussed below. 00 8-bit I/O, "Count By Bytes" Mode In 8-bit I/O, "count by bytes" mode, the address counter can be programmed to any address. The count register is programmed with the "number of bytes minus 1" to transfer. 01 16-bit I/O, "Count By Words" (Address Shifted) Mode In "count by words" mode (address shifted), the address counter can be programmed to any even address, but must be programmed with the address value shifted right by one bit. The Page registers are not shifted during DMA transfers. Thus, the least significant bit of the Low Page register is ignored when the address is driven out onto the bus. The Word Count register is programmed with the number of words minus 1 to be transferred. 10 32-Bit I/O, "Count By Bytes" Mode In 32-bit "count by bytes" mode, the address counter can be programmed to any byte address. For most DMA devices, however, it should only be programmed to a double-word aligned address. If the starting address is not double-word aligned then the DMA controller will do a partial dword transfer during the first and last transfers if necessary. The bus controller logic will do the byte/word assembly necessary to read or write any size memory device and both the DMA and bus controllers support burst for this mode. In this mode, the Address register is usually incremented or decremented by four and the byte count is usually decremented by four. The Count register should be programmed with the number of bytes to be transferred minus 1. 11 16-Bit I/O, "Count By Bytes" Mode In 16-bit "count by bytes" mode, the address counter can be programmed to any byte address. For most DMA devices, however, it should be programmed only to even addresses. If the address is programmed to an odd address, then the DMA controller will do a partial word transfer during the first and last transfer if necessary. The bus controller will do the byte/word assembly necessary to write any size memory device. In this mode, the Address register is incremented or decremented by two and the byte count is decremented by the number of bytes transferred during each bus cycle. The Word Count register is programmed with the "number of bytes minus 1" to be transferred. This mode is offered as an extension of the two ISA compatible modes discussed above. This mode should only be programmed for 16-bit ISA DMA slaves.
1:0	**DMA CHANNEL SELECT:** Bits [1:0] selects the particular channel that will have its DMA Channel Extend Mode Register programmed with bits [7:2]. **Bit 1** **Bit 0** **Channel** 0 0 Channel 0 (4) 0 1 Channel 1 (5) 1 0 Channel 2 (6) 1 1 Channel 3 (7)

3.2.4 DR—DMA REQUEST REGISTER

Register Name: DMA Request

Register Location: 09h—Channels 0–3
0D2h—Channels 4–7

Default Value: 000000xxb

Attribute: Write Only

Size: 4 bits

Each channel has a Request bit associated with it in one of the two 4-bit Request registers. The Request register is used by software to initiate a DMA request. The DMA responds to the software request as though DREQ[x] is asserted. These requests are non-maskable and subject to prioritization by the Priority Encoder network (refer to the Channel Priority Functional Description). Each register bit is set or reset separately under software control or is cleared upon generation of a TC. The entire register is cleared upon reset or a Master Clear. It is not cleared upon a RSTDRV output. To set or reset a bit, the software loads the proper form of the data word. Bits [1:0] determine which channel Request register will be written. In order to make a software request, the channel must be in Block Mode. The Request register status for DMA1 and DMA2 is output on Bits [7:4] of a Status register read to the appropriate port.

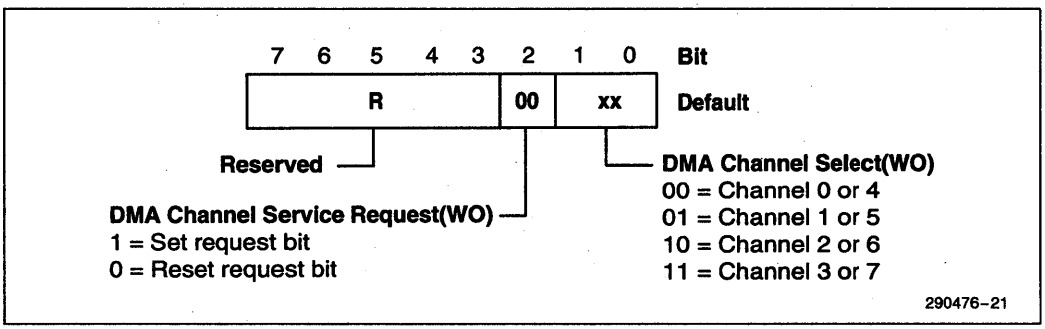


Figure 3-19. DMA Request Register

Table 3-19. DMA Request Register

Bit #	Description
7:3	**RESERVED:** (must be 0)
2	**DMA CHANNEL SERVICE REQUEST:** Writing a 0 to Bit 2 resets the individual software DMA channel request bit. Writing a 1 to Bit 2 will set the request bit. The request bit for each DMA channel is reset to 0 upon a reset or a Master Clear.
1:0	**DMA CHANNEL SELECT:** Bits [1:0] select the DMA channel mode register to program with bit 2. **Bit 1** **Bit 0** **Channel** 0 0 Channel 0 0 1 Channel 1 (5) 1 0 Channel 2 (6) 1 1 Channel 3 (7)

3.2.5 MASK REGISTER—WRITE SINGLE MASK BIT

Register Name: Mask Register—Write Single Mask Bit

Register Location: 0Ah—Channels 0-3
0D4h—Channels 4-7

Default Value: 000001xxb

Attribute: Write Only

Size: 1 bit/channel

Each DMA channel has a mask bit that can disable an incoming DMA channel service request DREQ[x] assertion. Two 4-bit registers store the current mask status for DMA1 and DMA2. Setting the mask bit disables the incoming DREQ[x] for that channel. Clearing the mask bit enables the incoming DREQ[x]. A channel's mask bit is automatically set when the Current Word Count register reaches terminal count (unless the channel is programmed for autoinitialization). Each mask bit may also be set or cleared under software control. The entire register is also set by a reset or a Master Clear. Setting the entire register disables all DMA requests until a clear Mask register instruction allows them to occur. This instruction format is similar to the format used with the Request register.

Individually masking DMA Channel 4 (DMA controller 2, Channel 0) will automatically mask DMA Channels [3:0], as this Channel group is logically cascaded onto Channel 4. Setting this mask bit disables the incoming DREQ's for Channels [3:0].

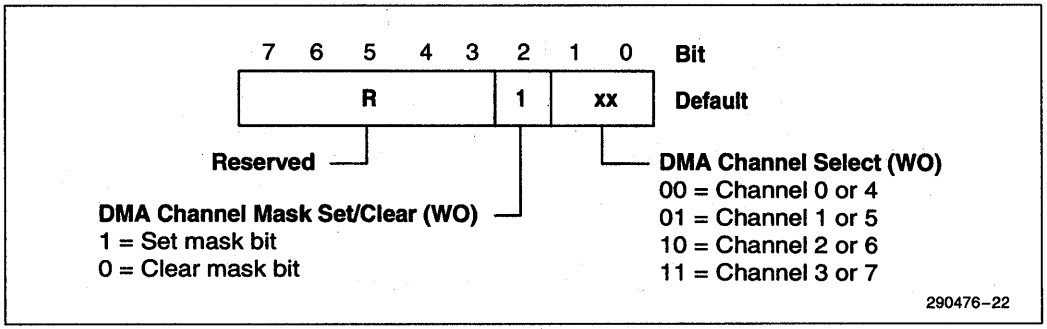


Figure 3-20. Mask Register Single Bit

Table 3-20. Mask Register Single Bit

Bit #	Description
7:3	**RESERVED:** (must be 0)
2	**DMA CHANNEL MASK SET/CLEAR:** Writing a 1 to Bit 2 sets the mask bit and disables the incoming DREQ for the selected channel. Writing a 0 to Bit 2 clears the mask bit and enables the incoming DREQ for the elected channel.
1:0	**DMA CHANNEL SELECT:** Bits [1:0] select the DMA Channel Mode Register to program with bit 2. **Bit 1** **Bit 0** **Channel** 0 0 Channel 0 (4) 0 1 Channel 1 (5) 1 0 Channel 2 (6) 1 1 Channel 3 (7)

3.2.6 MASK REGISTER—WRITE ALL MASK REGISTER BITS

Register Name: Mask Register-Write All Mask Register Bits

Register Location: 0Fh—Channels 0–3
0DEh—Channels 4–7

Default Value: 00001111b

Attribute: Read/Write

Size: 4 bits

This command allows enabling and disabling of incoming DREQ assertions by writing the mask bits for each controller, DMA1 or DMA2, simultaneously rather than by individual channel as is done with the "Write Single Mask Bit" command. Two 4-bit registers store the current mask status for DMA1 and DMA2. Setting the mask bit disables the incoming DREQ[x] for that channel. Clearing the mask bit enables the incoming DREQ[x]. Unlike the "Write Single Mask Bit" command, this command includes a status read to check the current mask status of the selected DMA channel group. When read, the mask register current status appears on Bits [3:0]. A channel's mask bit is automatically set when the Current Word Count register reaches terminal count (unless the channel is programmed for autoinitialization). The entire register is also set by a reset or a Master Clear. Setting the entire register disables all DMA requests until a clear Mask register instruction allows them to occur.

Two important points should be taken into consideration when programming the mask registers. First, individually masking DMA Channel 4 (DMA controller 2, Channel 0) will automatically mask DMA Channels [3:0], as this channel group is logically cascaded onto Channel 4. Second, masking off DMA controller 2 with a write to port 0DEh will also mask off DREQ assertions from DMA controller 1 for the same reason: when DMA Channel 4 is masked, so are DMA Channels 0–3.

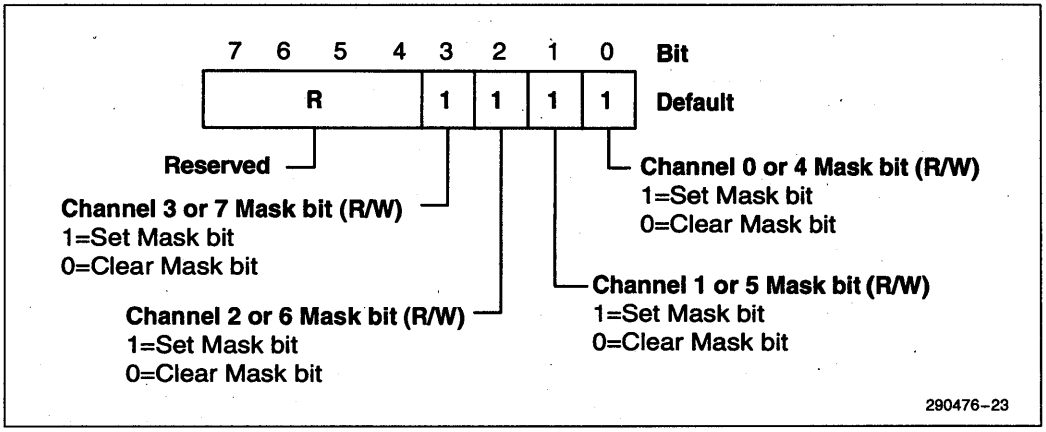


Figure 3-21. Mask Register All Bits

Table 3-21. Mask Register All Bits

Bit #	Description
7:4	**RESERVED:** Must be 0
3:0	**CHANNEL MASK BITS:** Setting the bit(s) to a 1 disables the corresponding DREQ(s). Setting the bit(s) to a 0 enables the corresponding DREQ(s). Bits [3:0] are set to 1 upon PCIRST# or Master Clear. When read, bits [3:0] indicate the DMA channel [3:0] ([7:4]) mask status. **Bit** **Channel** 0 0 (4) 1 1 (5) 2 2 (6) 3 3 (7) **NOTE:** Disabling channel 4 also disables channels 0–3 due to the cascade of DMA1 through channel 4 of DMA2.

3.2.7 DS—DMA STATUS REGISTER

Register Name: Status
Register Location: 08h—Channels 0–3
0D0h—Channels 4–7
Default Value: 00h
Attribute: Read Only
Size: 8 bits

Each DMA controller has a read-only Status register. A Status register read is used when determining which channels have reached terminal count and which channels have a pending DMA request. Bits [3:0] are set every time a TC is reached by that channel. These bits are cleared upon reset and on each Status Read. Bits [7:4] are set whenever their corresponding channel is requesting service.

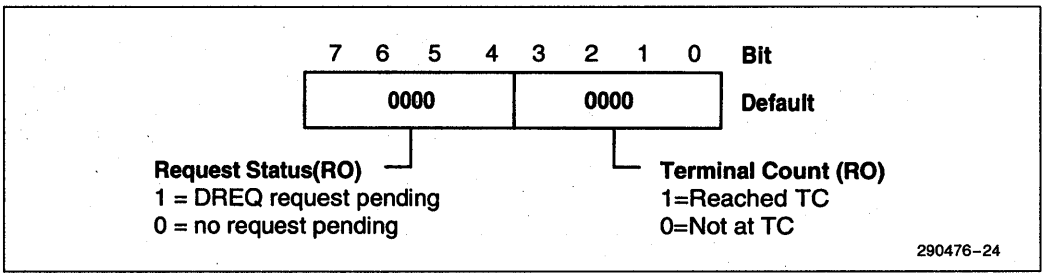


Figure 3-22. DMA Status Register

Table 3-22. DMA Status Register

Bit #	Description
7:4	**REQUEST STATUS:** When a valid DMA request is pending for a channel (on its DREQ signal line), the corresponding bit is set to 1. When a DMA request is not pending for a particular channel, the corresponding bit is set to 0. The source of the DREQ may be hardware, a timed-out block transfer, or a software request. Note that channel 4 does not have DREQ or DACK lines, so the response for a read of DMA2 status for channel 4 is irrelevant. **Bit Channel** 4 0 5 1 (5) 6 2 (6) 7 3 (7)
3:0	**TERMINAL COUNT STATUS:** When a channel reaches terminal count (TC), its status bit is set to 1. If TC has not been reached, the status bit is set to 0. Note that channel 4 is programmed for cascade, and is not used for a DMA transfer. Therefore, the TC bit response for a status read on DMA2 for channel 4 is irrelevant. **Bit Channel** 0 0 1 1 (5) 2 2 (6) 3 3 (7)

3.2.8 DMA BASE AND CURRENT ADDRESS REGISTER (8237 COMPATIBLE SEGMENT)

Register Name: DMA Base and Current Address Register (8237 Compatible Segment)

Register Location:
000h—DMA Channel 0
002h—DMA Channel 1
004h—DMA Channel 2
006h—DMA Channel 3
0C0h—DMA Channel 4
0C4h—DMA Channel 5
0C8h—DMA Channel 6
0CCh—DMA Channel 7

Default Value: 0000h

Attribute: Read/Write

Size: 16 bits per channel

Each channel has a 16-bit Current Address register. This register holds the value of the 16 least significant bits of the full 32-bit address used during DMA transfers. The address is automatically incremented or decremented after each transfer and the intermediate values of the address are stored in the Current Address register during the transfer. This register is written to or read from by the microprocessor or bus master in successive 8-bit bytes. The programmer must issue the "Clear Byte Pointer Flip-Flop" command to reset the internal byte pointer and correctly align the write prior to programming the Current address register. After clearing the Byte Pointer Flip-flop, the first write to the Current Address port programs the low byte, Bits [7:0], and the second write programs the high byte, Bits [15:8]. This procedure applies for read cycles also. It may also be re-initialized by an Autoinitialize back to its original value. Autoinitialize takes place only after a TC or EOP.

Each channel has a Base Address register located at the same port address as the corresponding Current Address register. These registers store the original value of their associated Current registers. During autoinitialize these values are used to restore the Current registers to their original values. The Base registers are written simultaneously with their corresponding Current register in successive 8-bit bytes by the microprocessor. The Base registers cannot be read by any external agents.

In Scatter-Gather Mode these registers store the lowest 16 bits of the current memory address. During a Scatter-Gather transfer the DMA will load a reserve buffer into the base memory address register.

In Chaining Mode, these registers store the lowest 16 bits of the current memory address. The CPU will program the base register set with a reserve buffer.

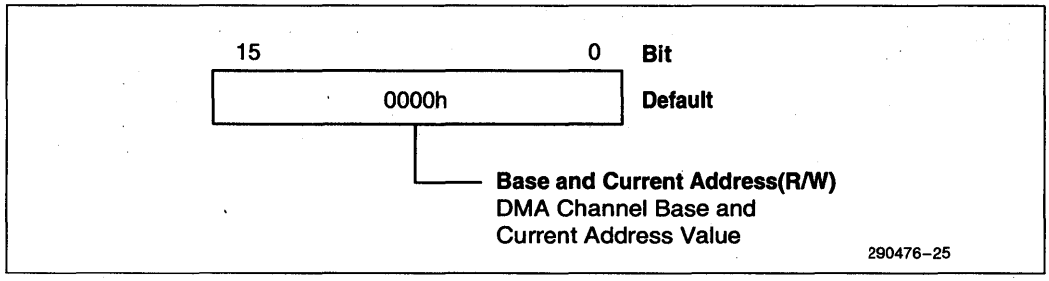


Figure 3-23. DMA Base and Current Address Register

Table 3-23. DMA Base and Current Address Register

Bit #	Description
15:0	**BASE AND CURRENT ADDRESS:** These bits represent the 16 least significant address bits used during DMA transfers. Together with the DMA Low Page register, they help form the ISA-compatible 24-bit DMA address. As an extension of the ISA compatible functionality, the DMA High Page register completes the 32-bit address needed when implementing ESC extensions such as DMA to the PCI bus slaves that can take advantage of full 32-bit addressability. Upon reset or Master Clear, the value of these bits is 0000h.

3.2.9 DMA BASE AND CURRENT BYTE/WORD COUNT REGISTER (8237 COMPATIBLE SEGMENT)

Register Name: DMA Base and Current Byte/Word Count Register (8237 Compatible Segment)

Register Location:
001h—DMA Channel 0
003h—DMA Channel 1
005h—DMA Channel 2
007h—DMA Channel 3
0C2h—DMA Channel 4
0C6h—DMA Channel 5
0CAh—DMA Channel 6
0CEh—DMA Channel 7

Default Value: 0000h

Attribute: Read/Write

Size: 16 bits per channel

Each channel has a 16-bit Current Byte/Word Count register. This register determines the lower 16 bits for the number of transfers to be performed. There is a total of 24 bits in the Byte/Word Count registers. The uppermost 8 bits are in the High Byte/Word Count register. The actual number of transfers will be one more than the number programmed in the Current Byte/Word Count register (i.e., programming a count of 100 will result in 101 transfers). The byte/word count is decremented after each transfer. The intermediate value of the byte/word count is stored in the register during the transfer. When the value in the register goes from zero to 0FFFFFFh, a TC will be generated.

Following the end of a DMA service it may also be re-initialized by an Autoinitialization back to its original value. Autoinitialize can occur only when a TC occurs. If it is not Autoinitialized, this register will have a count of FFFFh after TC.

When the Extended Mode register is programmed for "count by word" transfers to/from a 16-bit I/O, with shifted address, the Byte/Word count will indicate the number of 16-bit words to be transferred.

When the Extended Mode register is programmed for "count by byte" transfers, the Byte/Word Count will indicate the number of bytes to be transferred. The number of bytes does not need to be a multiple of the transfer size in this case.

Each channel has a Base Byte/Word Count register located at the same port address as the corresponding Current Byte/Word Count register. These registers store the original value of their associated Current registers. During Autoinitialize these values are used to restore the Current registers to their original values. The Base registers cannot be read by any external agents.

In Scatter-Gather mode these registers store the lowest 16 bits of the current Byte/Word Count. During a Scatter-Gather transfer the DMA will load a reserve buffer into the base Byte/Word Count register.

In Chaining Mode these registers store the lowest 16 bits of the current Byte/Word Count. The CPU will then program the base register set with a reserve buffer.

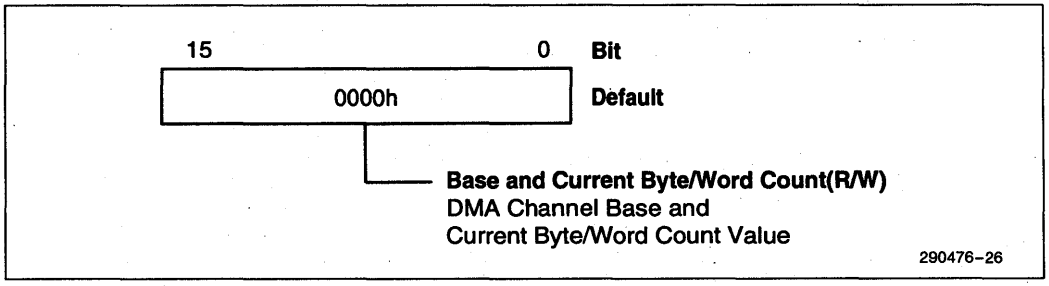


Figure 3-24. DMA Base and Current Byte/Word Count Register

Table 3-24. DMA Base and Current Byte/Word Count Register

Bit #	Description
15:0	**BASE AND CURRENT BYTE/WORD COUNT:** These bits represent the lower 16 byte/word count bits used when counting down a DMA transfer. Upon reset or Master Clear, the value of these bits is 0000h.

3.2.10 DMA BASE AND CURRENT HIGH BYTE/ WORD COUNT REGISTER DMA BASE HIGH BYTE/WORD COUNT REGISTER

Register Name: DMA Base and Current High Byte/Word Count (Read/Write)
DMA Base High Byte/Word Count (Write Only)

Register Location: 401h—DMA Channel 0
403h—DMA Channel 1
405h—DMA Channel 2
407h—DMA Channel 3
4C6h—DMA Channel 5
4CAh—DMA Channel 6
4CEh—DMA Channel 7

Default Value: 00h

Attribute: Read/Write

Size: 8 bits per channel

Each channel has a 8-bit Current High Byte/Word Count register. This register provides the uppermost 8 bits for the number of transfers to be performed. The byte/word count is decremented after each transfer. The intermediate value of the byte/word count is stored in the register during the transfer. When the value in the register goes from zero to FFFFh, a TC may be generated.

Following the end of a DMA service it may also be re-initialized by an Autoinitialization back to its original value. Autoinitialize can occur only when a TC occurs. If it is not Autoinitialized, this register will have a count of FFFFh after TC.

The High Byte/Word Count register must be the last Byte/Word Count register programmed. Writing to the 8237 Compatible Byte/Word Count registers will clear the High Byte/Word Count register to 00h.

When the Extended Mode register is programmed for "count by word" transfers to/from a 16-bit I/O, with shifted address, the Byte/Word count will indicate the number of 16-bit words to be transferred.

When the Extended Mode register is programmed for "count by byte" transfers, the Byte/Word Count will indicate the number of bytes to be transferred. The number of bytes does not need to be a multiple of the transfer size in this case.

Each channel has a Base High Byte/Word Count register located at the same port address as the corresponding Current High Byte/Word Count register. These registers store the original value of their associated Current registers. During Autoinitialize these values are used to restore the Current registers to their original values. Normally, the Base registers are written simultaneously with their corresponding Current register in successive 8-bit bytes by the microprocessor. However, in Chaining Mode only the Base register set is programmed and the Current register is not affected. The Base registers cannot be read by any external agents.

In Scatter-Gather mode these registers store the lowest 8 bits of the current High Byte/Word Count. During a Scatter-Gather transfer the DMA will load a reserve buffer into the base High Byte/Word Count register.

In Chaining Mode these registers store the lowest 8 bits of the current High Byte/Word Count. The CPU will then program the base register set with a reserve buffer.

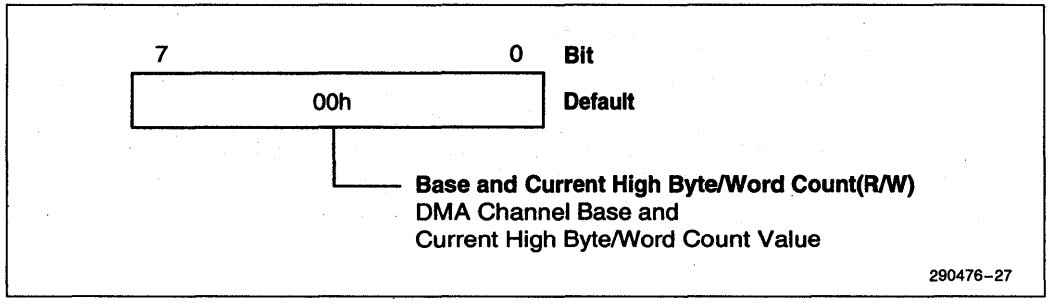


Figure 3-25. DMA Base and Current High Byte/Word Count Register

Table 3-25. DMA Base and Current High Byte/Word Count Register

Bit #	Description
7:0	**BASE AND CURRENT HIGH BYTE/WORD COUNT:** These bits represent the 8 high order byte/word count bits used when counting down a DMA transfer. Upon reset or Master Clear, the value of these bits is 00h.

3.2.11 DMA MEMORY LOW PAGE REGISTER DMA MEMORY BASE LOW PAGE REGISTER

Register Name: DMA Memory Low Page (Read/Write)
DMA Memory Base Low Page (Write Only)

Register Location: 087h—DMA Channel 0
083h—DMA Channel 1
081h—DMA Channel 2
082h—DMA Channel 3
08Bh—DMA Channel 5
089h—DMA Channel 6
08Ah—DMA Channel 7

Default Value: 00h

Size: 8 bits per channel

Each channel has an 8-bit Low Page register associated with it. The DMA memory Low Page register contains the eight second most-significant bits of the 32-bit address. It works in conjunction with the DMA controller's High Page register and Current Address register to define the complete (32-bit) address for the DMA channel. This 8-bit register is read or written directly by the processor or bus master. It may also be re-initialized by an Autoinitialize back to its original value. Autoinitialize takes place only after a TC or EOP.

Each channel has a Base Low Page Address register located at the same port address as the corresponding Current Low Page register. These registers store the original value of their associated Current Low Page registers. During Autoinitialize these values are used to restore the Current Low Page registers to their original values. The 8-bit Base Low Page registers are written simultaneously with their corresponding Current Low Page register by the microprocessor. The Base Low Page registers cannot be read by any external agents.

During Scatter-Gather these registers store the 8 bits from the third byte of the current memory address. During a Scatter-Gather transfer the DMA will load a reserve buffer into the base memory address register.

In Chaining Mode these registers store the 8 bits from the third byte of the current memory address. The CPU will program the base register set with a reserve buffer.

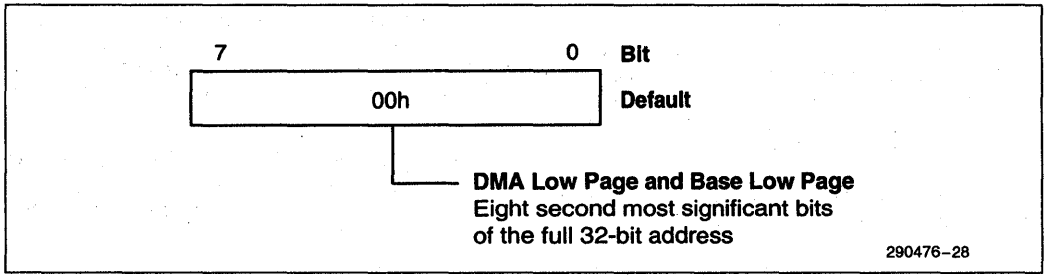


Figure 3-26. DMA Memory Low Page and Base Low Page Register

Table 3-26. DMA Memory Low Page and Base Low Page Register

Bit #	Description
7:0	**DMA LOW PAGE AND BASE LOW PAGE:** These bits represent the eight second most-significant address bits when forming the full 32-bit address for a DMA transfer. Upon reset or Master Clear, the value of these bits is 00h.

3.2.12 DMA PAGE REGISTER

Register Name: DMA Page (Read/Write)

Register Location: 080h, 84h, 85h, 86h, 88h, 8Ch, 8Dh, 8Eh

Default Value: xxh

Attribute: Read/Write

Size: 8 bits

These registers have no effect on the DMA operation. These registers provide extra storage space in the I/O space for DMA routines.

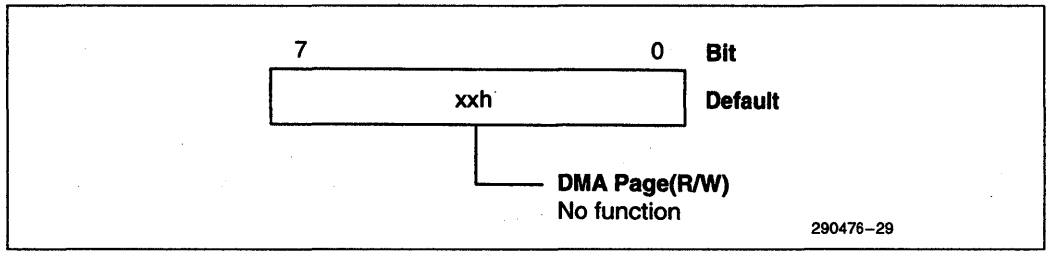


Figure 3-27. DMA Page Register

Table 3-27. DMA Page Register

Bit #	Description
7:0	**DMA PAGE:** These bits have no effect on the DMA operation. These bits only provide storage space in the I/O map.

3.2.13 DMA LOW PAGE REFRESH REGISTER

Register Name: DMA Low Page Refresh
Register Location: 08Fh
Default Value: xxh
Attribute: Read/Write
Size: 8 bits

The contents of this register are driven on the address byte 2 (LA[23:16]#) during Refresh cycles.

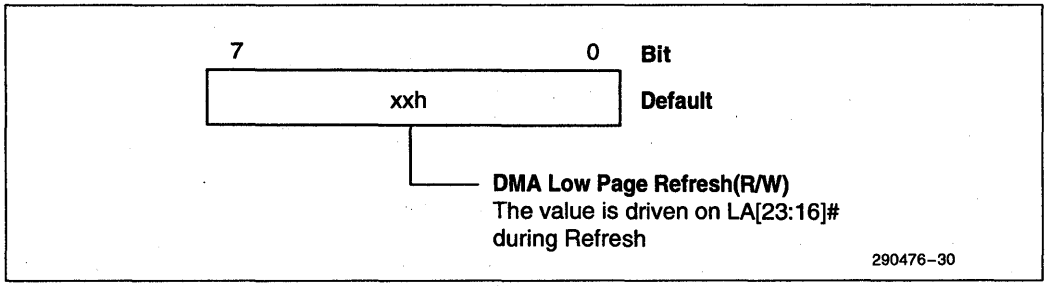


Figure 3-28. DMA Low Page Refresh Register

Table 3-28. DMA Low Page Refresh Register

Bit #	Description
7:0	**DMA LOW PAGE REFRESH:** The contents of the bits are driven on to the address bus during refresh.

3.2.14 DMA MEMORY HIGH PAGE REGISTER DMA MEMORY BASE HIGH PAGE REGISTER

Register Name: DMA Memory High Page (Read/Write)
DMA Memory Base High Page (Write Only)

Register Location: 0487h—DMA Channel 0
0483h—DMA Channel 1
0481h—DMA Channel 2
0482h—DMA Channel 3
048Bh—DMA Channel 5
0489h—DMA Channel 6
048Ah—DMA Channel 7

Default Value: 00h

Size: 8 bits per channel

Each channel has an 8-bit High Page register. The DMA memory High Page register contains the eight most-significant bits of the 32-bit address. It works in conjunction with the DMA controller's Low Page register and Current Address register to define the complete (32-bit) address for the DMA channels and corresponds to the "Current Address" register for each channel. This 8-bit register is read or written directly by the processor or bus master. It may also be re-initialized by an Autoinitialize back to its original value. Autoinitialize takes place only after a TC or EOP.

This register is reset to 00h during the programming of both the low page register and the Current Address register. Thus, if this register is not programmed after the other address and Low Page registers are programmed, then its value will be zero. In this case, the DMA channel will operate the same as an 82C37 (from an addressing standpoint). This is the address compatibility mode.

If the high 8 bits of the address are programmed after the other addresses, then the channel will modify its operation to increment (or decrement) the entire 32-bit address. This is unlike the 82C37 "Page" register in the original PCs which could only increment to a 64k boundary (for 8-bit channels) or 128k (for 16-bit channels). This is extended address mode. In this mode, the ISA bus controller will generate the signals MEMR# and MEMW# only for addresses below 16 Mbytes.

Each channel has a Base High Page Address register located at the same port address as the corresponding Current High Page Address register. These registers store the original value of their associated Current registers. During Autoinitialize these values are used to restore the Current registers to their original values. The 8-bit Base High Page registers are written simultaneously with their corresponding Current register by the microprocessor. The Base registers cannot be read by any external agents.

During Scatter-Gather these registers store the 8 bits from the highest byte of the current memory address. During a Scatter-Gather transfer the DMA will load a reserve buffer into the base memory address register.

In Chaining Mode these registers store the 8 bits from the highest byte of the current memory address. The CPU will program the base register set with a reserve buffer.

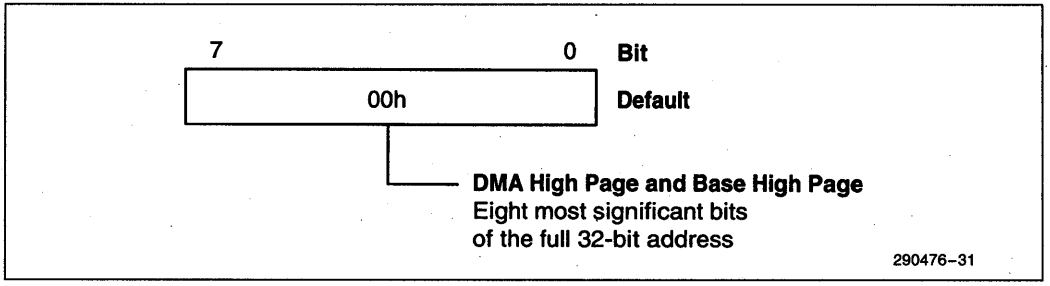


Figure 3-29. DMA Memory High Page and Base High Page Register

Table 3-29. DMA Memory High Page and Base High Page Register

Bit #	Description
7:0	**DMA HIGH PAGE AND BASE HIGH PAGE:** These bits represent the eight most-significant address bits when forming the full 32-bit address for a DMA transfer. Upon reset or Master Clear, the value of these bits is 00h.

3.2.15 DMA HIGH PAGE REGISTER REFRESH

Register Name: DMA High Page Register Refresh (Read/Write)

Register Location: 048Fh

Default Value: xxh

Attribute: Read/Write

Size: 8 bits per channel

The contents of this register are driven on the address byte 3 (LA[31:24]#) during Refresh cycles.

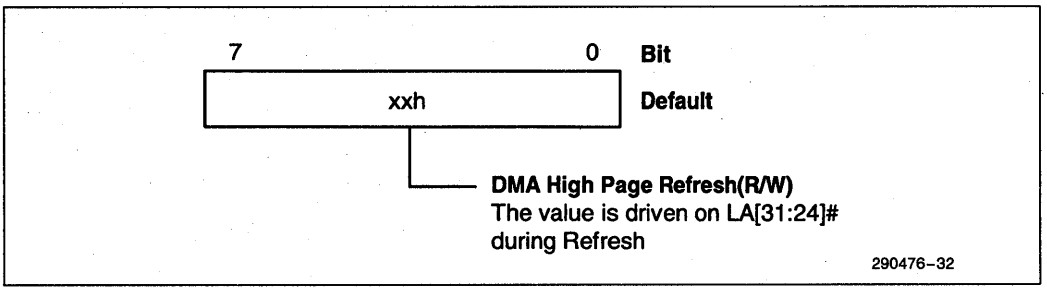


Figure 3-30. DMA High Page Refresh Register

Table 3-30. DMA High Page Refresh Register

Bit #	Description
7:0	**DMA HIGH PAGE REFRESH:** The contents of the bits are driven on to the address bus during refresh

3.2.16 STOP REGISTERS

Register Name: Stop Registers

Register Location:

04E0h—CH0 Stop Reg Bits [7:2]
04E1h—CH0 Stop Reg Bits [15:8]
04E2h—CH0 Stop Reg Bits [23:16]
04E4h—CH1 Stop Reg Bits [7:2]
04E5h—CH1 Stop Reg Bits [15:8]
04E6h—CH1 Stop Reg Bits [23:16]
04E8h—CH2 Stop Reg Bits [7:2]
04E9h—CH2 Stop Reg Bits [15:8]
04EAh—CH2 Stop Reg Bits [23:16]
04ECh—CH3 Stop Reg Bits [7:2]
04EDh—CH3 Stop Reg Bits [15:8]
04EEh—CH3 Stop Reg Bits [23:16]
04F4h—CH5 Stop Reg Bits [7:2]
04F5h—CH5 Stop Reg Bits [15:8]
04F6h—CH5 Stop Reg Bits [23:16]
04F8h—CH6 Stop Reg Bits [7:2]
04F9h—CH6 Stop Reg Bits [15:8]
04FAh—CH6 Stop Reg Bits [23:16]
04FCh—CH7 Stop Reg Bits [7:2]
04FDh—CH7 Stop Reg Bits [15:8]
04FEh—CH7 Stop Reg Bits [23:16]

Default Value: See Below
Attribute: Read/Write
Size: See Below

The Stop registers are used to support a common data communication structure, the ring buffer. The ring buffer data structure and Stop register operation are described in Section 6.7.4. The Stop registers, in conjunction with a channel's Base and Current address and byte count registers, are used to define a fixed portion of memory for use by the ring buffer data structure. Following a reset, these registers are all reset to "0".

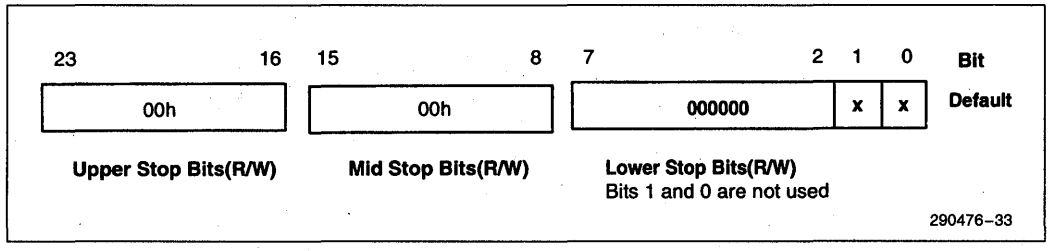


Figure 3-31. Stop Registers

Table 3-31. Stop Registers

Bit #	Description
23:2	**UPPER, MID, LOWER STOP BITS:** These 22 bits provide the Stop Address. If the Stop function is enabled then the channel will Stop whenever its Memory Address matches the Stop Address.

3.2.17 CHAINING MODE REGISTER

Register Name: Chaining Mode Register
Register Location: 040Ah—Channels 0–3
04D4h—Channels 4–7
Default Value: 000000xxb
Attribute: Write Only
Size: 8 bits

Each channel has a Chaining Mode register. The Chaining Mode register enables or disables DMA buffer chaining and indicates when the DMA Base registers are being programmed. When writing to the register, Bits [1:0] determine which channel's Chaining Mode register to program. The chaining status and interrupt status for all channels can be determined by reading the Chaining Mode Status, Channel Interrupt Status, and Chain Buffer Expiration Control registers. The Chaining Mode register is reset to zero upon reset, access (read or write) of a channel's Mode register or Extended Mode register, or a Master Clear. The values upon reset are disable chaining mode and generate IRQ13.

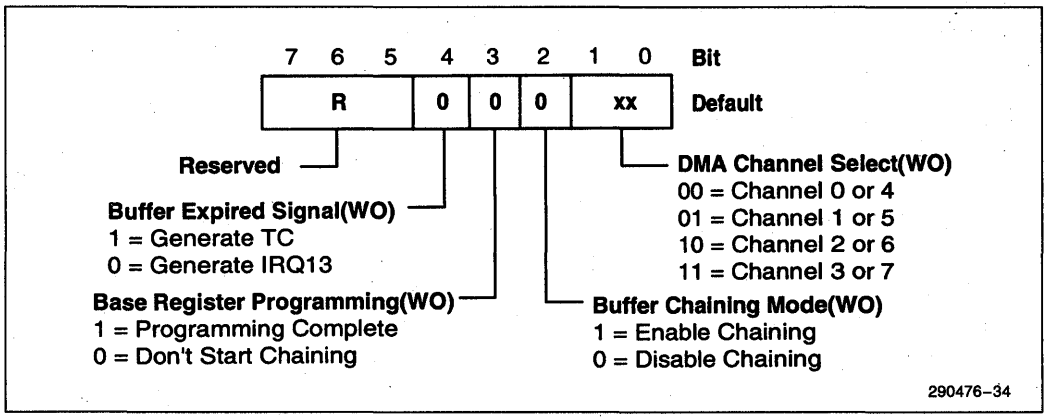


Figure 3-32. Chaining Mode Register

Table 3-32. Chaining Mode Register

Bit #	Description
7:5	**RESERVED:** (must be 0)
4	**BUFFER EXPIRED SIGNAL:** After one of the two buffers in the DMA expires then the DMA will inform the CPU that the next buffer should be loaded into the base register set. This bit determines whether IRQ13 or EOP should be used to inform the CPU that the buffer is complete.
3	**BASE REGISTER PROGRAMMING:** After the reserve buffer's address and word count are written to the base register set, this bit should be set to 1 to inform the DMA that the second buffer is ready for transfer.
2	**BUFFER CHAINING MODE:** Bit 2 enables the chaining mode logic. If the bit is set to 1 after the initial DMA address and word count are programmed, then the Base address and word count are available for programming the next buffer in the chain.
1:0	**DMA CHANNEL SELECT:** Bits [1:0] select the DMA channel mode register to program with Bits [4:2].

3.2.18 CHAINING MODE STATUS REGISTER

Register Name: Chaining Mode Status Register
Register Location: 04D4h
Default Value: 00h
Attribute: Read Only
Size: 8 bits

This register is read only and is used to determine if chaining mode for a particular channel is enabled or disabled. A 1 read in this register indicates that the channel's chaining mode is enabled. A 0 indicates that the chaining mode is disabled. All Chaining mode bits are disabled after a reset with reset. After the DMA is used in Chaining mode the CPU will need to clear the Chaining mode enable bit if non-Chaining mode is desired.

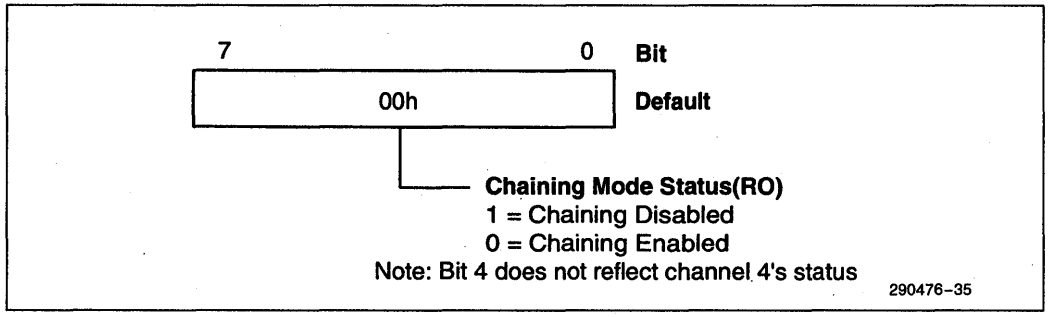


Figure 3-33. Chaining Mode Status Register

Table 3-33. Chaining Mode Status Register

Bit #	Description
7:5, 3:0	**CHAINING MODE STATUS:** If this bit is set to 1 then this channel has chaining enabled by writing 1 to Bit 2 of the Chaining Mode Register. This bit can be reset to 0 by either writing a 0 to Bit 2 of the Chaining Mode Register or reset being asserted or by a Master Clear Command.

3.2.19 CHANNEL INTERRUPT STATUS REGISTER

Register Name: Channel Interrupt Status
Register Location: 040Ah
Default Value: 00h
Attribute: Read Only
Size: 8 bits

Channel Interrupt Status is a read only register and is used to indicate the source (channel) of a DMA chaining interrupt on IRQ13. The DMA controller asserts IRQ13 after reaching terminal count, with chaining mode enabled. It does not assert IRQ13 during the initial programming sequence that loads the Base registers. After a reset, a read of this register will produce 00h.

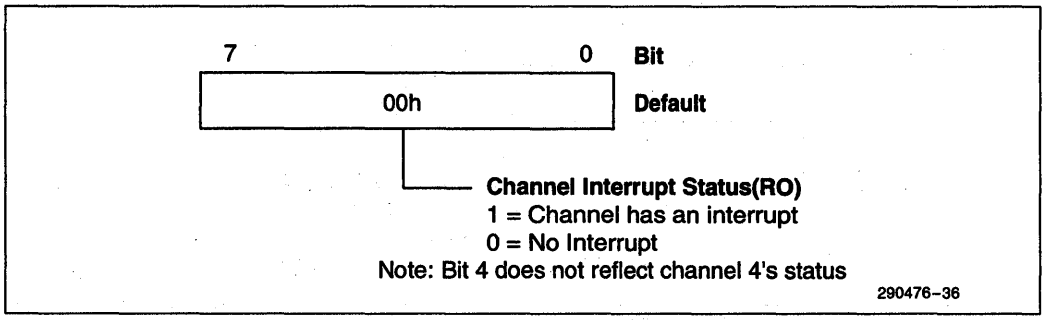


Figure 3-34. Channel Interrupt Status Register

Table 3-34. Channel Interrupt Status Register

Bit #	Description
7:5, 3:0	**CHAINING INTERRUPT STATUS:** When a channel interrupt status read returns a "0", Bit [7:5, 3:0] indicates that channel did not assert IRQ13. When a channel interrupt status read returns a "1", then that channel asserted IRQ13 after reaching a Terminal Count.

3.2.20 CHAIN BUFFER EXPIRATION CONTROL REGISTER

Register Name: Chain Buffer Expiration Control
Register Location: 040Ch
Default Value: 00h
Attribute: Read Only
Size: 8 bits

This register is read only and reflects the outcome of the expiration of a chain buffer. A Chain Buffer Expiration Control register bit with 0 indicates the DMA controller asserts IRQ13 when the DMA controller reaches terminal count. A 1 indicates the DMA controller asserts TC when the DMA controller reaches terminal count. This bit is programmed in Bit 4 of the Chaining Mode register.

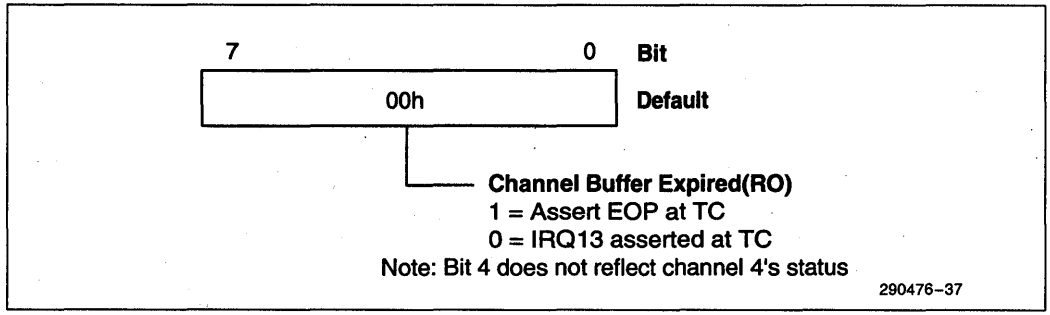


Figure 3-35. Chain Buffer Expiration Control Register

Table 3-35. Chain Buffer Expiration Control Register

Bit #	Description
7:5, 3:0	**CHAINING BUFFER EXPIRED:** When a chain buffer expiration control read returns "0", Bit [7:5, 3:0] indicates that Channel [7:5, 3:0] will assert IRQ13 when the DMA channel reaches terminal count. When a chain buffer expiration control read returns "1", Bit [7:5, 3:0] indicates that Channel [7:5, 3:0] will assert TC when the DMA controller reaches terminal count. This bit will reset to 0 following a reset.

3.2.21 SCATTER-GATHER COMMAND REGISTER

Register Name: Scatter-Gather Command

Register Location: 0410h—Channels 0
0411h—Channels 1
0412h—Channels 2
0413h—Channels 3
0415h—Channels 5
0416h—Channels 6
0417h—Channels 7

Default Value: 00xxxx00b

Attribute: Write Only, Relocateable

Size: 8 bits

The Scatter-Gather command register controls operation of the Descriptor Table aspect of S-G transfers. The S-G command register is write only. The current S-G transfer status can be read in the S-G channel's corresponding S-G Status register. The S-G command register can initiate a S-G transfer, and stop a transfer.

Scatter-Gather commands are issued with command codes. Bits [1:0] are used to implement the code mechanism. The S-G codes are described in the table below. Bit 7 is used to control the IRQ13/EOP assertion that follows a terminal count. Bit 6 controls the effect of Bit 7. Common Scatter-Gather command writes are listed in Table 3-36.

Table 3-36. Scatter-Gather Command Bits

Command	Bits	
	7654	3210
No S-G Operation (S-G NOOP)	0000	0000b
Start S-G	xx00	0001b
Stop S-G	xx00	0010b
Issue IRQ13 on Terminal Count	0100	00xxb
Issue EOP on Terminal Count	1100	00xxb

Note that the "x" don't care states in the above table do not preclude programming those bits during the command write. For instance, for any S-G command code on Bits [1:0], an optional selection of IRQ13 or EOP can take place if Bit 7 is set to 1 and the appropriate choice is made for Bit 6. All 0's in the command byte indicate an S-G NOOP: no S-G command is issued, and EOP/IRQ13 modification is disabled. Note that an EOP/IRQ13 modification can be made while disabling the S-G command bits (Bits[1:0] = 00b); conversely, an S-G command may be issued while EOP/IRQ13 modification is disabled (Bit 6 = 0b). After a reset, or Master Clear, IRQ13 is disabled and EOP is enabled.

The Start command assumes the Base and Current registers are both empty and will request a prefetch automatically. It also sets the status register to S-G Active, Base Empty, Current Empty, not Terminated, and Next Null Indicator to 0. The EOP/IRQ13 bit will still reflect the last value programmed.

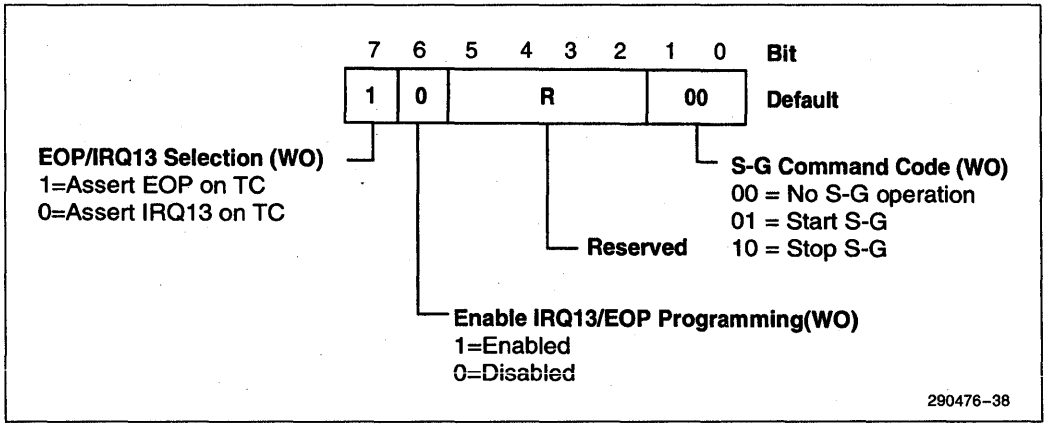


Figure 3-36. Scatter-Gather Command Register

Table 3-37. Scatter-Gather Command Register

Bit #	Description
7	**EOP/IRQ13 SELECTION:** Bit 7 is used to select whether EOP or IRQ will be asserted at termination caused by the last buffer expiring. The last buffer can be either the last buffer in the list or the last buffer loaded in the DMA while it is suspended. If this bit is set to 1 then EOP will be asserted whenever the last buffer is completed. If this bit is set to 0 then IRQ13 will be asserted whenever the last buffer is completed. EOP can be used to alert an expansion bus I/O device that a scatter-gather termination condition was reached; the I/O device in turn can assert its own interrupt request line, and invoke a dedicated interrupt handling routine. IRQ13 should be used whenever the CPU needs to be notified directly. Following reset, or Master Clear, the value stored for this bit is 0, and IRQ13 is selected. Bit 6 must be set to a 1 to enable this bit during an S-G Command register write. When Bit 6 is a 0 during the write, Bit 7 will not have any effect on the current EOP/IRQ13 selection.
6	**ENABLE IRQ13/EOP PROGRAMMING:** Enabling IRQ13/EOP programming allows initialization or modification of the S-G termination handling bits. If Bit 5 is reset to "0", Bit 7 will not have any affect on the state of IRQ13 or EOP assertion. When Bit 5 is set to a "1", Bit 7 determines the termination handling following a terminal count.
5:2	**RESERVED.**
1:0	**S-G COMMAND CODE:** 00 = No S-G command operation is performed. Bits[7:5] may still be used to program EOP/IRQ13 selection. 01 = The Start command initiates the scatter-gather process. Immediately after the Start command is issued a request is issued to fetch the initial buffer to fill the Base Register set in preparation for performing a transfer. The Buffer Prefetch request has the same priority with respect to other channels as the DREQ it is associated with. Within the channel, DREQ is higher in priority than a prefetch request. 10 = The Stop command halts a Scatter-Gather transfer immediately. When a Stop command is given, the Terminate bit in the S-G Status register and the DMA channel mask bit are both set. 11 = Reserved

The S-G Status register contains information on the S-G transfer status. This register maintains dynamic status information on S-G Transfer Activity, the Current and Base Buffer state, S-G Transfer Termination, and the End of the List indicator.

3.2.22 SCATTER-GATHER STATUS REGISTER

Register Name:	Scatter-Gather Status
Register Location:	Channels 0 0419h—Channels 1 041Ah—Channels 2 041Bh—Channels 3 041Dh—Channels 5 041Eh—Channels 6 041Fh—Channels 7
Default Value:	08h
Attribute:	Read Only, Relocatable
Size:	8 bits

The Scatter-Gather Status register provides Scatter-Gather process status information to the CPU or Master. An Active bit is set to 1 after the S-G Start command is issued. The Active bit will be 0 before the initial Start command, following a terminal count, and after an S-G Stop command is issued. The Current Buffer and Base Buffer State bits indicate whether the corresponding register has a buffer loaded. It is possible for the Base Buffer State to be set while the Current Buffer State is cleared. When the Current Buffer transfer is complete, the Base Buffer will not be moved into the Current Buffer until the start of the next data transfer. Thus, the Current Buffer State is empty (cleared), while the Base Buffer State is full (set). The Terminate bit is set active after a Stop command, after TC for the last buffer in the list and both Base and Current buffers have expired. The EOP and IRQ13 bits indicate which end of process indicator will be used to alert the system of an S-G process termination. The EOL status bit is set if DMA controller has loaded the last buffer of the Link List.

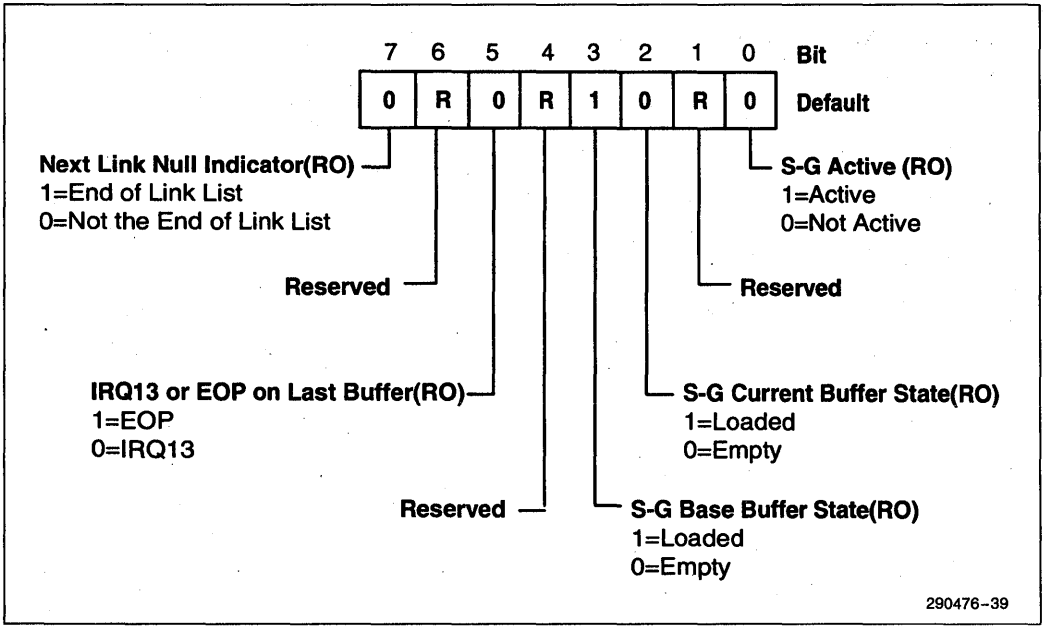


Figure 3-37. Scatter-Gather Status Register

Table 3-38. Scatter-Gather Status Register

Bit #	Description
7	**NEXT LINK NULL INDICATOR:** If the Next SGD fetched from memory during a fetch operation has the EOL value (1), the current value of the Next Link register is not overwritten. Instead, Bit 7 of the channel's S-G Status register, the Next Link Null indicator, is set to a "1". If the fetch returns a EOL value not equal to (1), this bit is reset to "0". This status bit is written after every fetch operation. Following reset, or Master Clear, this bit is reset to "0". This bit is also cleared by an S-G Start Command Write.
6	**RESERVED.**
5	**IRQ13 OR EOP ON LAST BUFFER:** When the IRQ13/EOP status bit contains a "1", EOP was either defaulted to at reset or selected through the S-G Command register as the S-G process termination indicator. EOP will be issued to alert the system when a terminal count occurs or following the Stop Command. When this bit is returned as a "0", an IRQ13 will be issued to alert the CPU of this same status.
4	**RESERVED.**
3	**S-G BASE BUFFER STATE:** When the Base Buffer status bit contains a "0", the Base Buffer is empty. When the Base Buffer Status bit is set to "1", the Base buffer has a buffer link loaded. Note that the Base Buffer State may be set while the Current buffer state is cleared. This condition occurs when the Current Buffer expires following a transfer; the Base Buffer will not be moved into the Current Register until the start of the next DMA transfer.
2	**S-G CURRENT BUFFER STATE:** When the Current Buffer status bit contains a "0", the Current Buffer is empty. When the Current Buffer status bit is set to "1", the Current Buffer has a buffer link loaded and is considered full. Following reset, Bit 2 is reset to "0".
1	**RESERVED.**
0	**S-G ACTIVE:** The Scatter-Gather Active bit indicates the current S-G transfer status. Bit 0 will be a 1 after an S-G Start Command is issued. Bit 0 will be a 0 before the Start command is issued. Bit 0 will be a 0 after terminal count on the last buffer on the channel is reached. Bit 0 will also be a 0 after an S-G Stop command has been issued. Following reset, or Master Clear, this bit is reset to "0".

3.2.23 SCATTER-GATHER DESCRIPTOR TABLE POINTER REGISTER

Register Name: Scatter-Gather Descriptor Table Pointer

Register Location: 0420h–0423h—Channels 0
0424h–0424h—Channels 1
0428h–042Bh—Channels 2
042Ch–042Ch—Channels 3
0434h–0437h—Channels 5
0438h–043Bh—Channels 6
043Ch–043Fh—Channels 7

Default Value: See below

Attribute: Read/Write, Relocatable

Size: 32 bits

The SGD Table Pointer register contains the 32-bit pointer to the first SGD entry in the SGD table in memory. Before the start of a S-G transfer, this register should have been programmed to point to the first SGD in the SGD table. Following a "Start" command, it initiates reading the first SGD entry by pointing to the first SGD entry to be fetched from the memory. Subsequently, at the end of the each buffer block transfer, the contents of the SGD table pointer registers are incremented by 8 till end of the SGD table is reached.

When programmed by the CPU, the SGD Table Pointer Registers can be programmed with a single 32-bit PCI write.

NOTE:
The PCEB and EISA Bus Controller will split the 32-bit write into four 8-bit writes.

Following a prefetch to the address pointed to by the channel's SGD table pointer register, the new Memory Address is loaded into the Base Address register, the new Byte Count is loaded into the Base Byte Count register, and the newly fetched Next SGD replaces the current Next SGD value.

The end of the SGD table is indicated by a End of Table field having a MSB equal to 1. When this value is read during a SGD fetch, the current SGD value is not replaced. Instead, Bit 7 of the channel's status register is set to a 1 when the EOL is read from memory.

Table 3-39. Scatter-Gather Table Pointer Register

Bit #	Description
31:0	**SGD TABLE POINTER:** The SGD table pointer register contains 32-bit pointer to the main memory location where the software maintains the Scatter-Gather Descriptors for the linked-list buffers. These bits are translated into A[31:0] signals for accessing memory on the PCI.

3.2.24 CLEAR BYTE POINTER FLIP-FLOP REGISTER

Register Name: Clear Byte Pointer Flip-Flop
Register Location: 00Ch—Channels 0–3
0D8h—Channels 4–7
Default Value: xxh
Attribute: Write Only
Size: n/a

This command is executed prior to writing or reading new address or word count information to the DMA. This initializes the flip-flop to a known state so that subsequent accesses to register contents by the microprocessor will address upper and lower bytes in the correct sequence.

The Clear Byte Pointer command clears the internal latch used to address the upper or lower byte of the 16-bit address and Word Count registers. The latch is also cleared at power on by reset and by the Master Clear command. The Host CPU may read or write a 16-bit DMA controller register by performing two consecutive accesses to the I/O port. The Clear Byte Pointer command precedes the first access. The first I/O write to a register port loads the least significant byte, and the second access automatically accesses the most significant byte.

When the Host CPU is reading or writing DMA registers, two Byte Pointer Flip-Flops are used; one for Channels 0–3 and one for Channels 4–7. Both of these act independently. There are separate software commands for clearing each of them (0Ch for Channels 0–3, 0D8h for Channels 4–7).

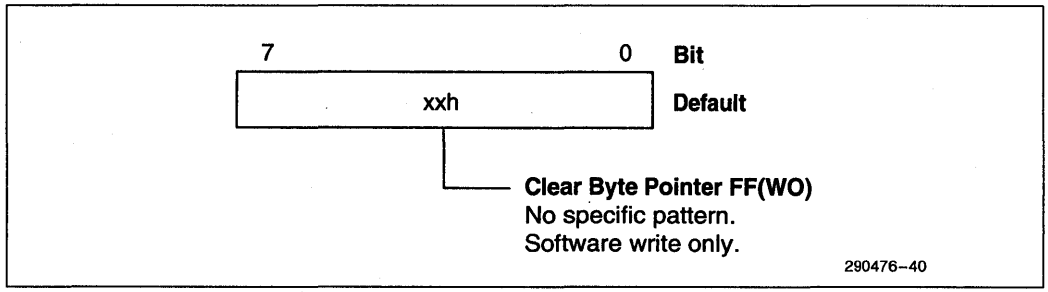


Figure 3-38. Clear Byte Pointer Flip-Flop Register

Table 3-40. Clear Byte Pointer Flip-Flop Register

Bit #	Description
7:0	**CLEAR BYTE POINTER FF:** No specific pattern. Command enabled with a write to the I/O port address.

3.2.25 DMC—DMA MASTER CLEAR REGISTER

Register Name: Master Clear
Register Location: 00Dh—Channel 0–3
0DAh—Channel 4–7
Default Value: xxh
Attribute: Write Only
Size: n/a

This software instruction has the same effect as the hardware Reset. The Command, Status, Request, and Internal First/Last Flip-Flop registers are cleared and the Mask register is set. The DMA controller will enter the idle cycle.

There are two independent Master Clear Commands, 0Dh which acts on Channels 0–3, and 0DAh which acts on Channels 4–7.

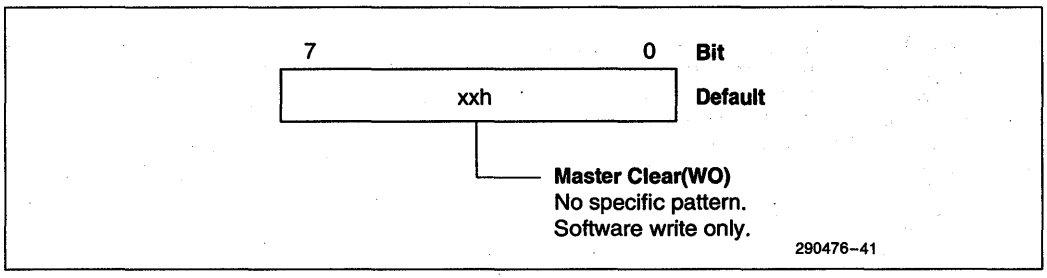


Figure 3-39. DMA Master Clear Register

Table 3-41. DMA Master Clear Register

Bit #	Description
7:0	**MASTER CLEAR:** No specific pattern. Command enabled with a write to the I/O port address.

3.2.26 DCM—DMA CLEAR MASK REGISTER

Register Name: Clear Mask

Register Location: Port Address: 00Eh—Channel 0–3
0DCh—Channel 4–7

Default Value: xxh

Attribute: Write Only

Size: n/a

This command clears the mask bits of all four channels, enabling them to accept DMA requests. I/O port 0Eh is used for Channels 0–3 and I/O port 0DCh is used for Channels 4–7.

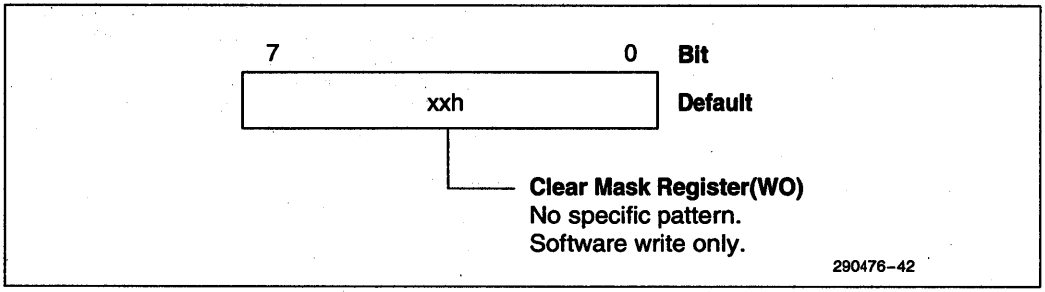


Figure 3-40. DMA Clear Mask Register

Table 3-42. DMA Clear Mask Register

Bit #	Description
7:0	**CLEAR MASK:** No specific pattern. Command enabled with a write to the I/O port address.

3.3 Timer Unit Registers

The ESC contains five counters that are equivalent to those found in the 82C54 Programmable Interval Timer. The Timer registers control these counters and can be accessed from the EISA Bus via I/O space. This section describes the counter/timer registers on the ESC. The counter/timer operations are further described in Chapter 8.0, Interval Timers.

3.3.1 TCW—TIMER CONTROL WORD REGISTER

Register Name:	Timer 1 and Timer 2 Control Word
Register Location:	043h—Timer 1 04Bh—Timer 2
Default Value:	xxh
Attribute:	Write Only
Size:	8 bits

The Timer Control Word specifies the counter selection, the operating mode, the counter byte programming order and size of the COUNT value, and whether it counts down in a 16-bit or binary-coded decimal (BCD) format. After writing the control word, a new count may be written at any time. The new value will take effect according to the programmed mode.

There are six programmable counting modes. Typically, the ESC Timer Unit Counters 0 and 2 are programmed for Mode 3, the Square Wave Mode, while Counter 1 is programmed in Mode 2, the Rate Generator Mode.

Two special commands are selected through the Control Word Register. The Counter Latch Command is selected when Bits[5:4] are both "0". The Read-Back Command is selected when Bits[7:6] are both "1". When either of these two commands are selected with the Control Word Register, the meaning of the other bits in the register changes. Both of these special commands, and the respective changes they make to the bit definitions in this register, are covered in detail under separate register descriptions later in this Section.

Bits 4 and 5 are also used to select the count register programming mode. The programming process is simple:

1. Write a control word.
2. Write an initial count for each counter.
3. Load the LSB, MSB, or LSB then MSB.

The read/write selection chosen with the control word dictates the programming sequence that must follow when initializing the specified counter.

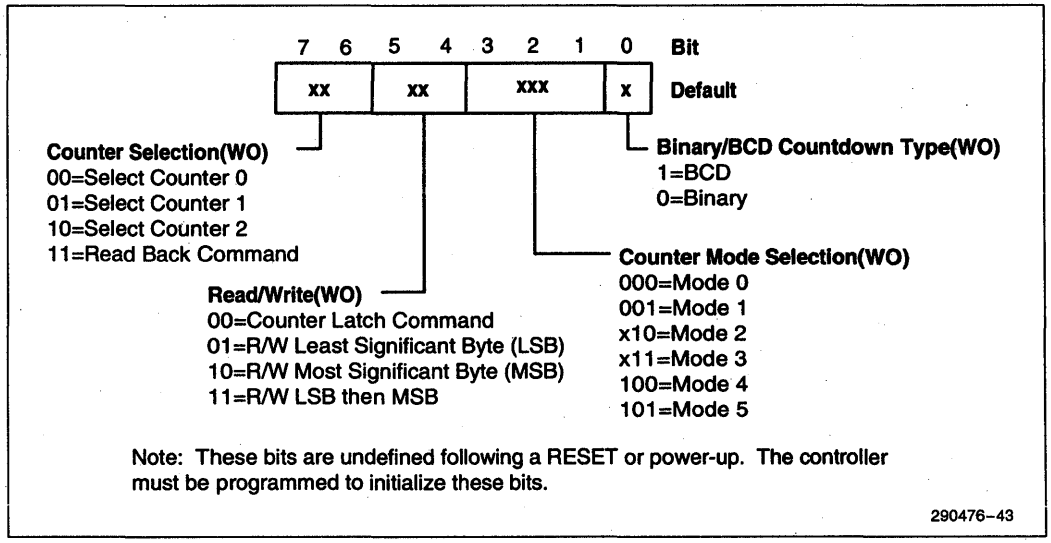


Figure 3-41. Timer Control Word Register

If a counter is programmed to read/write two-byte counts, the following precaution applies: A program must not transfer control between writing the first and second byte to another routine which also writes into that same counter. Otherwise, the counter will be loaded with an incorrect count. The count must always be completely loaded with both bytes.

Bits 6 and 7 are also used to select the counter for the control word you are writing.

Following reset, the control words for each register are undefined. You must program each timer to bring it into a known state. However, each counter OUT signal is reset to 0 following reset. The SPKR output, interrupt controller input IRQ0 (internal), Bit 5 of port 061h, and the internally generated Refresh request are each reset to 0 following reset.

Table 3-43. Timer Control Word Register

Bit #	Description
7:6	**COUNTER SELECT:** The Counter Selection bits select the counter the control word acts upon as shown below. The Read Back Command is selected when bits[7:6] are both 1. **Bit 7** **Bit 6** **Function** 0 0 Counter 0 select 0 1 Counter 1 select 1 0 Counter 2 select 1 1 Read Back Command (see Section 3.3.2)
5:4	**READ/WRITE SELECT:** Bits [5:4] are the read/write control bits. The Counter Latch Command is selected when bits[5:4] are both 0. The read/write options include r/w least significant byte, r/w most significant byte, or r/w the LSB and then the MSB. The actual counter programming is done through the counter I/O port (040h, 041h, and 042h for counters 0, 1, and 2, respectively). **Bit 5** **Bit 4** **Function** 0 0 Counter Latch Command (see Section 3.3.3) 0 1 R/W Least Significant Byte (LSB) 1 0 R/W Most Significant Byte (MSB) 1 1 R/W LSB then MSB
3:1	**COUNTER MODE SELECTION:** Bits [3:1] select one of six possible modes of operation for the counter as shown below. **Bit 3** **Bit 2** **Bit 1** **Mode** **Function** 0 0 0 0 Out signal on end of count (= 0) 0 0 1 1 Hardware retriggerable one-shot X 1 0 2 Rate generator (divide by n counter) X 1 1 3 Square wave output 1 0 0 4 Software triggered strobe 1 0 1 5 Hardware triggered strobe
0	**BINARY/BCD COUNTDOWN SELECT:** When bit 0 = 0, a binary countdown is used. The largest possible binary count is 2^{16}. When bit 0 = 1, a binary coded decimal (BCD) count is used. The largest BCD count allowed is 10^4.

3.3.2 TIMER READ BACK COMMAND REGISTER

Register Name: Timer Read Back Command

Register Location: 043h—Timer 1
04Bh—Timer 2

Default Value: xxh

Attribute: Write Only

Size: 8 bits

The Read-Back command is used to determine the count value, programmed mode, and current states of the OUT pin and Null Count flag of the selected counter or counters. The Read-Back command is written to the Control Word register, which latches the current states of the above mentioned variables. The value of the counter and its status may then be read by I/O access to the counter address.

Status and/or count may be latched on one, two, or all three of the counters by selecting the counter during the write. The Count latched will stay latched until read, regardless of further latch commands. The count must be read before newer latch commands latch a new count. The Status latched by the read-back command will also remain latched until after a read to the counter's I/O port. To reiterate, the Status and Count are unlatched only after a counter read of the Status register, the Count register, or the Status and Count register in succession.

Both count and status of the selected counter(s) may be latched simultaneously by setting both the COUNT# and STATUS# Bits [5:4] = 00b. This is functionally the same as issuing two consecutive, separate read-back commands. As stated above, if multiple count and/or status read-back commands are issued to the same counter(s) without any intervening reads, all but the first are ignored.

If both count and status of a counter are latched, the first read operation from that counter will return the latched status, regardless of which was latched first. The next one or two reads (depending on whether the counter is programmed for one or two byte counts) return the latched count. Subsequent reads return an unlatched count.

A register description of the Status Byte read follows later in this Section. Note that bit definitions for a write to this port changed when the read-back command was selected, when compared to a normal control word write to this same port.

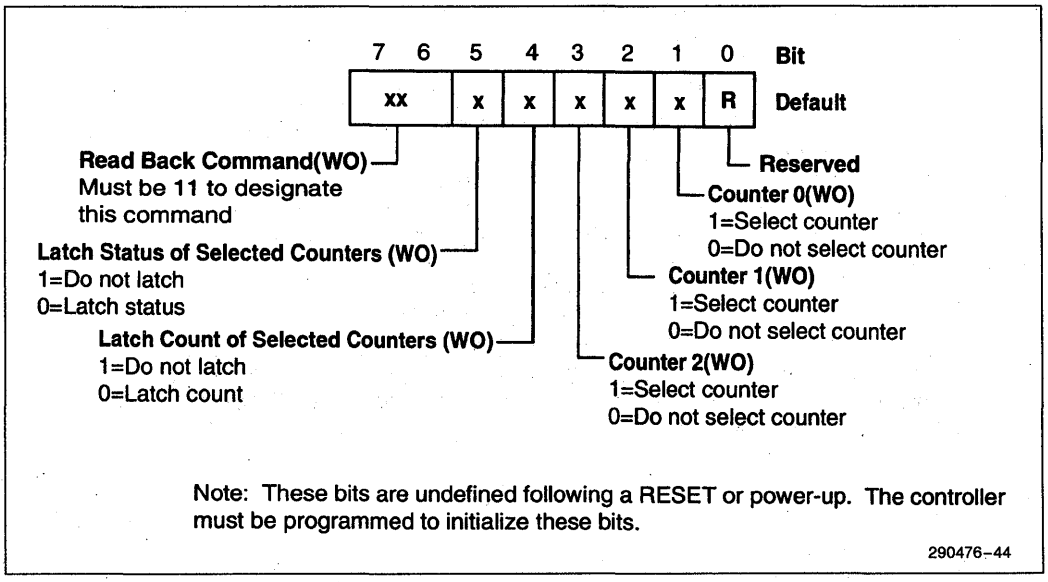


Figure 3-42. Timer Read Back Command Register

Table 3-44. Timer Read Back Command Register

Bit #	Description
7:6	**READ BACK COMMAND:** When Bits[7:6] are both "1", the read-back command is selected during a write to the control word. The normal meanings (mode, countdown, r/w select) of the bits in the control register at I/O address 043h change when the read-back command is selected. Following the read-back command, I/O reads from the selected counter's I/O addresses produce the current latch status, the current latched count, or both if Bits 4 and 5 are both "0".
5	**LATCH STATUS OF SELECTED COUNTERS:** When Bit 5 is a "1", the Current Count value of the selected counters will be latched. When Bit 4 is a "0", the Status will not be latched.
4	**LATCH COUNT OF SELECTED COUNTERS:** When Bit 4 is a "1", the Status of the selected counters will be latched. When Bit 4 is a "0", the Status will not be latched. The Status byte format is described in the next register description.
3	**COUNTER 2:** Counter 2 is selected for the latch command selected with Bits 4 and 5 if Bit 3 is a "1". If Bit 3 is a "0", Status and/or Count will not be latched.
2	**COUNTER 1:** Counter 1 is selected for the latch command selected with Bits 4 and 5 if Bit 2 is a "1". If Bit 2 is a "0", Status and/or Count will not be latched.
1	**COUNTER 0:** Counter 0 is selected for the latch command selected with Bits 4 and 5 if Bit 1 is a "1". If Bit 1 is a "0", Status and/or Count will not be latched.
0	**RESERVED:** Must be 0.

3.3.3 COUNTER LATCH COMMAND REGISTER

Register Name: Counter Latch Command
Register Location: 043h—Timer 1
04Bh—Timer 2
Default Value: xxh
Attribute: Write Only
Size: 8 bits

The Counter Latch command latches the current count value at the time the command is received. This command is used to insure that the count read from the counter is accurate (particularly when reading a two-byte count). The count value is then read from each counter's Count register. One, two or all three counters may be latched with one counter latch command.

If a Counter is latched once and then, some time later, latched again before the count is read, the second Counter Latch Command is ignored. The count read will be the count at the time the first Counter Latch Command was issued.

The count must be read according to the programmed format. Specifically, if the Counter is programmed for two byte counts, two bytes must be read. The two bytes do not have to be read one right after the other; read, write, or programming operations for other Counters may be inserted between them.

One precaution is worth noting. If a Counter is programmed to read/write two-byte counts, a program must not transfer control between reading the first and second byte to another routine which also reads from that same Counter. Otherwise, an incorrect count will be read. Finish reading the latched two-byte count before transferring control to another routine.

Note that bit definitions for a write to this port have changed when the read-back command was selected, when compared to a normal control word write to this same port.

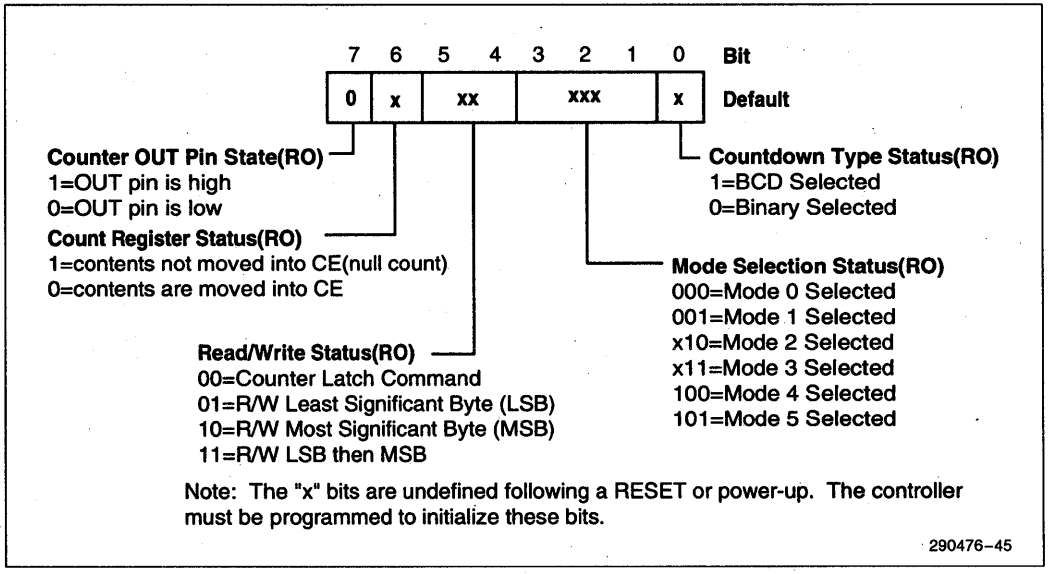


Figure 3-43. Counter Latch Command Register

Table 3-45. Counter Latch Command Register

Bit #	Description
7:6	**COUNTER SELECTION:** Bits 6 and 7 are used to select the counter for latching. **Bit 7** **Bit 6** **Function** 0 0 Latch counter 0 select 0 1 Latch counter 1 select 1 0 Latch counter 2 select 1 1 Read Back Command select
5:4	**SPECIFIES COUNTER LATCH COMMAND:** When Bits[5:4] are both "0", the Counter Latch command is selected during a write to the control word. The normal meanings (mode, countdown, r/w select) of the bits in the control register at I/O address 043h change when the Counter Latch command is selected. Following the Counter Latch command, I/O reads from the selected counter's I/O addresses produce the current latched count.
3:0	**RESERVED:** Must be 0.

3.3.4 TIMER STATUS BYTE FORMAT REGISTER

Register Name: Timer Status Byte Format

Register Location: 040h—Timer 1, Counter 0
041h—Timer 1, Counter 1
042h—Timer 1, Counter 2
048h—Timer 2, Counter 0
04Ah—Timer 2, Counter 2

Default Value: 0xxxxxxxb

Attribute: Read Only

Size: 8 bits per counter

Each Counter's Status Byte may be read following a Timer Read-Back Command. The Read-Back command is programmed through the counter control register. If "Latch Status" is chosen as a Read-Back option for a given counter, the next read from the counter's I/O port address returns the Status byte.

The Status byte returns the countdown type, either BCD or binary; the Counter Operational Mode; the Read/Write Selection status; the Null count, also referred to as the Count Register Status; and the current State of the counter OUT pin.

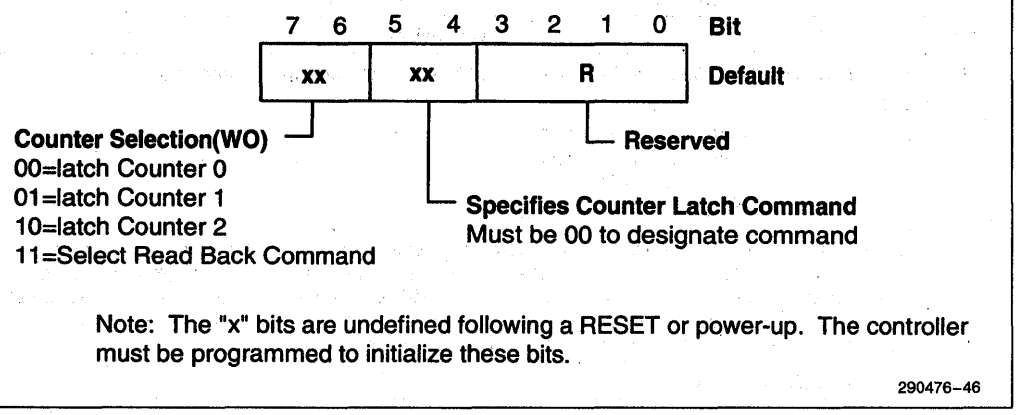


Figure 3-44. Timer Status Byte Format Register

Table 3-46. Timer Status Byte Format Register

Bit #	Description
7	**COUNTER OUT PIN STATE:** When this bit is a "1", the OUT pin of the counter is also a "1". When this bit is a "0", the OUT pin of the counter is also a "0".
6	**COUNT REGISTER STATUS:** Also referred to as Null Count, indicates when the last count written to the Count Register (CR) has been loaded into the counting element (CE). The exact time this happens depends on the counter Mode and is described in the Mode definitions, but until the count is loaded into the counting element (CE), it can't be read from the counter. If the count is latched or read before the load time, the count value returned will not reflect the new count written to the register. When Bit 6 is a "0", the count has been transferred from CR to CE and is available for reading. When Bit 6 is a "1", the Null count condition exists. The count has not been transferred from CR to CE and is not yet available for reading.
5:4	**READ/WRITE STATUS:** Bits[5:4] reflect the read/write selection made through Bits[5:4] of the control register. The binary codes returned during the status read match the codes used to program the counter read/write selection.
3:1	**MODE SELECTION STATUS:** Bits[3:1] return the counter mode programming. The binary code returned matches the code used to program the counter mode, as listed under the bit function above.
0	**COUNTDOWN TYPE STATUS:** Bit 0 reflects the current countdown type, either 0 for binary countdown or a 1 for binary coded decimal (BCD) countdown.

3.3.5 COUNTER ACCESS PORTS

Register Name: Counter Access Ports

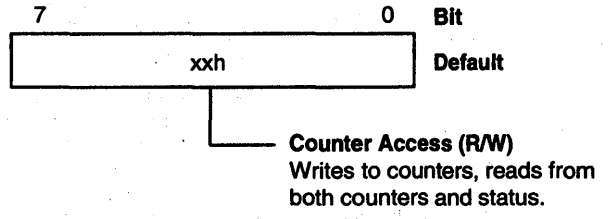
Register Location: 040h—Timer 1, Counter 0
041h—Timer 1, Counter 1
042h—Timer 1, Counter 2
048h—Timer 2, Counter 0
04Ah—Timer 2, Counter 2

Default Value: xxh

Attribute: Read/Write

Size: 8 bits per counter

Each of these I/O ports is used for writing count values to the count registers; reading the current count value from the counter by either an I/O read, after a counter-latch command, or after a read-back command; and reading the Status byte following a read-back command.



Note: These bits are undefined following a RESET or power-up. The controller must be programmed to initialize these bits.

290476-47

Figure 3-45. Counter Access Ports Register

Table 3-47. Counter Access Ports Register

Bit #	Description
7:0	**COUNTER ACCESS:** Each counter I/O port address is used to program the 16-bit count register. The order of programming, either LSB only, MSB only, or LSB then MSB, is defined with the Counter Control register at I/O port address 043h. The counter I/O port is also used to read the current count from the count register, and return the status of the counter programming following a read-back command.

3.4 Interrupt Controller Registers

The ESC contains an EISA compatible interrupt controller that incorporates the functionality of two 82C59 interrupt controllers. The interrupt registers control the operation of the interrupt controller and can be accessed from the EISA Bus via I/O space. This section describes the Interrupt registers. The operation of the Interrupt Controller is described in Chapter 9.0.

3.4.1 ICW1—INITIALIZATION COMMAND WORD 1

Register Name:	Initialization Command Word 1
Register Location:	020h—INT CNTRL-1 0A0h—INT CNTRL-2
Default Value:	xxh
Attribute:	Write Only
Size:	8 bits per controller

A write to Initialization Command Word One starts the interrupt controller initialization sequence. Addresses 020h and 0A0h are referred to as the base addresses of CNTRL-1 and CNTRL-2 respectively.

An I/O write to the CNTRL-1 or CNTRL-2 base address with Bit 4 equal to 1 is interpreted as ICW1. For ESC-based EISA systems, three I/O writes to "base address + 1" must follow the ICW1. The first write to "base address + 1" performs ICW2, the second write performs ICW3, and the third write performs ICW4.

ICW1 starts the initialization sequence during which the following automatically occur:

1. The edge sense circuit is reset, which means that following initialization, an interrupt request (IRQ) input must make a low-to-high transition to generate an interrupt.
2. The Interrupt Mask register is cleared.
3. IRQ7 input is assigned priority 7.
4. The slave mode address is set to 7.
5. Special Mask Mode is cleared and Status Read is set to IRR.
6. If IC4 was set to "0", then all functions selected by ICW4 are set to zero. However, ICW4 must be programmed in the ESC implementation of this interrupt controller, and IC4 must be set to a "1".

ICW1 has three significant functions within the ESC interrupt controller configuration. ICW4 is needed, so Bit 0 must be programmed to a "1". There are two interrupt controllers in the system, so Bit 1, SNGL, must be programmed to a 0 on both CNTRL-1 and CNTRL-2, to indicate a cascade configuration. Bit D4 must be a 1 when programming ICW1. OCW2 and OCW3 are also addressed at the same port as ICW1. This bit indicates that ICW1, and not OCW2 or OCW3, will be programmed during the write to this port.

Bit 2, ADI, and Bits [7:5], A7–A5, are specific to an MSC-85 implementation. These bits are not used by the ESC interrupt controllers. Bits [7:5,2] should each be initialized to "0".

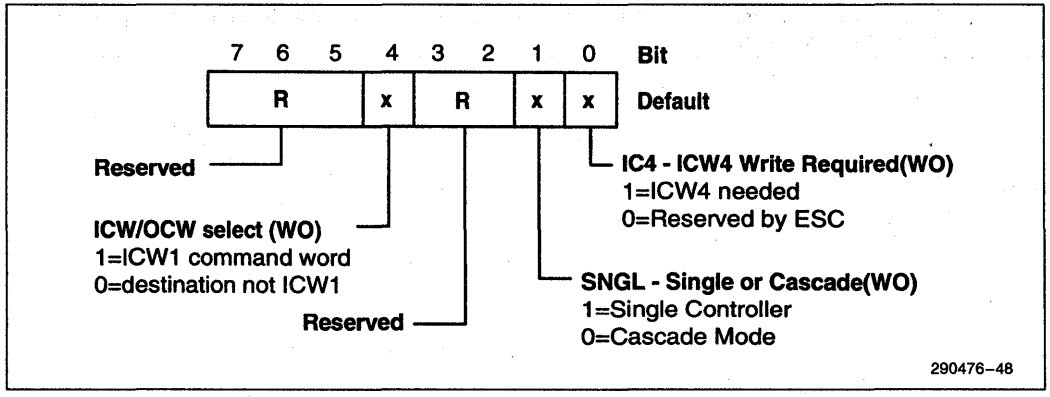


Figure 3-46. Initialization Command Word 1 Register

Table 3-48. Initialization Command Word 1 Register

Bit #	Description
7:5	**RESERVED:** A7–A5 are MCS-85 implementation specific bits. They are not needed by the ESC. These bits should be 000b when programming the ESC.
4	**ICW/OCW SELECT:** Bit 4 must be a 1 to select ICW1. After the fixed initialization sequence to ICW1, ICW2, ICW3, and ICW4, the controller base address is used to write to OCW2 and OCW3. Bit 4 is a 0 on writes to these registers. A 1 on this bit at any time will force the interrupt controller to interpret the write as an ICW1. The controller will then expect to see ICW2, ICW3, and ICW4.
3	**RESERVED:** This bit is not used in the ESC.
2	**RESERVED:** ADI: Ignored for the ESC.
1	**SNGL:** This bit must be programmed to a 0 to indicate that two interrupt controllers are operating in cascade mode on the ESC.
0	**IC4:** This bit must be set to a "1". IC4 indicates that ICW4 needs to be programmed. The ESC requires that ICW4 be programmed to indicate that the controllers are operating in an 80x86 type system.

3.4.2 ICW2—INITIALIZATION COMMAND WORD 2

Register Name: Initialization Command Word 2

Register Location: 021h—INT CNTRL-1
0A1h—INT CNTRL-2I

Default Value: xxh

Attribute: Write Only

Size: 8 bits per controller

ICW2 is used to initialize the interrupt controller with the five most significant bits of the interrupt vector address. The value programmed for Bits[7:3] is used by the CPU to define the base address in the interrupt vector table for the interrupt routines associated with each IRQ on the controller. Typical ISA ICW2 values are 04h for CNTRL-1 and 70h for CNTRL-2. Section 9.8.1 of the Interrupt Unit Functional Description contains a table detailing the interrupt vectors for each interrupt request level, as they would appear when the vector is driven onto the data bus.

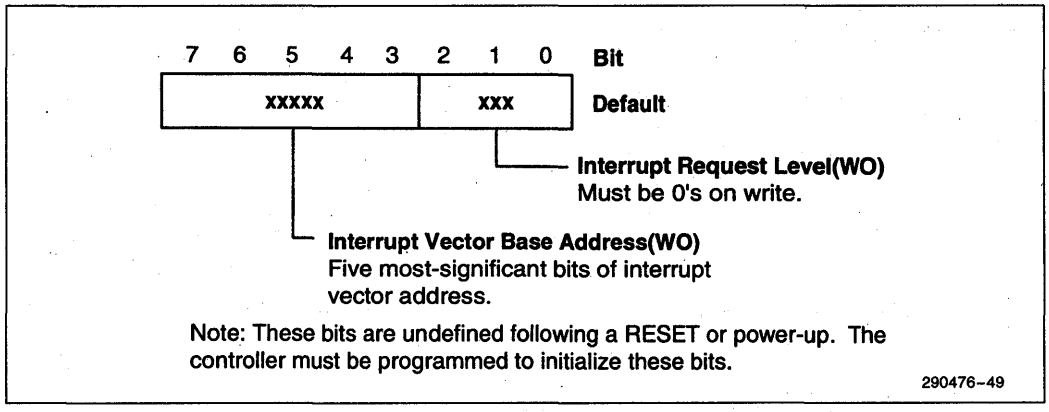


Figure 3-47. Initialization Command Word 2 Register

Table 3-49. Initialization Command Word 2 Register

Bit #	Description
7:3	**INTERRUPT VECTOR BASE ADDRESS:** Bits [7:3] define the base address in the interrupt vector table for the interrupt routines associated with each interrupt request level input. For CNTRL-1, a typical value is 00001b, and for CNTRL-2, 10000b. The interrupt controller combines a binary code representing the interrupt level to receive service with this base address to form the interrupt vector that is driven out onto the bus. For example, the complete interrupt vector for IRQ[0] (CNTRL-1), would be 0000 1000b (CNTRL-1 [7:3] = 00001b and 000b representing IRQ[0]). This vector is used by the CPU to point to the address information that defines the start of the interrupt routine.
2:0	**INTERRUPT REQUEST LEVEL:** When writing ICW2, these bits should all be "0". During an interrupt acknowledge cycle, these bits will be programmed by the interrupt controller with the interrupt code representing the interrupt level to be serviced. This interrupt code is combined with Bits [7:3] to form the complete interrupt vector driven onto the data bus during the second INTA# cycle. The table in Section 9.8.1 outlines each of these codes. The code is a simple three bit binary code: 000b represents IRQ0 (IRQ8), 001b IRQ1 (IRQ9), 010b IRQ2 (IRQ10), and so on until 111b IRQ7 (IRQ15).

3.4.3 ICW3—INITIALIZATION COMMAND WORD 3 (MASTER)

Register Name: Initialization Command Word 3—Controller 1—Master Unit

Register Location: 021h—INT CNTRL-1

Default Value: xxh

Attribute: Write Only

Size: 8 bits

The meaning of ICW3 differs between CNTRL-1 and CNTRL-2. On CNTRL-1, the master controller, ICW3 indicates which CNTRL-1 IRQ line physically connects the INT output of CNTRL-2 to CNTRL-1. ICW3 must be programmed to 04h, indicating the cascade of the CNTRL-2 INT output to the IRQ[2] input of CNTRL-1.

An interrupt request on IRQ2 causes CNTRL-1 to enable CNTRL-2 to present the interrupt vector address during the second interrupt acknowledge cycle.

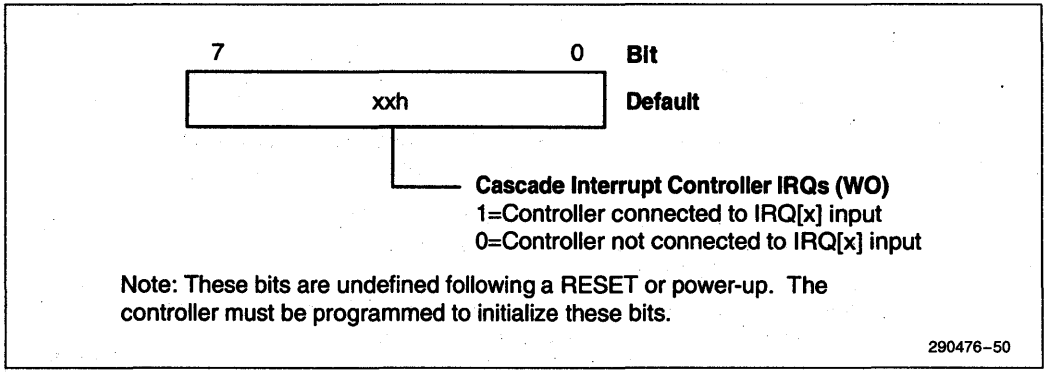


Figure 3-48. Initialization Command Word 3 Register (Master)

Table 3-50. Initialization Command Word 3 Register (Master)

Bit #	Description
7:3, 1:0	**CASCADE INTERRUPT CONTROLLER IRQs:** Bits [7:3] and Bits [1:0] must be programmed to "0".
2	**CASCADE INTERRUPT CONTROLLER IRQs:** Bit 2 must always be programmed to a "1". This bit indicates that CNTRL-2, the slave controller, is cascaded on interrupt request line two (IRQ[2]). When an interrupt request is asserted to CNTRL-2, the IRQ goes through the priority resolver. After the slave controller priority resolution is finished, the INT output of CNTRL-2 is asserted. However, this INT assertion does not go directly to the CPU. Instead, the INT assertion cascades into IRQ[2] on CNTRL-1. IRQ[2] must go through the priority resolution process on CNTRL-1. If it wins the priority resolution on CNTRL-1 and the CNTRL-1 INT signal is asserted to the CPU, the returning interrupt acknowledge cycle is really destined for CNTRL-2. The interrupt was originally requested at CNTRL-2, so the interrupt acknowledge is destined for CNTRL-2, and not a response for IRQ[2] on CNTRL-1. When an interrupt request from IRQ[2] wins the priority arbitration, in reality an interrupt from CNTRL-2 has won the arbitration. Because Bit 2 of ICW3 on the master is set to "1", the master knows which identification code to broadcast on the internal cascade lines, alerting the slave controller that it is responsible for driving the interrupt vector during the second INTA# pulse.

3.4.4 ICW3—INITIALIZATION COMMAND WORD 3 (SLAVE)

Register Name: Initialization Command Word 3—Controller 2—Slave Unit

Register Location: INT CNTRL-2 Port Address—0A1h

Default Value: xxh

Attribute: Write Only

Size: 8 bits

On CNTRL-2 (the slave controller), ICW3 is the slave identification code broadcast by CNTRL-1 from the trailing edge of the first INTA# pulse to the trailing edge of the second INTA# pulse. CNTRL-2 compares the value programmed in ICW3 with the incoming identification code. The code is broadcast over three ESC internal cascade lines. ICW3 must be programmed to 02h for CNTRL-2. When 010b is broadcast by CNTRL-1 during the INTA# sequence, CNTRL-2 assumes responsibility for broadcasting the interrupt vector during the second interrupt acknowledge cycle.

As an illustration, consider an interrupt request on IRQ[2] of CNTRL-1. By definition, a request on IRQ[2] must have been asserted by CNTRL-2. If IRQ[2] wins the priority resolution on CNTRL-1, the interrupt acknowledge cycle returned by the CPU following the interrupt is destined for CNTRL-2, not CNTRL-1. CNTRL-1 will see the INTA# signal, and knowing that the actual destination is CNTRL-2, will broadcast a slave identification code across the internal cascade lines. CNTRL-2 will compare this incoming value with the 010b stored in ICW3. Following a positive decode of the incoming message from CNTRL-1, CNTRL-2 will drive the appropriate interrupt vector onto the data bus during the second interrupt acknowledge cycle.

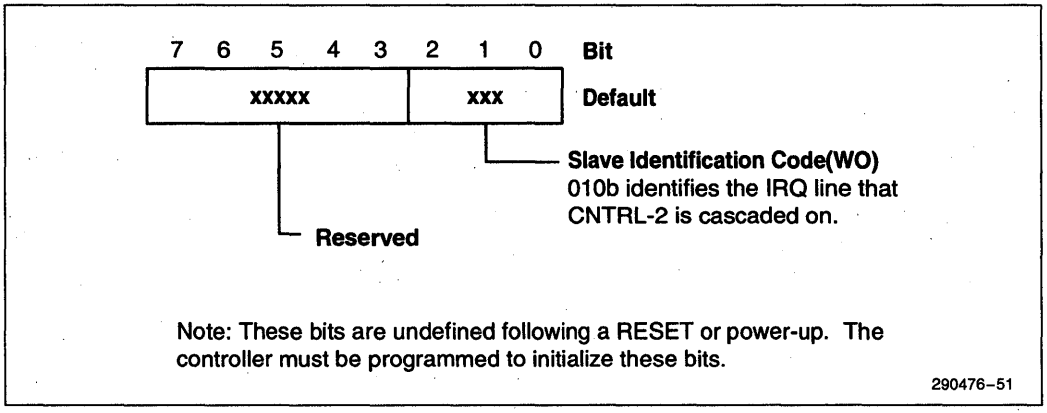


Figure 3-49. Initialization Command Word 3 Register (Slave)

Table 3-51. Initialization Command Word 3 Register (Slave)

Bit #	Description
7:3	**RESERVED:** Must be 0.
2:0	**SLAVE IDENTIFICATION CODE:** The Slave Identification code must be programmed to 010b during the initialization sequence. The code stored in ICW3 is compared to the incoming slave identification code broadcast by the master controller during interrupt acknowledge cycles.

3.4.5 ICW4—INITIALIZATION COMMAND WORD 4

Register Name: Initialization Command Word 4
Register Location: 021h—INT CNTRL-1
0A1h—INT CNTRL-2
Default Value: xxh
Attribute: Write Only
Size: 8 bits

Both ESC interrupt controllers must have ICW4 programmed as part of their initialization sequence. Minimally, the microprocessor mode bit, Bit 0, must be set to a 1 to indicate to the controller that it is operating in an 80x86 based system. Failure to program this bit will result in improper controller operation during interrupt acknowledge cycles. Additionally, the Automatic End of Interrupt (AEOI) may be selected, as well as the Special Fully Nested Mode (SFNM) of operation.

The default programming for ICW4 is 01h, which selects 80x86 mode, normal EOI, buffered mode, and special fully nested mode disabled.

Bits 2 and 3 must be programmed to 0 for the ESC interrupt unit to function correctly.

Both Bit 1, AEOI, and Bit 4, SFNM, can be programmed if the system developer chooses to invoke either mode.

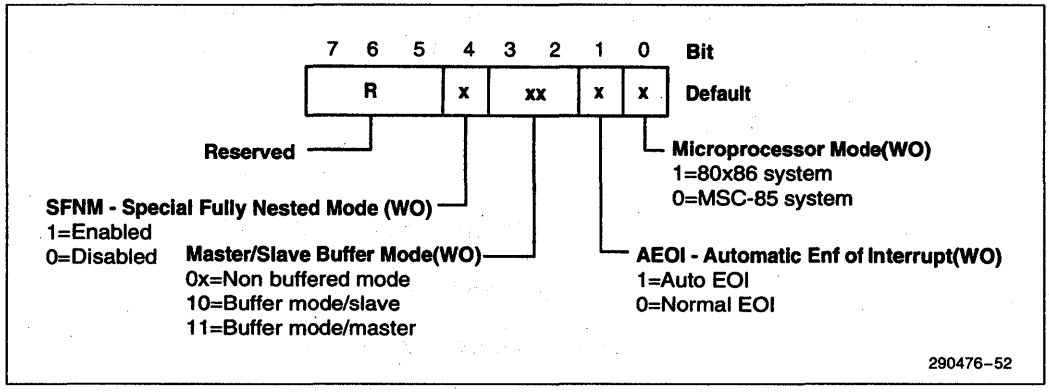


Figure 3-50. Operation Control Word 1 Register

Table 3-52. Operation Command Word 1 Register

Bit #	Description
7:5	**RESERVED:** Must be 0.
4	**SFNM:** Bit 4, SFNM, should normally be disabled by writing a 0 to this bit. If SFNM = 1, the special fully nested mode is programmed.
3:2	**MASTER/SLAVE BUFFER MODE—BUF:** Bit 3, BUF, must be programmed to 0 for the ESC. This is non-buffered mode. As illustrated in Figure 3-50, different programming options are offered for Bits 2 and 3. However, within the ESC interrupt unit, Bits 2 and 3 must always be programmed to 00b.
1	**AEOI:** Bit 1, AEOI, should normally be programmed to "0". This is the normal end of interrupt. If AEOI = 1, the automatic end of interrupt mode is programmed.
0	**MICROPROCESSOR MODE:** The Microprocessor Mode bit must be programmed to 1 to indicate that the interrupt controller is operating in an 80x86 based system. Never program this bit to "0".

3.4.6 OCW1—OPERATION CONTROL WORD 1

Register Name: Operation Control Word 1
Register Location: 021h—INT CNTRL-1
0A1h—INT CNTRL-2
Default Value: xxh
Attribute: Read/Write
Size: 8 bits

OCW1 sets and clears the mask bits in the interrupt Mask register (IMR). Each interrupt request line may be selectively masked or unmasked any time after initialization. A single byte is written to this register. Each bit position in the byte represents the same-numbered channel: Bit 0 = IRQ[0], Bit 1 = IRQ[1] and so on. Setting the bit to a 1 sets the mask, and clearing the bit to a 0 clears the mask. Note that masking IRQ[2] on CNTRL-1 will also mask all of controller 2's interrupt requests (IRQ8–IRQ15). Reading OCW1 returns the controller's mask register status.

The IMR stores the bits which mask the interrupt lines to be masked. The IMR operates on the IRR. Masking of a higher priority input will not affect the interrupt request lines of lower priority.

Unlike status reads of the ISR and IRR, for reading the IMR, no OCW3 is needed. The output data bus will contain the IMR whenever I/O read is active and the I/O port address is 021h or 0A1h (OCW1).

All writes to OCW1 must occur following the ICW1–ICW4 initialization sequence, since the same I/O ports are used for OCW1, ICW2, ICW3 and ICW4.

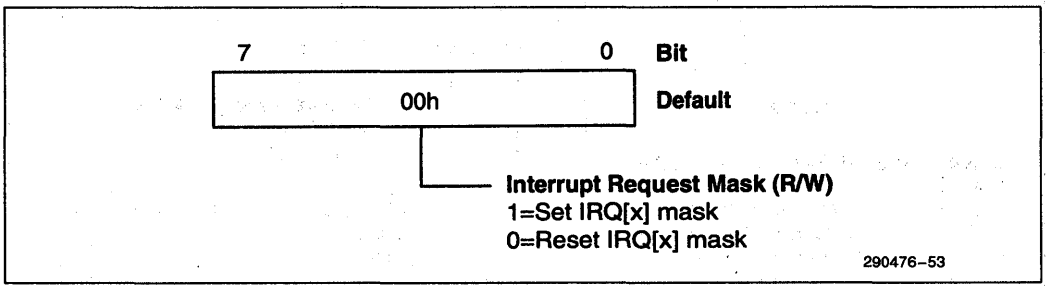


Figure 3-51. Operation Control Word 1 Register

Table 3-53. Operation Control Word 1 Register

Bit #	Description
7:0	**INTERRUPT REQUEST MASK:** When a 1 is written to any bit in this register, the corresponding IRQ[x] line is masked. For example, if Bit 4 is set to a "1", then IRQ[4] will be masked. Interrupt requests on IRQ[4] will not set Channel 4's interrupt request register (IRR) bit as long as the channel is masked. When a 0 is written to any bit in this register, the corresponding IRQ[x] mask bit is cleared, and interrupt requests will again be accepted by the controller. Note that masking IRQ[2] on CNTRL-1 will also mask the interrupt requests from CNTRL-2, which is physically cascaded to IRQ[2].

3.4.7 OCW2—OPERATION CONTROL WORD 2

Register Name: Operation Control Word 2
Register Location: 020h—INT CNTRL-1
0A0h—INT CNTRL-2
Default Value: xxh
Attribute: Write Only
Size: 8 bits

OCW2 controls both the Rotate Mode and the End of Interrupt Mode, and combinations of the two. The three high order bits in an OCW2 write represent the encoded command. The three low order bits are used to select individual interrupt channels during three of the seven commands. The three low order bits (labeled L2, L1 and L0) are used when Bit 6, the SL bit, is set to a 1 during the command.

Following a reset and ICW initialization, the controller enters the fully nested mode of operation. Non-specific EOI without rotation is the default. Both rotation mode and specific EOI mode are disabled following initialization.

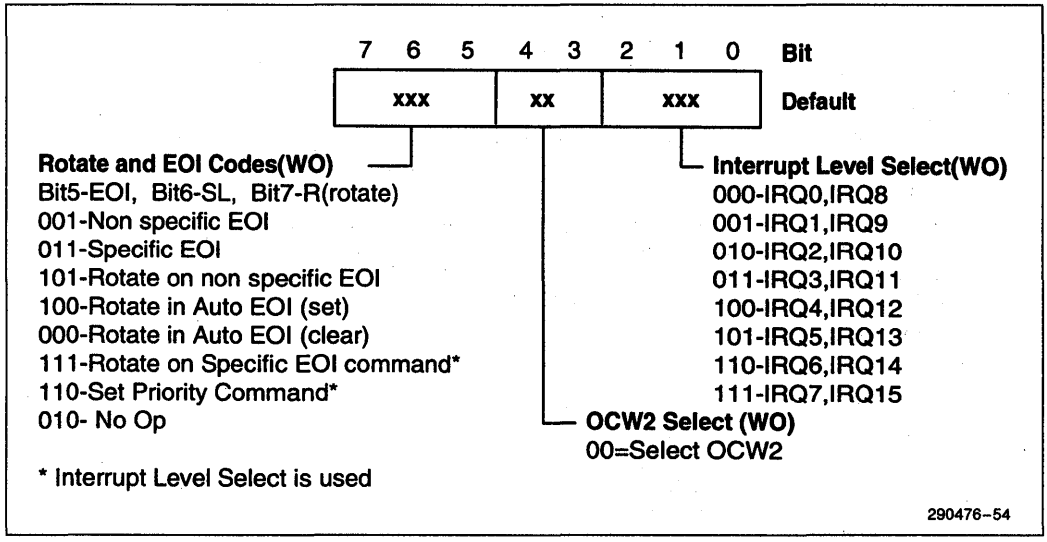


Figure 3-52. Operation Control Word 2 Register

Table 3-54. Operation Control Word 2 Register

Bit #	Description
7:5	**ROTATE AND EOI CODES:** R, SL, EOI—These three bits control the Rotate and End of Interrupt modes and combinations of the two. A chart of these combinations is listed above under the bit definition. **Bit 7 / Bit 6 / Bit 5 / Function** 0 0 1 Non-specific EOI command 0 1 1 Specific EOI Command 1 0 1 Rotate on Non-Specific EOI Command 1 0 0 Rotate in Auto EOI Mode (Set) 0 0 0 Rotate in Auto EOI Mode (Clear) 1 1 1 Rotate on Specific EOI Command* 1 1 0 Set Priority Command* 0 1 0 No Operation *L0–L2 Are Used
4:3	**OCW2 SELECT:** When selecting OCW2, Bits 3 and 4 must both be "0". If Bit 4 is a "1", the interrupt controller interprets the write to this port as an ICW1. Therefore, always ensure that these bits are both 0 when writing an OCW2.
2:0	**INTERRUPT LEVEL SELECT (L2, L1, L0):** L2, L1, and L0 determine the interrupt level acted upon when the SL bit is active. A simple binary code, outlined above, selects the channel for the command to act upon. When the SL bit is inactive, these bits do not have a defined function; programming L2, L1 and L0 to 0 is sufficient in this case. **Bit 2 / Bit 1 / Bit 0 / Interrupt Level** 0 0 0 IRQ 0(8) 0 0 1 IRQ 1(9) 0 1 0 IRQ 2(10) 0 1 1 IRQ 3(11) 1 0 0 IRQ 4(12) 1 0 1 IRQ 5(13) 1 1 0 IRQ 6(14) 1 1 1 IRQ 7(15)

3.4.8 OCW3—OPERATION CONTROL WORD 3

Register Name: Operation Control Word 3

Register Location: 020h—INT CNTRL-1
0A0h—INT CNTRL-2

Default Value: x01xxx10b

Attribute: Read/Write

Size: 8 bits

OCW3 serves three important functions; Enable Special Mask Mode, Poll Mode control, and IRR/ISR register read control.

First, OCW3 is used to set or reset the Special Mask Mode (SMM). The Special Mask Mode can be used by an interrupt service routine to dynamically alter the system priority structure while the routine is executing, through selective enabling/disabling of the other channel's mask bits.

Second, the Poll Mode is enabled when a write to OCW3 is issued with Bit 2 equal to "1". The next I/O read to the Interrupt controller is treated like an interrupt acknowledge; a binary code representing the highest priority level interrupt request is released onto the bus.

Third, OCW3 provides control for reading the In-Service Register (ISR) and the Interrupt Request Register (IRR). Either the ISR or IRR is selected for reading with a write to OCW3. Bits 0 and 1 carry the encoded command to select either register. The next I/O read to the OCW3 port address will return the register status specified during the previous write. The register specified for a status read is retained by the interrupt controller. Therefore, a write to OCW3 prior to every status read command is unnecessary, provided the status read desired is from the register selected with the last OCW3 write.

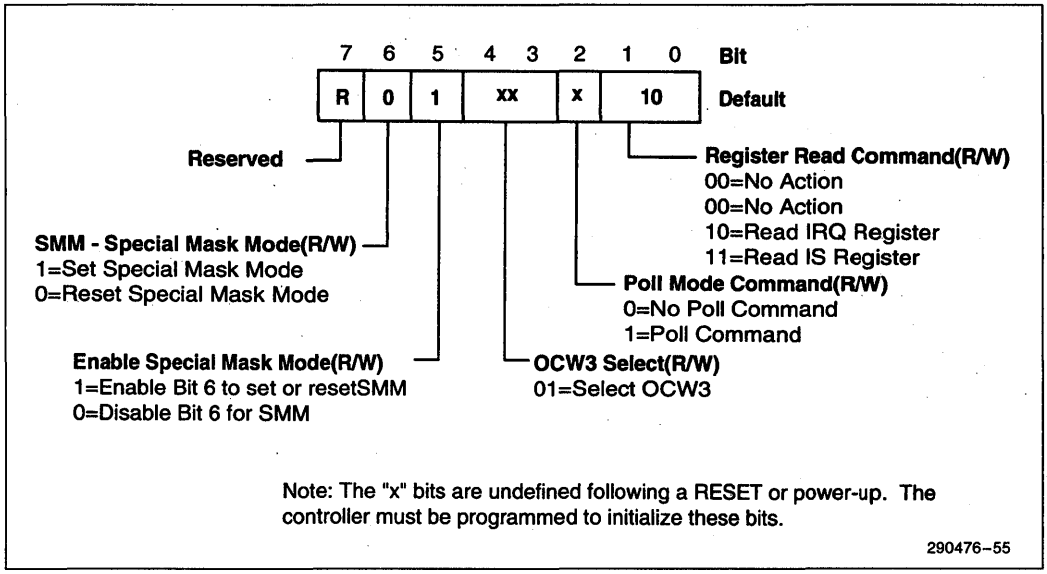


Figure 3-53. Operation Control Word 3 Register

Table 3-55. Operation Control Word 3 Register

Bit #	Description
7	**RESERVED:** Must be 0.
6	**SMM:** If ESMM = 1 and SMM = 1 the Interrupt Controller will enter Special Mask Mode. If ESMM 1 and SMM = 0 the Interrupt Controller will revert to normal mask mode. When ESMM = 0, SMM has no effect.
5	**ENABLE SPECIAL MASK MODE:** When this bit is set to 1 it enables the SMM bit to set or reset the Special Mask Mode. When ESMM = 0 the SMM bit becomes a "don't care".
4:3	**OCW3 SELECT:** When selecting OCW3, Bit 3 must be a 1 and Bit 4 must be "0". If Bit 4 is a "1", the Interrupt Controller interprets the write to this port as an ICW1. Therefore, always ensure that Bits[4:3] are "01b" when writing an OCW3.
2	**POLL MODE COMMAND:** When Bit 2 is a "0", the Poll command is not issued. When Bit 2 is a "1", the next I/O read to the Interrupt Controller is treated as an Interrupt Acknowledge cycle. An encoded byte is driven onto the data bus, representing the highest priority level requesting service.
1:0	**REGISTER READ COMMAND:** Bits [1:0] provide control for reading the In-Service Register (ISR) and the Interrupt Request Register (IRR). When bit 1 = 0, bit 0 will not affect the register read selection. When bit 1 = 1, bit 0 selects the register status returned following an OCW3 read. If bit 0 = 0, the IRR will be read. If bit 0 = 1, the ISR will be read. Following ICW initialization, the default OCW3 port address read will be "read IRR". To retain the current selection (read ISR or read IRR), always write a 0 to bit 1 when programming this register. The selected register can be read repeatedly without reprogramming OCW3. To select a new status register, OCW3 must be reprogrammed prior to attempting the read. **Bit 1** **Bit 0** **Function** 0 0 No Action 0 1 No Action 1 0 Read IRQ Register 1 1 Read IS Register

3.4.9 EDGE/LEVEL CONTROL REGISTER

Register Name: Edge/Level Control
Register Location: 04D0h—INT CNTRL-1
04D1h—INT CNTRL-1
Default Value: 00h
Attribute: Read/Write
Size: 8 bits

The Edge/Level Control Register is used to set the interrupts to be triggered by either the signal edge or the logic level. INT0, INT1, INT2, INT8, INT13 must be set to edge sensitive. After a reset all the INT signals are set to edge sensitive. Table 3-56 shows which bit numbers represent the various INT signals.

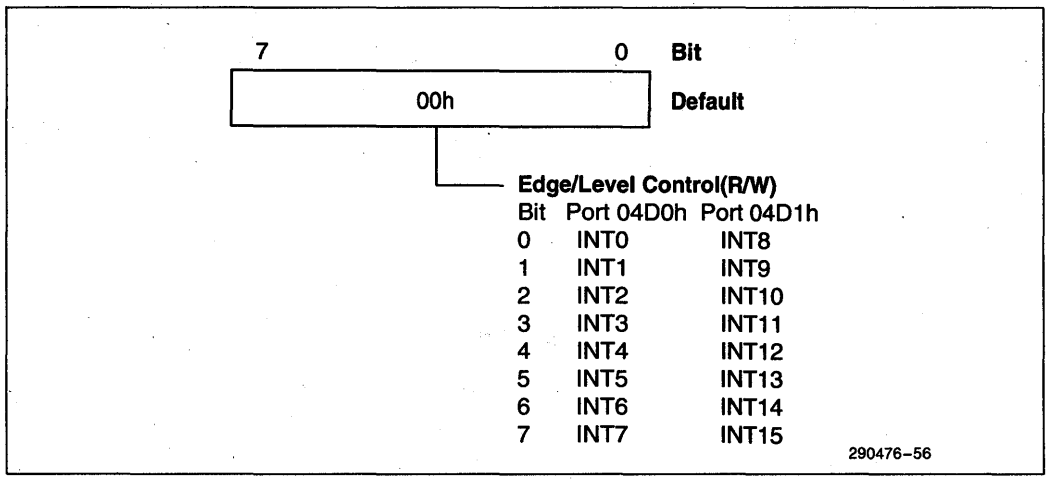


Figure 3-54. Edge/Level Select Register

Table 3-56. Edge/Level Select Register

Bit #	Description
7:0	**EDGE/LEVEL SELECT:** The bits select if the interrupts are triggered by either the signal edge or the logic level. A 0 bit represents an edge sensitive interrupt, and a 1 is for level sensitive. Bit[2:0] and bit 13 are must always be set to 0. After a reset or power-on these registers are set to 00h.

3.4.10 NMI STATUS AND CONTROL REGISTER

Register Name: NMI Status and Control
Register Location: 061h
Default Value: 00h
Attribute: Read/Write
Size: 8 bits

This register is used to check the status of different system components, control the output of the Speaker Counter (Timer 1, Counter 2), and gate the counter output that drives the SPKR signal.

Bits 4, 5, 6, and 7 are read-only. When writing to this port, these bits must be written as 0's. Bit 6 returns the IOCHK# NMI status. This input signal comes from the EISA bus. It is used for parity errors on memory cards plugged into the bus, and for other high priority interrupts. The current status of Bit 3 enables or disables this IOCHK# NMI source. Bit 5 is the current state of the OUT pin of Timer 1, Counter 2. Bit 4 toggles from 1–0 or from 0–1 after every Refresh cycle. Following reset, Bits 4 and 6 are both "0". Bit 5 is undetermined until Counter 2 is properly programmed. Bit 7 returns the PCI System Board Parity Error status (PERR#). If "0", Bit 7 indicates that PERR# was not pulsed active by a PCI agent. If "1", Bit 7 indicates that PERR# was pulsed active by a PCI agent and that an NMI will be issued to the CPU. This NMI can be disabled with Bit 2 of this register.

Bits 0–3 are both read and write. Bit 0 is the GATE input signal for Timer 1, Counter 2. The GATE input is used to disable counting in Counter 2. The Counter 2 output is ANDed with Bit 1 to form the SPKR output signal. Bit 1 gates the Counter 2 OUT value. When Bit 1 is disabled, the SPKR signal is disabled; when Bit 1 is enabled, the SPKR output follows the value at the OUT pin of Counter 2. The Counter 2 OUT pin status can be checked by reading port 061h and checking Bit 5. Bit 2 is used to enable the system board error (ERR#) signal. Bit 3 enables or disables the incoming IOCHK# NMI signal from the expansion bus. Each of these Bits is reset to 0 following reset.

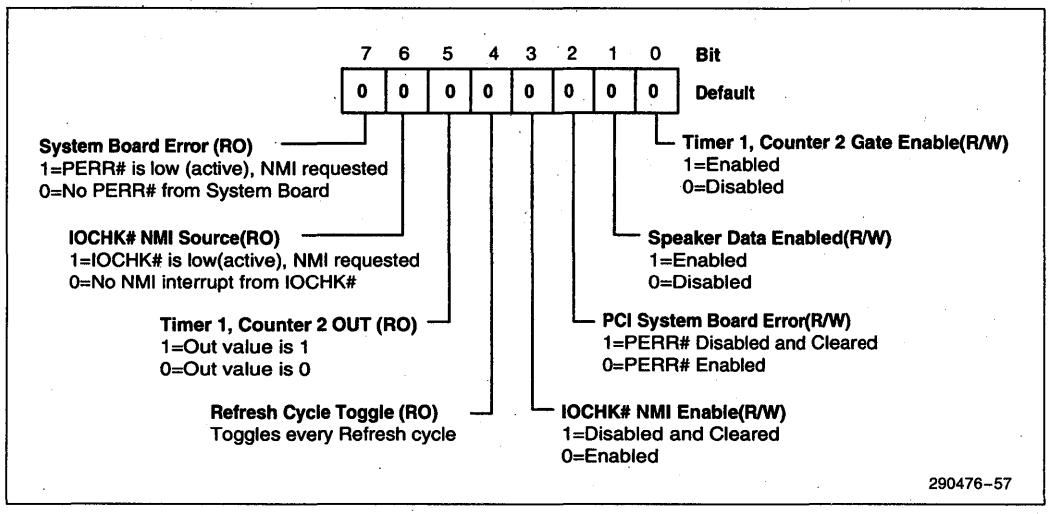


Figure 3-55. NMI Status and Control Register

Table 3-57. NMI Status and Control Register

Bit #	Description
7	**SYSTEM BOARD ERROR:** Bit 7 is set if a system board agent (PCI devices or main memory) detects a system board error and pulses the PCI ERR# line. This interrupt is enabled by setting Bit 2 to "0". To reset the interrupt, set Bit 2 to 0 and then set it to "1". This bit is read-only. When writing to port 061h, Bit 6 must be a "0".
6	**IOCHK# NMI SOURCE:** Bit 6 is set if an expansion board asserts IOCHK# on the ISA/ESC bus. This interrupt is enabled by setting Bit 3 to "0". To reset the interrupt, set Bit 3 to 0 and then set it to "1". This bit is read-only. When writing to port 061h, Bit 6 must be a "0".
5	**TIMER 1, COUNTER 2:** The Timer 1, Counter 2 OUT signal state is reflected in Bit 5. The value on this bit following a read is the current state of the Counter 2 OUT signal. Counter 2 must be programmed following a reset for this bit to have a determinate value. Bit 5 is read-only. When writing to port 061h, Bit 5 must be a "0".
4	**REFRESH CYCLE TOGGLE:** The Refresh Cycle Toggle signal toggles from either 0 to 1 or 1 to 0 following every refresh cycle. This bit is a 0 following reset. This bit is read-only. When writing to port 061h, Bit 4 must be a "0".
3	**IOCHK# NMI ENABLE:** When Bit 3 is a "1", IOCHK# NMI's are disabled and cleared, and when Bit 3 is a "0", IOCHK# NMI's are enabled. Following reset, Bit 3 is reset to 0 and IOCHK# NMI's are enabled.
2	**PCI SYSTEM BOARD ERROR:** When Bit 2 is a "1", the system board error is disabled and cleared. When Bit 2 is a "0", the system board parity error is enabled. Following reset, Bit 2 is a "0", and system board errors are enabled.
1	**SPEAKER DATA ENABLED:** Speaker Data Enable is ANDed with the Timer 1, Counter 2 OUT signal to drive the SPKR output signal. When Bit 1 is a "0", the result of the AND is always 0 and the SPKR output is always "0". When Bit 1 is a "1", the SPKR output is equivalent to the Counter 2 OUT signal value. Following reset, Bit 1 is a 0 and the SPKR output is low.
0	**TIMER 1, COUNTER 2 GATE ENABLE:** When Bit 0 is a "0", Timer 1, Counter 2 counting is disabled. Counting is enabled when Bit 0 is a "1". This bit controls the GATE input to Counter 2. Following reset, the value of this bit is 0 and counting is disabled.

3.4.11 NMI CONTROL AND REAL-TIME CLOCK ADDRESS

Register Name: NMI Enable/Disable and Real-Time Clock Address

Register Location: 070h

Default Value: See below

Attribute: Write Only

Size: 8 bits

The Mask register for the NMI interrupt is at I/O address 070h. The most-significant bit enables or disables all NMI sources including PERR#, SERR#, IOCHK#, Fail-Safe Timer, Bus Timeout, and the NMI Port. Write an 80h to port 70h to mask the NMI signal. This port is shared with the real-time clock. The real-time-clock uses the lower six bits of this port to address memory locations. Writing to port 70h sets both the enable/disable bit and the memory address pointer. Do not modify the contents of this register without considering the effects on the state of the other bits.

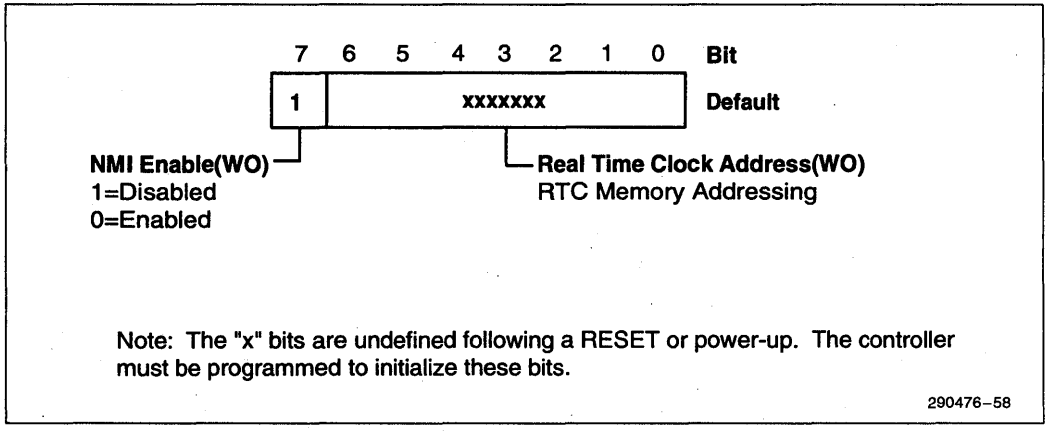


Figure 3-56. NMI Control and Real Time Clock Address

Table 3-58. NMI Control and Real Time Clock Address Register

Bit #	Description
7	**NMI ENABLE:** Setting Bit 7 to a 1 will disable all NMI sources. Resetting the bit to a 0 enables the NMI interrupt.
6:0	**REAL TIME CLOCK ADDRESS:** Used by the Real Time Clock on the Base I/O component to address memory locations. Not used for NMI enabling/disabling.

3.4.12 NMI EXTENDED STATUS AND CONTROL REGISTER

Register Name: NMI Extended Status and Control

Register Location: Port address-0461h

Default Value: See below

Attribute: Read/Write

Size: 8 bits

This register is used to check the status of different system components, control the output of the Speaker Counter (Timer 1, Counter 2), and gate the counter output that drives the SPKR signal.

Bits 4, 5, 6, and 7 are read-only. Bits 0–3 are both read and write. When writing to this port, these bits must be written as 0's. Bit 7 returns the Fail-Safe Timer Status. This input comes from Timer 2, Counter 0. The current status of Bit 2 enables or disables this Fail-Safe Timer NMI source. Bit 6 returns the Bus Timeout Status. Bit 6 is set if either a 64 BCLK or a 256 BCLK occurs. The current status of Bit 3 enables or disables this Fail-Safe Timer NMI source. If NMI is caused by a Bus Timeout, Bit 4 distinguished between the 8 μs (64 BCLK) and 32 μs (256 BCLK) timeout. Bit 5 is the current state of an I/O write to port 0462h. The current status of Bit 1 enables or disables Software generated NMI. Bit 0 controls the state of the RSTDRV output signal. If Bit 0 is set to "1", the RSTDRV signal is asserted and a system bus reset is performed. Bit 0 should be set long enough (>8 BCLKs) for the system bus devices to be properly reset.

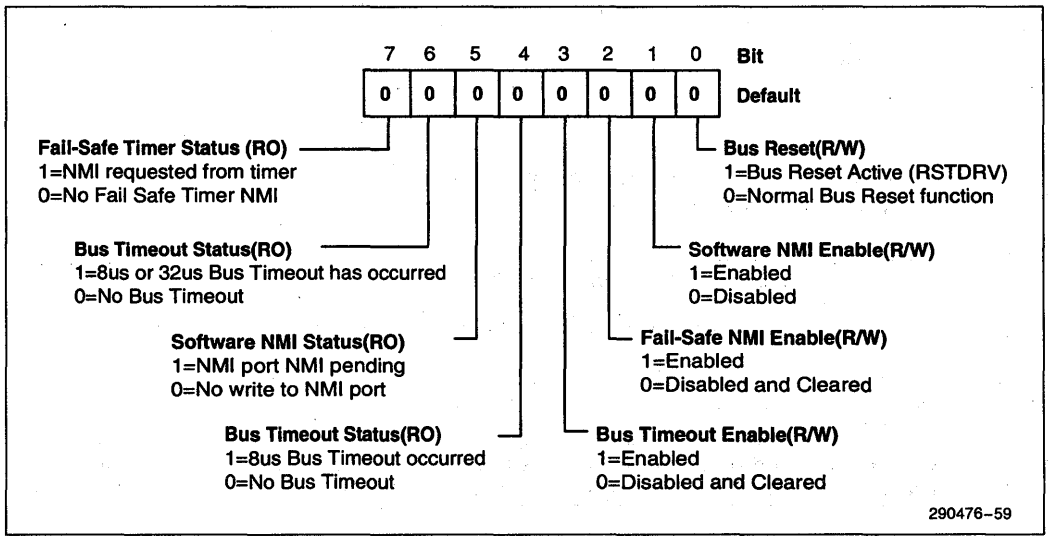


Figure 3-57. NMI Extended Status and Control Register

Table 3-59. NMI Extended Status and Control Register

Bit #	Description
7	**FAIL-SAFE TIMER STATUS:** This bit indicates the status of the Fail-Safe Timer. When Timer 2, Counter 0 count expires, this bit is set to a 1 if Bit 2 as previously been set to "1". A value of 0 indicates that the current NMI was not caused by the Fail-Safe Timer. A value of 1 indicates that the Fail-Safe timer has timed out.
6	**BUS TIMEOUT STATUS:** This bit indicates the status of Bus master timeout logic. If this bit is "0", the Bus Master timeout logic has not detected a bus timeout. If this bit is "1", the bus master timeout logic has detected a bus timeout.
5	**SOFTWARE NMI STATUS:** This bit indicates the status of the Software NMI port writes. A write to I/O port 0462 of any value will set this bit to 1 if Bit 1 is set to "1". If this bit is "0", the current NMI was not caused by a write to the NMI Port. If this bit is "1", the current NMI was caused by a write to the NMI Port.
4	**BUS TIMEOUT STATUS:** This bit indicates the status of the 8 μs EISA Bus master timeout event. If the bit is "0", the current NMI was not caused by the 8 μs EISA bus master timeout. If this bit is "1", the current NMI was caused by this bus timeout.
3	**BUS TIMEOUT ENABLE:** This bit enables/disables NMI EISA bus timeout. If this bit is "0", an NMI will not be generated for bus timeout. Also the NMI condition caused by the Bus timeout will be cleared. If this bit is 1 an NMI will be generated when Timer 2 Counter 0 count expires.
2	**FAIL-SAFE NMI ENABLE:** This bit enables/disables NMI when the Fail-Safe Timer times out. If this bit is "0", an NMI will not be generated when the Timer 2 Counter 0 count expires. Also the NMI condition caused by the Fail-Safe Timer will be cleared. If this bit is 1 an NMI will be generated when Timer 2 Counter 0 count expires.
1	**SOFTWARE NMI ENABLE:** This bit enables/disables software generated NMI. If this Bit is "0", a write to I/O port 0462h will not generate an NMI. If this bit is 1, NMI will be generated for a write to I/O port 0462h.
0	**BUS RESET:** When Bit 0 is a "0", RSTDRV signal functions as a normal reset drive signal. When Bit 0 is "1", the RSTDRV signal is asserted. Following reset, Bit 0 is a 0 and the RSTDRV output is low.

3.4.13 SOFTWARE NMI GENERATION

Register Name: Software NMI Generation
Register Location: 462h
Default Value: xxh
Attribute: Write Only
Size: 8 bits

A write to this port with any data will cause an NMI. This port provides a software mechanism to cause an NMI if interrupts are enabled.

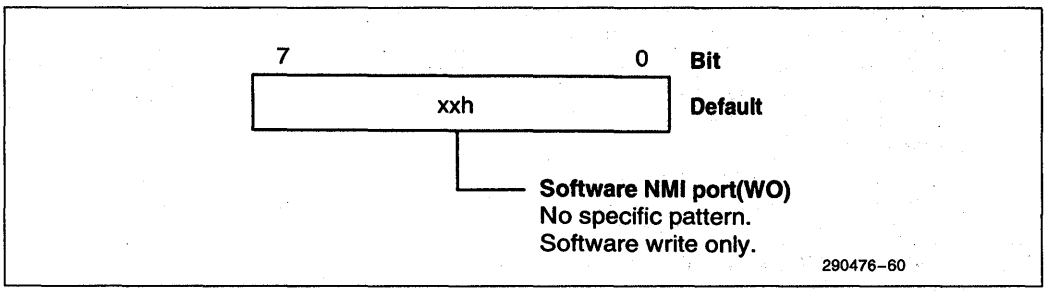


Figure 3-58. Software NMI Generation Register

Table 3-60. Configuration RAM Page Register

Bit #	Description
7:0	**SOFTWARE NMI PORT:** The bit pattern is not specific. A write to this port will generate a Software NMI if enabled.

3.5 EISA Configuration, Floppy Support, and Port 92h

3.5.1 CONFIGURATION RAM PAGE REGISTER

Register Name: Configuration RAM Page
Register Location: 0C00h
Default Value: xxx00000b
Attribute: Read/Write
Size: 8 bits

This register contains the Configuration RAM Page address. During accesses to the Configuration RAM (0800h–08FFh), the ESC drives the CPG[4:0] signals with the value of Bits[4:0] of this register. The CPG[4:0] signals are connected to address pins ADDR[12:8] of the Configuration RAM.

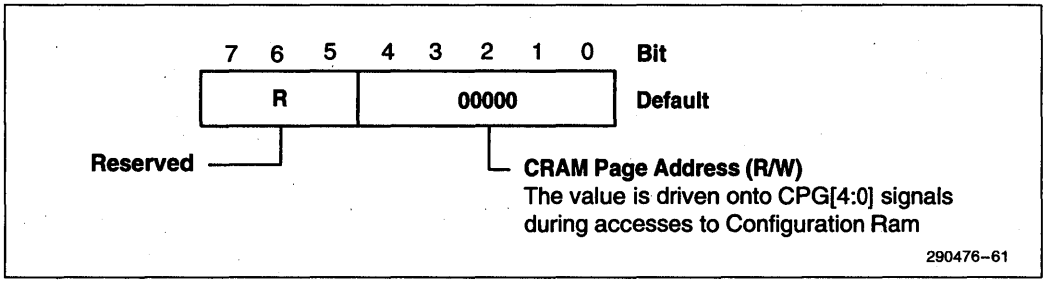


Figure 3-59. Configuration RAM Page Register

Table 3-61. Configuration RAM Page Register

Bit #	Description
7:5	**RESERVED.**
4:0	**CRAM PAGE ADDRESS:** The value of these bits selects a specific page from the Configuration RAM space. The SA[7:0] addresses select the location within this page during I/O accesses to the Configuration RAM.

3.5.2 DIGITAL OUTPUT REGISTER

Register Name: Digital Output
Register Location: 03F2h (Primary), 0372h (Secondary)
Default Value: xxxx0xxxb
Attribute: Write only
Size: 8 bits

This register is used to prevent XBUSOE# from responding to DACK2# during a DMA read accesses to a floppy controller on the ISA bus. If a second floppy (residing on the ISA bus) is using DACK2# in conjunction with a floppy on the X-Bus, this prevents the floppy on the X-Bus and the X-Bus transceiver from responding to an access targeted for the floppy on the ISA bus. This register is also located in the floppy controller device.

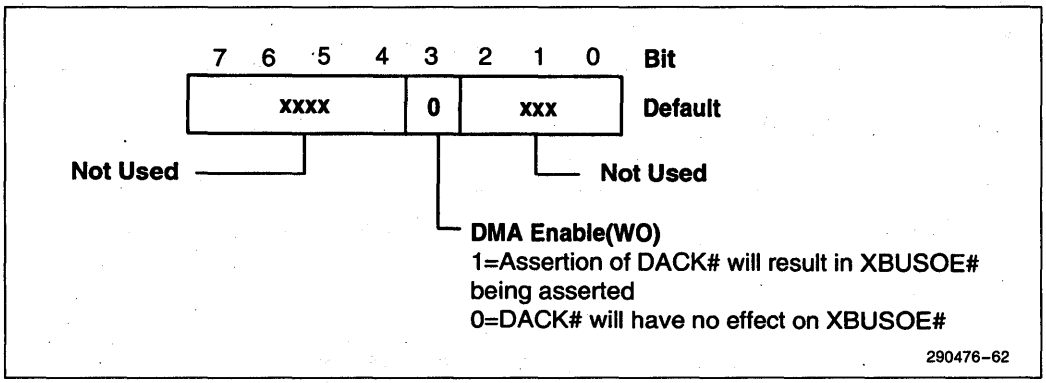


Figure 3-60. Digital Output Register

Table 3-62. Digital Output Register

Bit #	Description
7:4	**NOT USED:** These bits exist in the 82077 FDC. Refer to the 82077 data sheet for further details.
3	**DMA ENABLE:** When this bit is a 1, the assertion of DACK# will result in XBUSOE# being asserted. If this bit is 0, DACK2# has no effect on XBUSOE#. This port bit also exists on the 82077 FDC. This bit defaults to disable (0).
2:0	**NOT USED:** These bits exist in the 82077 FDC. Refer to the 82077 data sheet for further detail.

3.5.3 PORT 92 REGISTER

Register Name: Port 92
Register Location: 92h
Default Value: 00100100b
Attribute: Read/Write
Size: 8 bits

This register is used to support the alternate reset (ALTRST#), alternate A20 (ALTA20), power-on password protection, and fixed disk light function (DLIGHT#). This register is only accessible if Bit 6 in the Peripheral Chip Select Enable B Register is set to "1".

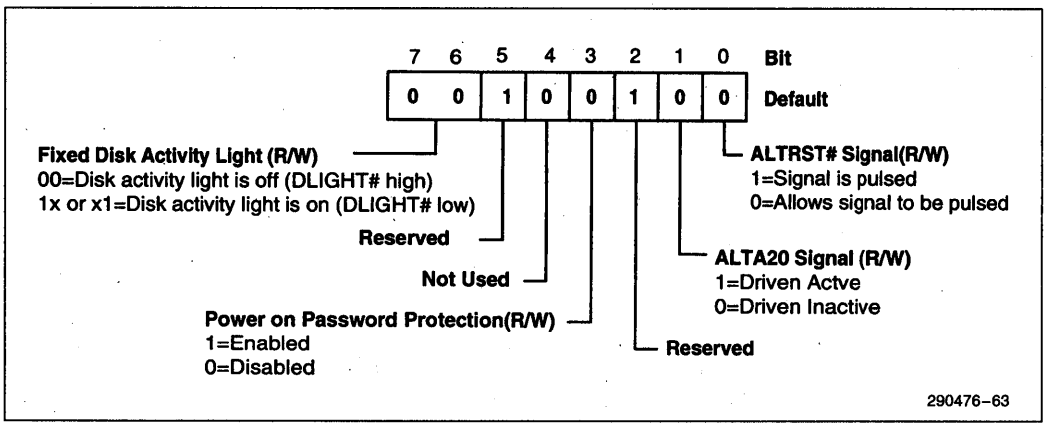


Figure 3-61. Port 92 Register

Table 3-63. Port 92 Register

Bit #	Description
7:6	**FIXED DISK ACTIVITY LIGHT:** These bits are used to turn the Fixed Disk Activity Light on and off. When either of these bits are set to a 1, the light is turned on (DLIGHT# driven active). To turn the light off, both of these bits must be 0.
5	**RESERVED:** This bit is reserved and will always return a 1 when read.
4	**NOT USED:** This bit is not used and will always return a 0 when read.
3	**POWER ON PASSWORD PROTECTION:** A 1 on this bit enables power-on password protection by inhibiting accesses to the RTC memory for RTC addresses (port 70h) from 36h to 3Fh. This is accomplished by not generating RTCRD# and RTCWR# signals for these accesses.
2	**RESERVED:** This bit is reserved and will always return a 1 when read.
1	**ALTA20 SIGNAL:** Writing a 0 to this bit causes the ALTA20 signal to be driven low. Writing a 1 to this bit causes the ALTA20 signal to be driven high.
0	**ALTRST# SIGNAL:** This read/write bit provides an alternate system reset function. This function provides an alternate means to reset the system CPU to effect a mode switch from Protected Virtual Address Mode to the Real Address Mode. This provides a faster means of reset than is provided by the Keyboard controller. This bit is set to a 0 by a system reset. Writing a 1 to this bit will cause the ALTRST# signal to pulse active (low) for approximately 4 SYSCLK's. Before another ALTRST# pulse can be generated, this bit must be written back to a 0.

3.5.4 LAST EISA BUS MASTER GRANTED REGISTER

Register Name: Last EISA Bus Master Granted
Register Location: 0464h
Default Value: xxh
Attribute: Read Only
Size: 8 bits

This register contains information about which EISA bus master most recently had control of the EISA bus. A bit read of 0 indicates that the corresponding slot most recently was granted the bus.

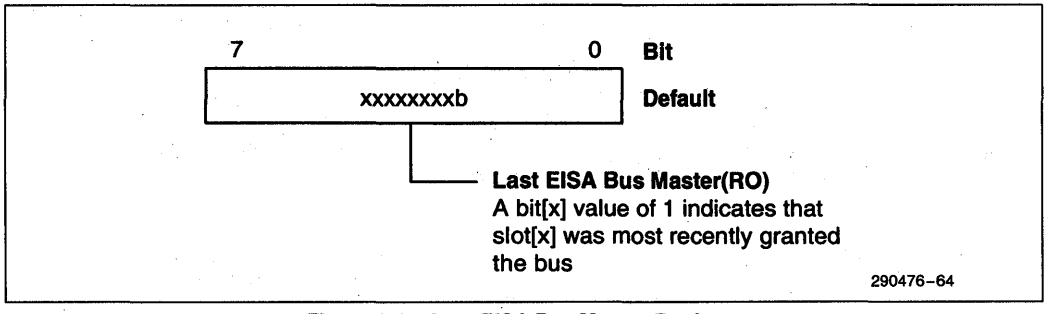


Figure 3-62. Last EISA Bus Master Register

Table 3-64. Last EISA Bus Master Register

Bit #	Description
7:0	**LAST EISA BUS MASTER:** A value of 1 is placed in the bit position of the most recently granted EISA Bus Master.

4.0 ADDRESS DECODING

The ESC contains an address decoder to decode EISA/ISA master cycles. The ESC address decoder uses the address line LA[31:2], and byte enable BE[3:0]# to decode EISA master cycles. For ISA master cycles, the ESC uses address line LA[31:2], SA[1:0], and high byte enable SHBE# for address decode.

The ESC decodes the following set of addresses.

1. BIOS memory space.
2. I/O addresses contained within the ESC.
3. Configuration registers.
4. X-Bus Peripherals

4.1 BIOS Memory Space

The ESC supports a total of 512 Kbytes of BIOS. The ESC will assert the LBIOSCS# signal for memory cycles decoded to be in the BIOS space. The 512 Kbytes of BIOS includes the conventional 128 Kbytes of BIOS and 384 Kbytes of enlarged BIOS.

The 128 Kbytes conventional BIOS memory space is mapped at 1 MB boundary between memory address 000E0000h–000FFFFFh. The 128K conventional BIOS memory space is split into one 64K region, and four 16K regions. These regions are Low BIOS region 1 (000E0000h–000E3FFFh), Low BIOS region 2 (000E4000h–000E7FFFh), Low BIOS region 3 (000E8000h–000EBFFFh), and Low BIOS region 4 (000EC000h–000EFFFFh) and High BIOS region (000F0000h–000FFFFFh). The ESC will assert the LBIOSCS# signal for memory cycles to these regions if the corresponding configuration bits in the BIOS Chip Select A register are set to enable (see Table 4-1).

The conventional BIOS is aliased at multiple memory regions. The aliased memory regions are at 16 MB boundary (High BIOS only), 4 GB minus 1M boundary, and 4 GB boundary. The ESC will assert LBIOSCS# for memory cycles to these aliased regions if the corresponding configuration bits in the BIOS Chip Select B register are also set to enable (see Table 4-1).

The ESC supported VGA BIOS on the motherboard by aliasing the VGA BIOS region to the conventional BIOS region. The VGA BIOS is accessed at memory region 0000C0000h–0000C7FFF. The VGA BIOS region is divided into a Low VGA region (000C0000h–000C3FFFh) and a High VGA region (000C4000h–000C7FFFh). If the BIOS Chip Select B register bit 0 (Low VGA BIOS Enable) and bit 1 (High VGA BIOS Enable) are set to enable, memory accesses to Low VGA BIOS region and High VGA BIOS region will be aliased to conventional Low BIOS region 1 and Low BIOS region 2 respectively and the ESC will assert LBIOSCS#

The ESC supports the 384 Kbytes of enlarged BIOS as specified by the PCI specification. This 384 Kbyte region is mapped in memory space below the 4G aliased conventional BIOS. The enlarged BIOS is accessed between FFF80000h–FFFDFFFFh memory space. If the enlarged BIOS is enabled in the BIOS Enable Chip Select 1 register bit 5 (Enlarged BIOS Enable), the ESC will assert LBIOSCS# signal for accesses to this region.

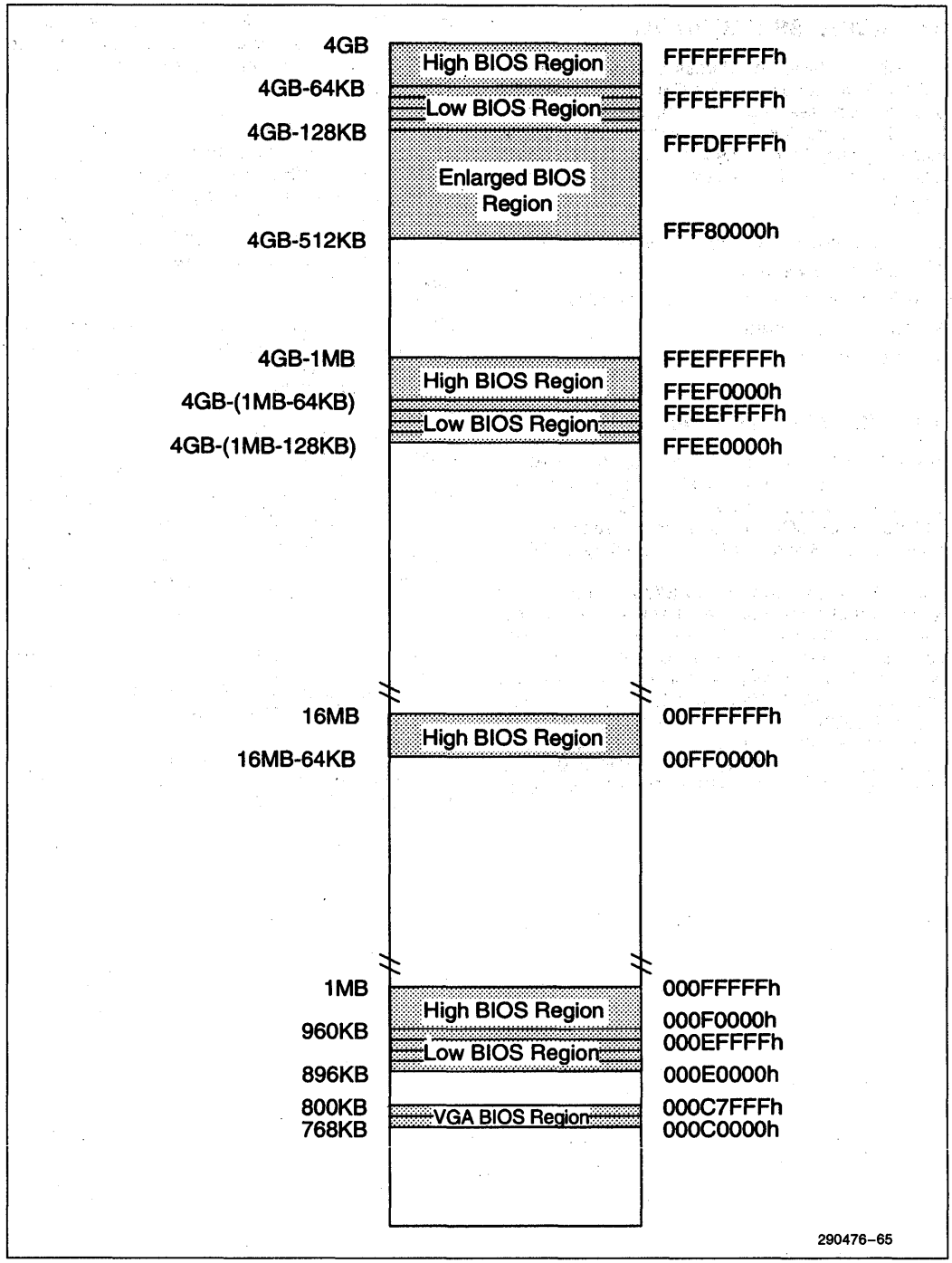


Figure 4-1. BIOS Memory Map

Table 4-1. BIOS Chip Select Enable Table

Memory Address Range	Low BIOS En 1	Low BIOS En 2	Low BIOS En 3	Low BIOS En 4	High BIOS En	ENL BIOS En	Low VGA BIOS En	High VGA BIOS En	16M BIOS En	LBIOSCS# Asserted
000C0000h to 000C3FFFh	x	x	x	x	x	x	**0**	x	x	No
	x	x	x	x	x	x	**1**	x	x	Yes
000C4000h to 000C7FFFh	x	x	x	x	x	x	x	**0**	x	No
	x	x	x	x	x	x	x	**1**	x	Yes
000E0000h to 000E3FFFh	**0**	x	x	x	x	x	x	x	x	No
	1	x	x	x	x	x	x	x	x	Yes
000E4000h to 000E7FFFh	x	**0**	x	x	x	x	x	x	x	No
	x	**1**	x	x	x	x	x	x	x	Yes
000E8000h to 000EBFFFh	x	x	**0**	x	x	x	x	x	x	No
	x	x	**1**	x	x	x	x	x	x	Yes
000EC000h to 000EFFFFh	x	x	x	**0**	x	x	x	x	x	No
	x	x	x	**1**	x	x	x	x	x	Yes
000F0000h to 000FFFFFh (960 KB to 1 MB)	x	x	x	x	**0**	x	x	x	x	No
	x	x	x	x	**1**	x	x	x	x	Yes
00FF0000h to 00FFFFFFh (16 MB–64 KB to 16 MB)	x	x	x	x	**x**	x	x	x	**0**	No
	x	x	x	x	**0**	x	x	x	**1**	No
	x	x	x	x	**1**	x	x	x	**1**	Yes
FFEE0000h to FFEE3FFFh	**0**	x	x	x	x	x	x	x	x	No
	1	x	x	x	x	x	x	x	x	Yes
FFEE4000h to FFEE7FFFh	x	**0**	x	x	x	x	x	x	x	No
	x	**1**	x	x	x	x	x	x	x	Yes
FFEE8000h to FFEEBFFFh	x	x	**0**	x	x	x	x	x	x	No
	x	x	**1**	x	x	x	x	x	x	Yes
FFEEC000h to FFEEFFFFh	x	x	x	**0**	x	x	x	x	x	No
	x	x	x	**1**	x	x	x	x	x	Yes
FFEF0000h to FFEFFFFFh	x	x	x	x	**0**	x	x	x	x	No
	x	x	x	x	**1**	x	x	x	x	Yes
FFF80000h to FFFDFFFFh (4 GB–512 KB to 4G–128 KB)	x	x	x	x	x	**0**	x	x	x	No
	x	x	x	x	x	**1**	x	x	x	Yes
FFFE0000h to FFFE3FFFh	**0**	x	x	x	x	x	x	x	x	No
	1	x	x	x	x	x	x	x	x	Yes
FFFE4000h to FFFE7FFFh	x	**0**	x	x	x	x	x	x	x	No
	x	**1**	x	x	x	x	x	x	x	Yes
FFFE8000h to FFFEBFFFh	x	x	**0**	x	x	x	x	x	x	No
	x	x	**1**	x	x	x	x	x	x	Yes
FFFEC000h to FFFEFFFFh	x	x	x	**0**	x	x	x	x	x	No
	x	x	x	**1**	x	x	x	x	x	Yes
FFFF0000h to FFFFFFFFh	x	x	x	x	**0**	x	x	x	x	No
	x	x	x	x	**1**	x	x	x	x	Yes

NOTES:
1. "x" in the above table represents a don't care condition.
2. All the region control bits for the BIOS space are in the BIOS Chip Select A register and BIOS Chip Select 2 register at configuration offsets 42h and 43h respectively.

4.2 I/O Addresses Contained within the ESC

The ESC integrates functions like DMA, Programmable Interrupt Controller, and Timers. All the compatibility registers associated with these functions are also integrated into the ESC. The ESC also integrates some additional registers like EISA System ID register in order to reduce the overall chip count in the system.

All the registers integrated in the ESC are located in the I/O range. These are 8-bit registers and are accessed through the ESC EISA interface. The ESC internal registers are at fixed I/O locations with the exception of DMA Scatter-Gather registers. The DMA Scatter-Gather registers default to the I/O addresses 0410h to 043Fh upon reset. These registers can be relocated by programming the Scatter-Gather Relocate Base Address register. The DMA Scatter-Gather registers can be relocated to I/O addresses range xx10h–xx3Fh.

Registers at I/O addresses 70h, 372h, and 3F2h are shared registers between ESC and external logic. Port 70h is duplicated in the Real Time Clock logic. Bit 3 of ports 372h and 3F2h reside in the ESC and the other bits reside in the Floppy Disk Controller.

Table 4-2 documents the I/O address to the ESC internal registers.

Table 4-2. ESC I/O Register Address Map

Port	Address				Type	Name	Block
	FEDC	BA98	7654	3210			
0000h	0000	0000	000x	0000	R/W	DMA1 CH0 Base and Current Address	DMA
0001h	0000	0000	000x	0001	R/W	DMA1 CH0 Base and Current Count	DMA
0002h	0000	0000	000x	0010	R/W	DMA1 CH1 Base and Current Address	DMA
0003h	0000	0000	000x	0011	R/W	DMA1 CH1 Base and Current Count	DMA
0004h	0000	0000	000x	0100	R/W	DMA1 CH2 Base and Current Address	DMA
0005h	0000	0000	000x	0101	R/W	DMA1 CH2 Base and Current Count	DMA
0006h	0000	0000	000x	0110	R/W	DMA1 CH3 Base and Current Address	DMA
0007h	0000	0000	000x	0111	R/W	DMA1 CH3 Base and Current Count	DMA
0008h	0000	0000	000x	1000	R/W	DMA1 Status(r) Command(w) Register	DMA
0009h	0000	0000	000x	1001	WO	DMA1 Write Request Register	DMA
000Ah	0000	0000	000x	1010	WO	DMA1 Write Single Mask Bit	DMA
000Bh	0000	0000	000x	1011	WO	DMA1 Write Mode Register	DMA
000Ch	0000	0000	000x	1100	WO	DMA1 Clear Byte Pointer	DMA
000Dh	0000	0000	000x	1101	WO	DMA1 Master Clear	DMA
000Eh	0000	0000	000x	1110	WO	DMA1 Clear Mask Register	DMA
000Fh	0000	0000	000x	1111	R/W	DMA1 Read/Write All Mask Register Bits	DMA
0020h	0000	0000	001x	xx00	R/W	INT 1 Control Register	PIC
0021h	0000	0000	001x	xx01	R/W	INT 1 Mask Register	PIC
0022h	0000	0000	0010	0010	R/W	Configuration Address Index Register	CONF
0023h	0000	0000	0010	0011	R/W	Configuration Data Index Register	CONF
0040h	0000	0000	010x	0000	R/W	Timer 1—Counter 0 System Clock	TC
0041h	0000	0000	010x	0001	R/W	Timer 1—Counter 1 Refresh Request	TC
0042h	0000	0000	010x	0010	R/W	Timer 1 Counter 2 Speaker Tone	TC
0043h	0000	0000	010x	0011	WO	Timer 1 Command Mode Register	TC
0048h	0000	0000	010x	1000	R/W	Timer 2—Counter 0 Fail-Safe Timer	TC

Table 4-2. ESC I/O Register Address Map (Continued)

Port	Address				Type	Name	Block
	FEDC	BA98	7654	3210			
0049h	0000	0000	010x	1001	R/W	Timer 2—Counter 1 Reserved	TC
004Ah	0000	0000	010x	1010	R/W	Timer 2 Counter 2 CPU Speed Control	TC
004Bh	0000	0000	010x	1011	WO	Timer 2 Command Mode Register	TC
0061h	0000	0000	0110	00x1	R/W	NMI Status and Control	Control
0070h*	0000	0000	0111	0xx0	WO	NMI Mask Register	Control
0080h	0000	0000	100x	0000	R/W	DMA Page Register— Reserved	DMA
0081h	0000	0000	100x	0001	R/W	DMA Channel 2 Page Register	DMA
0082h	0000	0000	1000	0010	R/W	DMA Channel 3 Page Register	DMA
0083h	0000	0000	100x	0011	R/W	DMA Channel 1 Page Register	DMA
0084h	0000	0000	100x	0100	R/W	DMA Page Register— Reserved	DMA
0085h	0000	0000	100x	0101	R/W	DMA Page Register— Reserved	DMA
0086h	0000	0000	100x	0110	R/W	DMA Page Register— Reserved	DMA
0087h	0000	0000	100x	0111	R/W	DMA Channel 0 Page Register	DMA
0088h	0000	0000	100x	1000	R/W	DMA Page Register— Reserved	DMA
0089h	0000	0000	100x	1001	R/W	DMA Channel 6 Page Register	DMA
008Ah	0000	0000	100x	1010	R/W	DMA Channel 7 Page Register	DMA
008Bh	0000	0000	100x	1011	R/W	DMA Channel 5 Page Register	DMA
008Ch	0000	0000	100x	1100	R/W	DMA Page Register— Reserved	DMA
008Dh	0000	0000	100x	1101	R/W	DMA Page Register— Reserved	DMA
008Eh	0000	0000	100x	1110	R/W	DMA Page Register— Reserved	DMA
008Fh	0000	0000	100x	1111	R/W	DMA Refresh Page Register	DMA
0092h	0000	0000	1001	0010	R/W	System Control Port	Control
00A0h	0000	0000	101x	xx00	R/W	INT 2 Control Register	PIC
00A1h	0000	0000	101x	xx01	R/W	INT 2 Mask Register	PIC
00C0h	0000	0000	1100	000x	R/W	DMA2 CH0 Base and Current Address	DMA
00C2h	0000	0000	1100	001x	R/W	DMA2 CH0 Base and Current Count	DMA
00C4h	0000	0000	1100	010x	R/W	DMA2 CH1 Base and Current Address	DMA
00C6h	0000	0000	1100	011x	R/W	DMA2 CH1 Base and Current Count	DMA
00C8h	0000	0000	1100	100x	R/W	DMA2 CH2 Base and Current Address	DMA
00CAh	0000	0000	1100	101x	R/W	DMA2 CH2 Base and Current Count	DMA
00CCh	0000	0000	1100	110x	R/W	DMA2 CH3 Base and Current Address	DMA
00CEh	0000	0000	1100	111x	R/W	DMA2 CH3 Base and Current Count	DMA
00D0h	0000	0000	1101	000x	R/W	DMA2 Status(r) Command(w) Register	DMA
00D2h	0000	0000	1101	001x	WO	DMA2 Write Request Register	DMA
00D4h	0000	0000	1101	010x	WO	DMA2 Write Single Mask Bit	DMA
00D6h	0000	0000	1101	011x	WO	DMA2 Write Mode Register	DMA

Table 4-2. ESC I/O Register Address Map (Continued)

Port	Address				Type	Name	Block
	FEDC	BA98	7654	3210			
00D8h	0000	0000	1101	100x	WO	DMA2 Clear Byte Pointer	DMA
00DAh	0000	0000	1101	101x	WO	DMA2 Master Clear	DMA
00DCh	0000	0000	1101	110x	WO	DMA2 Clear Mask Register	DMA
00DEh	0000	0000	1101	111x	R/W	DMA2 Read/Write All Mask Register Bits	DMA
00F0h	0000	0000	1111	0000	WO	Reset IRQ13	IRQ13
0372h &	0000	0011	0111	0010	WO	Secondary Floppy Disk Digital Output Register	FDCCS#
03F2h &	0000	0011	1111	0001	WO	Primary Floppy Disk Digital Output Register	FDCCS#
0400h	0000	0100	0000	0000	R/W	Reserved	DMA
0401h	0000	0100	0000	0001	R/W	DMA1 CH0 Base/Current Count	DMA
0402h	0000	0100	0000	0010	R/W	Reserved	DMA
0403h	0000	0100	0000	0011	R/W	DMA1 CH1 Base/Current Count	DMA
0404h	0000	0100	0000	0100	R/W	Reserved	DMA
0405h	0000	0100	0000	0101	R/W	DMA1 CH2 Base/Current Count	DMA
0406h	0000	0100	0000	0110	R/W	Reserved	DMA
0407h	0000	0100	0000	0111	R/W	DMA1 CH3 Base/Current Count	DMA
0408h	0000	0100	0000	1000	R/W	Reserved	DMA
0409h	0000	0100	0000	1001	R/W	Reserved	DMA
040Ah	0000	0100	0000	1010	R/W	DMA Chaining Mode Status/Interrupt Pending	DMA
040Bh	0000	0100	0000	1011	WO	DMA1 Extended Mode Register	DMA
040Ch	0000	0100	0000	1100	WO	Chaining Buffer Control Register	DMA
040Dh	0000	0100	0000	1101	R/W	Reserved	DMA
040Eh	0000	0100	0000	1110	R/W	Reserved	DMA
040Fh	0000	0100	0000	1111	R/W	Reserved	DMA
0410h	0000	0100	0010	0000	WO	DMA CH0 S-G Command Register	DMA
0411h	0000	0100	0010	0001	WO	DMA CH1 S-G Command Register	DMA
0412h	0000	0100	0010	0010	WO	DMA CH2 S-G Command Regsiter	DMA
0413h	0000	0100	0010	0011	WO	DMA CH3 S-G Command Register	DMA
0415h	0000	0100	0010	0101	WO	DMA CH5 S-G Command Register	DMA
0416h	0000	0100	0010	0110	WO	DMA CH6 S-G Command Register	DMA
0417h	0000	0100	0010	0111	WO	DMA CH7 S-G Command Register	DMA
0418h	0000	0100	0010	1000	WO	DMA CH0 S-G Status Register	DMA
0419h	0000	0100	0010	1001	WO	DMA CH1 S-G Status Register	DMA
041Ah	0000	0100	0010	1010	WO	DMA CH2 S-G Status Register	DMA
041Bh	0000	0100	0010	1011	WO	DMA CH3 S-G Status Register	DMA
041Dh	0000	0100	0010	1101	WO	DMA CH5 S-G Status Register	DMA
041Eh	0000	0100	0010	1110	WO	DMA CH6 S-G Status Register	DMA

Table 4-2. ESC I/O Register Address Map (Continued)

Port	Address				Type	Name	Block
	FEDC	BA98	7654	3210			
041Fh	0000	0100	0010	1111	WO	DMA CH7 S-G Status Register	DMA
0420h	0000	0100	0010	0000	RO	DMA CH0 S-G Descriptor Pointer Register	DMA
0421h	0000	0100	0010	0001	RO	DMA CH0 S-G Descriptor Pointer Register	DMA
0422h	0000	0100	0010	0010	RO	DMA CH0 S-G Descriptor Pointer Register	DMA
0423h	0000	0100	0010	0011	RO	DMA CH0 S-G Descriptor Pointer Register	DMA
0424h	0000	0100	0010	0100	RO	DMA CH1 S-G Descriptor Pointer Register	DMA
0425h	0000	0100	0010	0101	RO	DMA CH1 S-G Descriptor Pointer Register	DMA
0426h	0000	0100	0010	0110	RO	DMA CH1 S-G Descriptor Pointer Register	DMA
0427h	0000	0100	0010	0111	RO	DMA CH1 S-G Descriptor Pointer Register	DMA
0428h	0000	0100	0010	1000	RO	DMA CH2 S-G Descriptor Pointer Register	DMA
0429h	0000	0100	0010	1001	RO	DMA CH2 S-G Descriptor Pointer Register	DMA
042Ah	0000	0100	0010	1010	RO	DMA CH2 S-G Descriptor Pointer Register	DMA
042Bh	0000	0100	0010	1011	RO	DMA CH2 S-G Descriptor Pointer Register	DMA
042Ch	0000	0100	0010	1100	RO	DMA CH3 S-G Descriptor Pointer Register	DMA
042Dh	0000	0100	0010	1101	RO	DMA CH3 S-G Descriptor Pointer Register	DMA
042Eh	0000	0100	0010	1110	RO	DMA CH3 S-G Descriptor Pointer Register	DMA
042Fh	0000	0100	0010	1111	RO	DMA CH3 S-G Descriptor Pointer Register	DMA
0434h	0000	0100	0011	0100	RO	DMA CH5 S-G Descriptor Pointer Register	DMA
0435h	0000	0100	0011	0101	RO	DMA CH5 S-G Descriptor Pointer Register	DMA
0436h	0000	0100	0011	0110	RO	DMA CH5 S-G Descriptor Pointer Register	DMA
0437h	0000	0100	0011	0111	RO	DMA CH5 S-G Descriptor Pointer Register	DMA
0438h	0000	0100	0011	1000	RO	DMA CH6 S-G Descriptor Pointer Register	DMA
0439h	0000	0100	0011	1001	RO	DMA CH6 S-G Descriptor Pointer Register	DMA
043Ah	0000	0100	0011	1010	RO	DMA CH6 S-G Descriptor Pointer Register	DMA
043Bh	0000	0100	0011	1011	RO	DMA CH6 S-G Descriptor Pointer Register	DMA
043Ch	0000	0100	0011	1100	RO	DMA CH7 S-G Descriptor Pointer Register	DMA
043Dh	0000	0100	0011	1101	RO	DMA CH7 S-G Descriptor Pointer Register	DMA
043Eh	0000	0100	0011	1110	RO	DMA CH7 S-G Descriptor Pointer Register	DMA
043Fh	0000	0100	0011	1111	RO	DMA CH7 S-G Descriptor Pointer Register	DMA
0461h	0000	0100	0110	0001	R/W	Extended NMI and Reset Control	Control
0462h	0000	0100	0110	0010	R/W	NMI I/O Interrupt Port	Control
0464h	0000	0100	0110	0100	RO	Last EISA Bus master granted (L)	Control
0480h	0000	0100	1000	0000	R/W	Reserved	DMA
0481h	0000	0100	1000	0001	R/W	DMA CH2 High Page Register	DMA
0482h	0000	0100	1000	0010	R/W	DMA CH3 High Page Register	DMA
0483h	0000	0100	1000	0011	R/W	DMA CH1 High Page Register	DMA

Table 4-2. ESC I/O Register Address Map (Continued)

Port	Address				Type	Name	Block
	FEDC	BA98	7654	3210			
0484h	0000	0100	1000	0100	R/W	Reserved	DMA
0485h	0000	0100	1000	0101	R/W	Reserved	DMA
0486h	0000	0100	1000	0110	R/W	Reserved	DMA
0487h	0000	0100	1000	0111	R/W	DMA CH0 High Page Register	DMA
0488h	0000	0100	1000	1000	R/W	Reserved	DMA
0489h	0000	0100	1000	1001	R/W	DMA CH6 High Page Register	DMA
048Ah	0000	0100	1000	1010	R/W	DMA CH7 High Page Register	DMA
048Bh	0000	0100	1000	1011	R/W	DMA CH5 High Page Register	DMA
048Ch	0000	0100	1000	1110	R/W	Reserved	DMA
048Dh	0000	0100	1000	1101	R/W	Reserved	DMA
048Eh	0000	0100	1000	1110	R/W	Reserved	DMA
048Fh	0000	0100	100x	1111	R/W	DMA Refresh High Page Register	DMA
04C2h	0000	0100	1100	0010	R/W	Reserved	DMA
04C6h	0000	0100	1100	0110	R/W	DMA CH5 High Base and Current Count Register	DMA
04CAh	0000	0100	1100	1010	R/W	DMA CH6 High Base and Current Count Register	DMA
04CEh	0000	0100	1100	1110	R/W	DMA CH7 High Base and Current Count Register	DMA
04D0h	0000	0100	1101	0000	R/W	INT-1 Edge/Level Control Register	PIC
04D1h	0000	0100	1101	0001	R/W	INT-2 Edge/Level Control Register	PIC
04D2h	0000	0100	1101	0010	R/W	Reserved	DMA
04D3h	0000	0100	1101	0011	R/W	Reserved	DMA
04D4h	0000	0100	1101	0100	R/W	DMA2 Chaining Mode	DMA
04D5h	0000	0100	1101	1001	R/W	Reserved	DMA
04D6h	0000	0100	1101	0010	WO	DMA2 Extended Mode Register	DMA
04D7h	0000	0100	1101	0111	R/W	Reserved	DMA
04D8h	0000	0100	1101	1000	R/W	Reserved	DMA
04D9h	0000	0100	1101	1001	R/W	Reserved	DMA
04DAh	0000	0100	1101	1010	R/W	Reserved	DMA
04DBh	0000	0100	1101	1011	R/W	Reserved	DMA
04DCh	0000	0100	1101	1100	R/W	Reserved	DMA
04DDh	0000	0100	1101	1101	R/W	Reserved	DMA
04DEh	0000	0100	1101	1110	R/W	Reserved	DMA
04DFh	0000	0100	1101	1111	R/W	Reserved	DMA
04E0h	0000	0100	1110	0000	R/W	DMA CH0 Stop Register Bits [7:2]	DMA
04E1h	0000	0100	1110	0001	R/W	DMA CH0 Stop Register Bits [15:8]	DMA
04E2h	0000	0100	1110	0010	R/W	DMA CH0 Stop Register Bits [23:16]	DMA
04E3h	0000	0100	1110	0011	R/W	Reserved	DMA

Table 4-2. ESC I/O Register Address Map (Continued)

Port	Address				Type	Name	Block
	FEDC	BA98	7654	3210			
04E4h	0000	0100	1110	0100	R/W	DMA CH1 Stop Register Bits [7:2]	DMA
04E5h	0000	0100	1110	0101	R/W	DMA CH1 Stop Register Bits [15:8]	DMA
04E6h	0000	0100	1110	0110	R/W	DMA CH1 Stop Register Bits [23:16]	DMA
04E7h	0000	0100	1110	0111	R/W	Reserved	DMA
04E8h	0000	0100	1110	1000	R/W	DMA CH2 Stop Register Bits [7:2]	DMA
04E9h	0000	0100	1110	1001	R/W	DMA CH2 Stop Register Bits [15:8]	DMA
04EAh	0000	0100	1110	1010	R/W	DMA CH2 Stop Register Bits [23:16]	DMA
04EBh	0000	0100	1110	1011	R/W	Reserved	DMA
04ECh	0000	0100	1110	1100	R/W	DMA CH3 Stop Register Bits [7:2]	DMA
04EDh	0000	0100	1110	1101	R/W	DMA CH3 Stop Register Bits [15:8]	DMA
04EEh	0000	0100	1110	1110	R/W	DMA CH3 Stop Register Bits [23:16]	DMA
04EFh	0000	0100	1110	1111	R/W	Reserved	DMA
04F0h	0000	0100	1111	0000	R/W	Reserved	DMA
04F1h	0000	0100	1111	0001	R/W	Reserved	DMA
04F2h	0000	0100	1111	0010	R/W	Reserved	DMA
04F3h	0000	0100	1111	0011	R/W	Reserved	DMA
04F4h	0000	0100	1111	0100	R/W	DMA CH5 Stop Register Bits [7:2]	DMA
04F5h	0000	0100	1111	0101	R/W	DMA CH5 Stop Register Bits [15:8]	DMA
04F6h	0000	0100	1111	0110	R/W	DMA CH5 Stop Register Bits [23:16]	DMA
04F7h	0000	0100	1111	0111	R/W	Reserved	DMA
04F8h	0000	0100	1111	1000	R/W	DMA CH6 Stop Register Bits [7:2]	DMA
04F9h	0000	0100	1111	1001	R/W	DMA CH6 Stop Register Bits [15:8]	DMA
04FAh	0000	0100	1111	1010	R/W	DMA CH6 Stop Register Bits [23:16]	DMA
04FBh	0000	0100	1111	1011	R/W	Reserved	DMA
04FC	0000	0100	1111	1100	R/W	DMA CH7 Stop Register Bits [7:2]	DMA
04FDh	0000	0100	1111	1101	R/W	DMA CH7 Stop Register Bits [15:8]	DMA
04FEh	0000	0100	1111	0111	R/W	DMA CH7 Stop Register Bits [23:16]	DMA
04FFh	0000	0100	1111	1111	R/W	Reserved	DMA
0C00h	0000	1100	0000	0000	R/W	Configuration RAM Page Register	Conf
0C80h	0000	1100	100	0000	RO	System Board ID Byte Lane 1 Bits [7:0]	Board ID
0C81h	0000	1100	100	0001	RO	System Board ID Byte Lane 2 Bits [15:8]	Board ID
0C82h	0000	1100	100	0010	RO	System Board ID Byte Lane 3 Bits [23:16]	Board ID
0C83h	0000	1100	1000	0011	RO	System Board ID Byte Lane 4 Bits [31:24]	Board ID

NOTES:
*Port 70h resides in the ESC, in addition the lower 7 bits of Port 70h reside in Real Time Clock.
& Bit 3 of ports 372h and 3F2h reside in the ESC while the other bits reside on the ISA bus.

4.3 Configuration Addresses

ESC configuration registers are accessed through I/O registers 22h and 23h. These I/O registers are used as index address register (22h) and index data register (23h). The index address register is used to write the configuration register address. The data (configuration register address) in register 22h is used to decode a configuration register. The selected configuration register can be read or written to by performing a read or a write operation to the index data register at I/O address 23h.

Table 4-3. Configuration Register Index Address

Configuration Offset	Abbreviation	Register Name
00h–01h		Reserved
02h	ESCID	ESC ID
03h–07h		Reserved
08h	RID	Revision ID
09h–3Fh		Reserved
40h	MS	Mode Select
41h		Reserved
42h	BIOSCSA	BIOS Chip Select A
43h	BIOSCSB	BIOS Chip Select B
44h–4Ch		Reserved
4Dh	CLKDIV	BCLK Clock Divisor
4Eh	PCSA	Peripheral Chip Select A
4Fh	PCSB	Peripheral Chip Select B
50h	EISAID1	EISA ID Byte 1
51h	EISAID2	EISA ID Byte 2
52h	EISAID3	EISA ID Byte 3
53h	EISAID4	EISA ID Byte 4
54h–56h		Reserved
57h	SGRBA	Scatter-Gather Relocate Base Address
58h–59h		Reserved
60h	PIRQRC0	PIRQ0# Route Control
61h	PIRQRC1	PIRQ1# Route Control

Table 4-3. Configuration Register Index Address (Continued)

Configuration Offset	Abbreviation	Register Name
62h	PIRQRC2	PIRQ2# Route Control
63h	PIRQRC3	PIRQ3# Route Control
64h	GPCSLA0	General Purpose Chip Select 0 Base Low Address
65h	GPCSHA0	General Purpose Chip Select 0 Base High Address
66h	GPCSM0	General Purpose Chip Select 0 Mask
67h		Reserved
68h	GPCSLA1	General Purpose Chip Select 1 Base Low Address
69h	GPCSHA1	General Purpose Chip Select 1 Base High Address
6Ah	GPCSM1	General Purpose Chip Select 1 Mask
6Bh		Reserved
6Ch	GPCSLA2	General Purpose Chip Select 2 Base Low Address
6Dh	GPCSHA2	General Purpose Chip Select 2 Base High Address
6Eh	GPCSM2	General Purpose Chip Select 2 Mask
6Fh	GPXBC	General Purpose Peripheral X-Bus Control
70h–87h		Reserved
88h	TSTC	Test Control
89h–9Fh		Reserved

4.4 X-Bus Peripherals

The ESC generates chip selects for certain functions that typically reside on the X-Bus. The ESC asserts the chip selects combinatorially from the LA addresses. The ESC generates chip selects signals for Keyboard Controller, Floppy Disk Controller, IDE, Parallel Port, Serial Port, and General Purpose peripherals. The ESC also generates read and write strobes for Real Time Clock and Configuration RAM. The read and write strobes are a function of LA addresses, the ISA read and write strobes (IORC# and IOWC#), and BCLK. All of the peripherals supported by the ESC are at fixed I/O addresses with the exception of the general purpose peripherals. The ESC support for these peripherals can be enabled or disabled through configuration registers Peripheral Chip Select A and Peripheral Chip Select B. The general purpose peripherals are mapped to I/O addresses by programming a set of configuration registers: General Purpose Chip Select x Base Low Address register, General Purpose Chip Select x Base High Address register, and General Purpose Chip Select x Mask register.

Table 4-4. X-Bus Chip Selects Decode

Port	FEDC	BA98	7654	3210	R/W	Name	Chip Select
0060h	0000	0000	0110	00x0	R/W	Keyboard Controller	KYBDCS#
0064h	0000	0000	0110	01x0	R/W	Keyboard Controller	KYBDCS#
0070h	0000	0000	0111	0xx0	W	Real Time Clock	RTCALE
0071h	0000	0000	0111	0xx1	R/W	Real Time Clock	RTCWR#/RTCRD#
0170h to 0177h	0000	0001	0111	0xxx	R/W	IDE Controller 0-Secondary	ECS[2:0] = 011 (IDECS0#)
01F0h to 01F7h	0000	0001	1111	0xxx	R/W	IDE Controller 0-Primary	ECS[2:0] = 011 (IDECS0#)
0278h to 027Bh	0000	0010	0111	1000 to 1011	R/W	Parallel Port LPT3	ECS[2:0] = 010 (LPTCS#)
02F8h to 02FFh	0000	0010	1111	xxxx	R/W	Serial Port COM2	ECS[2:0] = 00x (COMxCS#)
0370h to 0357h	0000	0011	0111	0000 to 0101	R/W	Floppy Disk Controller-Secondary	FDCCS#
0376h	0000	0011	0111	0111	R/W	IDE Controller 1-Secondary	ECS[2:0] = 100 (IDECS1#)
0377h	0000	0011	0111	0110	R/W	IDE Controller 1-Secondary	ECS[2:0] = 100 (IDECS1#)
0377h	0000	0011	0111	0111	R/W	Floppy Disk Controller-Secondary	FDCCS#
0378h to 037Bh	0000	0011	0111	1000 to 1011	R/W	Parallel Port LPT2	ECS[2:0] = 010 (LPTCS#)
03BCh to 03BFh	0000	0011	1011	11xx	R/W	Parallel Port LPT1	ECS[2:0] = 010 (LPTCS#)
03F0h to 0375h	0000	0011	1111	0000 to 0101	R/W	Floppy Disk Controller-Primary	FDCCS#
03F6h	0000	0011	0111	0110	R/W	IDE Controller 1-Primary	ECS[2:0] = 100 (IDECS1#)
03F7h	0000	0011	0111	0111	R/W	IDE Controller 1-Primary	ECS[2:0] = 100 (IDECS1#)
03F7h	0000	0011	0111	0111	R/W	Floppy Disk Controller-Secondary	FDCCS#
03F8h to 03FFh	0000	0011	1111	1000	R/W	Serial Port COM 1	ECS[2:0] = 00x (COMxCS#)
0800h to 08FFh	0000	1000	xxxx	xxxx	W/R	Configuration RAM	CRAMWR#/CRAMRD#

5.0 EISA CONTROLLER FUNCTIONAL DESCRIPTION

5.1 Overview

The EISA controller in the ESC provides Master/ Slave EISA interface function for the ESC internal resources. In addition the ESC acts as an EISA central resource for the system. As a system central resource, the EISA controller is responsible for generating the translation control signals necessary for bus to bus transfers. This translation includes transfer between devices on EISA Bus and ISA Bus and transfers between different size master device and slave device. The EISA controller generates the control signals for EISA Data Swap Buffers integrated in the PCEB. The ESC EISA interface generates cycles for DMA transfers, and refresh. The ESC internal registers are accessed through the EISA slave interface. The ESC is responsible for supporting the following:

Service EISA Master cycles to:

— EISA slaves devices.

— ISA slave devices.

— ESC internal registers.

Service ISA Master cycles to:

— EISA slave devices.

— ISA (mis-matched) slave devices.

— ESC internal registers.

Service DMA cycles :

— From/to DMA slave on the EISA bus to/from memory on the EISA/ISA bus.

— From/to DMA slave on the ISA bus to/from memory on the EISA/ISA bus.

— From/to DMA slave on the EISA/ISA bus to/ from memory on the PCI bus.

Service REFRESH cycles :

The EISA controller will service the refresh cycle by generating the appropriate address and command signals. These cycles are initiated by either the ESC internal refresh logic or by an external ISA-Bus Master.

Generates DATA SWAP BUFFER control:

The EISA controller generates the control signals for the data bus swap control (assembly/disassembly) and swapping process to support data size mismatches of the devices on the EISA and ISA buses. The actual data steering and swapping is performed by the PCEB.

Generate WAIT STATES:

The wait state generator is responsible for generating the wait states based on the sampling of the EXRDY, CHRDY, NOWS# and the default wait states. The default wait state depends on the cycle type.

5.2 Clock Generation

The ESC generates the EISA Bus clock. The ESC uses a divider circuit to generate the EISA Bus clock. The ESC supports PCI bus frequencies between 25 MHz and 33 MHz. The PCI clock is divided by 3 or 4 by the clock generation logic in the ESC. The EISA Clock Divisor register bits [2:0] select the divide value.

The ESC provides the EISA Bus clock as the BCLKOUT output. Although the ESC is capable of driving 240 pF load on the BCLKOUT pin, it is recommended that this signal be buffered to product the EISA BCLK signal.

The ESC EISA control logic and EISA interface is synchronous to the BCLK input. A maximum delay of 15 ns is allowed between the BCLKOUT output and the BCLK input for proper device functionality.

Table 5-1. PCICLK and BCLK Frequency Relationship

PCICLK (MHz)	DIVISOR (Programmable)	BCLK (MHz)
25	3	8.33
30	4*	7.5
33.3	4*	8.33

NOTE:
*The ESC wakes up after reset with a default divisor value of 4.

5.2.1 CLOCK STRETCHING

The ESC is capable of stretching EISA Bus clock (BCLKOUT) for PCEB generated EISA cycles. The ESC stretches the EISA Bus clock (BCLKOUT) in order to minimize the synchronization penalty between PCI clock and EISA clock for accesses to EISA Bus by PCI agents. The PCEB initiates an EISA cycle by asserting START# synchronous to PCICLK. The ESC ensures the START# mimimum pulse width is met by stretching the EISA Bus clock low time.

The ESC samples START# on every PCICLK when the PCEB has the EISA Bus. After sampling START# asserted, the ESC delays the rising edge of BCLKOUT until the START# has met the 115 ns minimum pulse width specification.

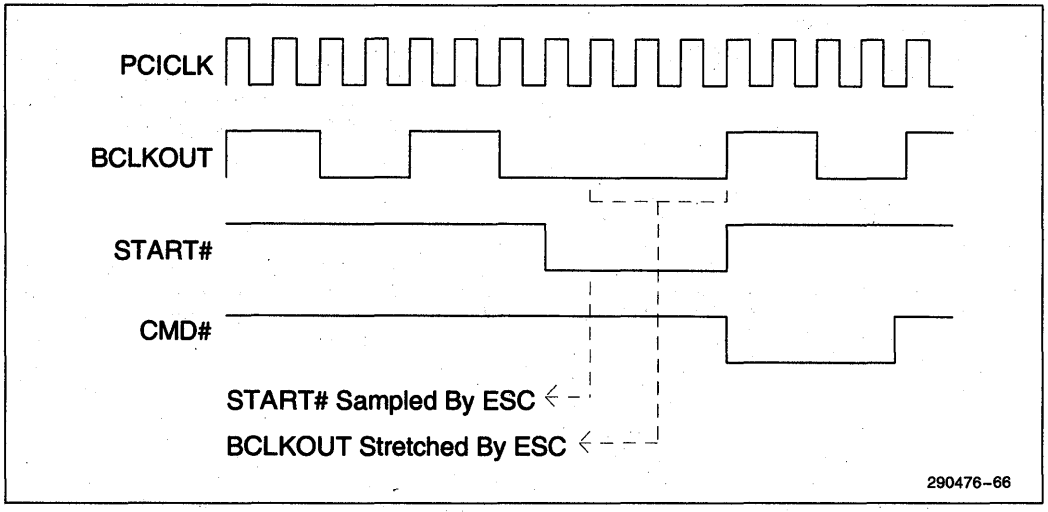


Figure 5-1. BCLK Stretching

5.3 EISA Master Cycles

EISA Master cycles are initiated on the EISA bus by an EISA Master (including PCEB for PCI agents). These cycles are accesses to the following resources:

— EISA slaves devices (including PCEB for PCI agents).

— ISA slave devices.

— ESC internal registers (8-bit EISA Slave).

An EISA master gains control of the bus by asserting MREQx# (PEREQ# in case of PCEB) to the ESC. The ESC after performing the necessary arbitration asserts the corresponding MACKn# (negates EISAHOLD in case of the PCEB). Refer to Chapter 7.0 for arbitration protocol.

In response to receiving the acknowledge signal, the EISA Master starts the cycle by driving the bus with LA[31:02], BE[3:0], W/R, and M/IO. The EISA Master then asserts START# to indicate the beginning of the current cycle. A 16-bit EISA Master will also assert MASTER16# at this time. The ESC generates SBEH#, S1, and S0 signals from the BE[3:0]# signals.

5.3.1 EISA MASTER TO 32 EISA SLAVE

An EISA slave after decoding its address asserts EX32# or EX16#. The EISA master and the ESC use these signals to determine the EISA slave data size. The 32-bit or 16-bit EISA master continues with the cycles if EX32# or EX16# is asserted respectively. The ESC acts as a central resources for the EISA master and generates CMD# for the cycles. The ESC asserts CMD# on the same BCLK edge that START# is negated. The ESC monitors the EXRDY signal on the EISA bus to determine when to negate the CMD#. An EISA Slave can extend the cycle by negating EXRDY. EISA specifications require that EXRDY not be held negated for more than 2.5 μs. A burstable EISA slave asserts SLBURST# signal the same time the slave decodes its address. The EISA master will sample SLBURST# and assert MSBURST# if it is capable of bursting. The ESC keeps the CMD# asserted during a burst EISA transfer. The ESC deasserts CMD# to indicate the end of the burst transfer after the EISA master deasserts MSBURST#.

If EX16# is asserted, a 32-bit EISA master backs-off the bus by floating BE[3:0]# and START# (see Section 5.3.4). The ESC acts as a central resources for the EISA master in this case and takes over the mastership of the EISA bus by deriving START#, CMD#, and the appropriate byte enables. The ESC generates the necessary translation cycles for the EISA master and returns the bus ownership to the master by asserting EX32# and EX16#. The ESC monitors the EXRDY signal on the EISA bus to determine when to negate the CMD#. An EISA Slave can extend the cycle by negating EXRDY. EISA specification require that EXRDY not be held negated for more than 2.5 μs. A burstable EISA slave will assert SLBURST# signal the same time when its address is decoded. The EISA master will sample SLBURST# and assert MSBURST# if it is capable of bursting. The ESC keeps the CMD# asserted during a burst EISA transfer. The ESC deasserts CMD# to indicate the end of the burst transfer after the EISA master deasserts MSBURST#.

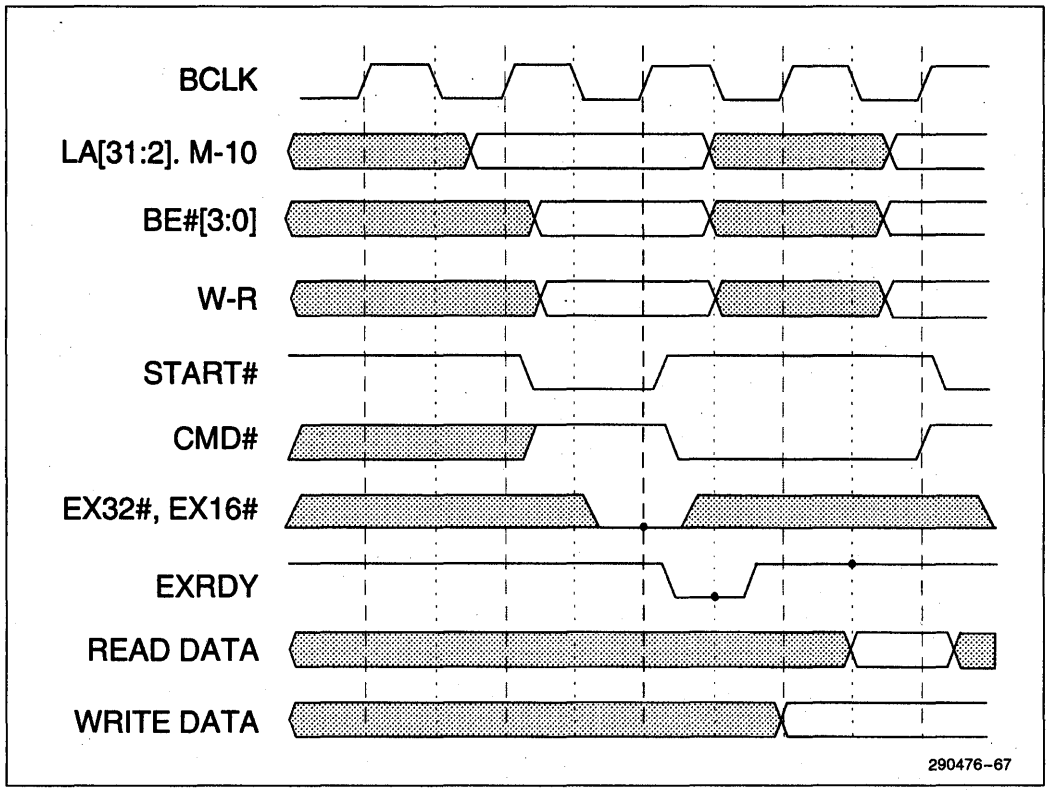


Figure 5-2. Standard EISA Master to EISA Slave Cycle

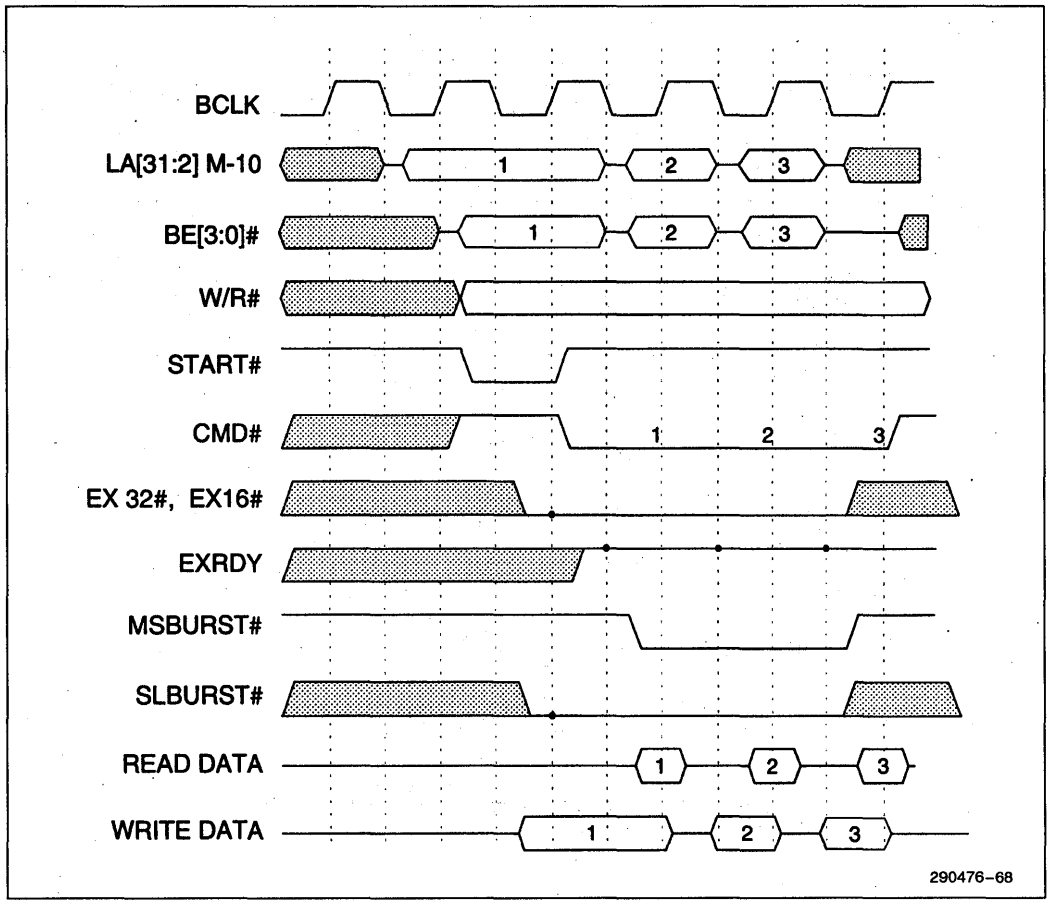


Figure 5-3. Burst EISA Master to EISA Slave Cycle

5.3.2 EISA MASTER TO 16-BIT ISA SLAVE

An ISA slave after decoding its address asserts M16# or IO16#. The ESC monitors the EX32#, EX16#, M16#, and IO16# signals to determine the slave type. If EX32# and EX16# are negated and M16# or IO16# is asserted, the ESC performs ISA translation cycles for the EISA Bus master by generating BALE, MRDC#, MWRC#, IORC#, IOWC# signals as appropriate. The ISA slave can add wait states by negating CHRDY. The ESC samples CHRDY and translate it into EXRDY.

5.3.3 EISA MASTER TO 8-BIT EISA/ISA SLAVES

An 8-bit slave does not positively acknowledge its selection by asserting any signal. The absence of an asserted EX32#, EX16#, M16#, and IO16# indicate to the ESC that an 8-bit device has been selected. The EISA master is backed-off the bus, and the ESC takes over mastership of the EISA/ISA bus. The ESC will run 8-bit translation cycles on the bus by deriving the EISA control signals and the ISA control signals. A slave can extend the cycles by negating EXRDY or CHRDY signals.

The ESC (Internal Registers) is accessed as an 8-bit slave.

5.3.4 EISA MASTER BACK-OFF

During EISA master transfer where the master and slave size is mis-matched, the EISA master is required to back-off the bus on the first falling edge of BCLK after START# is negated. The EISA master floats its START#, BE[3:0]#, and data lines at this time. This allows the ESC to perform a translation cycle. The master must back-off the bus if a master/slave data size mis-match is determined, regardless if data size translations is performed. At the end of the data size translation or transfer cycle, control is transferred back to the bus master by the ESC by driving EX32# and EX16# active on the falling edge of BCLK, before the rising edge of BCLK that the last CMD# is negated. An additional BCLK is added at the end of the transfer to allow the exchanging of cycle control to occur.

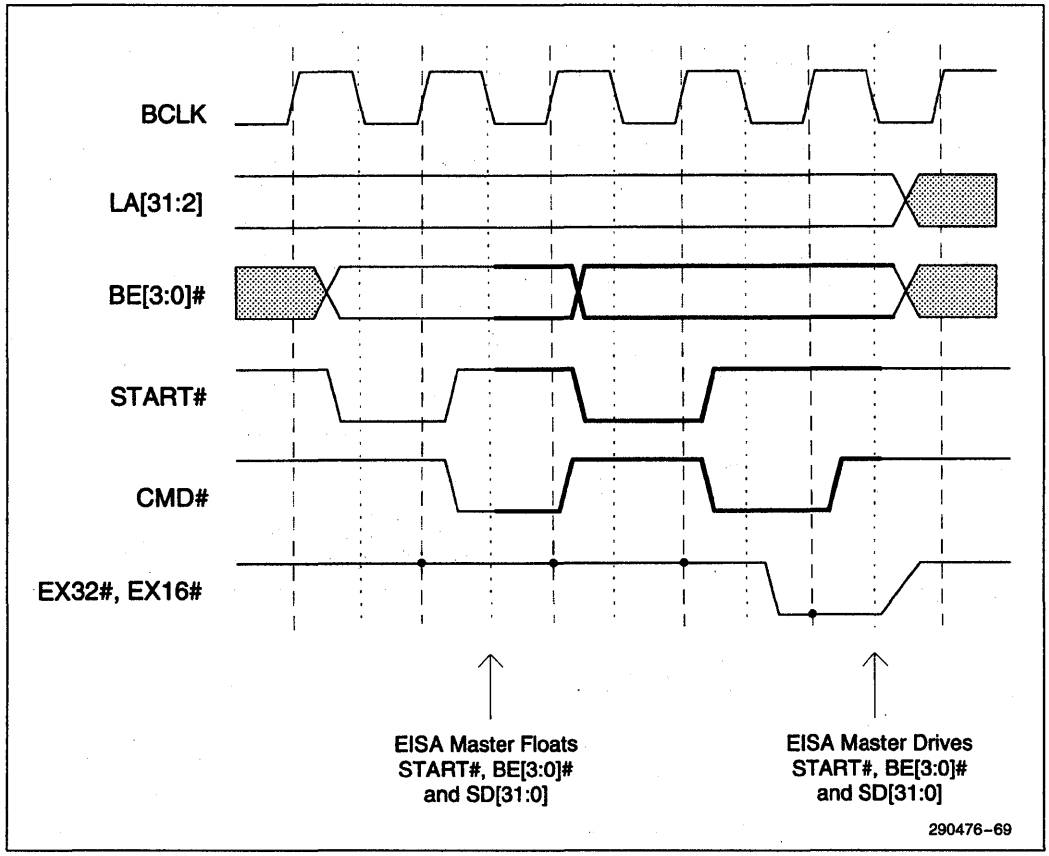


Figure 5-4. EISA Master Back-Off Cycle

5.4 ISA Master Cycles

ISA cycles are initiated on the ISA bus by an ISA master. These cycles are accesses to the following system resources:

— EISA slaves devices (including PCEB for PCI agents).

— ISA slave devices.

— ESC internal registers (8-bit EISA Slave).

The ISA Master initiates such a cycle by asserting the DREQx# line to the ESC. The ESC after performing the necessary arbitration asserts the corresponding DACKx# line. Upon receiving an acknowledge from the ESC, the ISA master asserts the MASTER16# signal line to indicate that it has control of the ISA bus and a cycle on the ISA bus will take place.

The ESC translates the ISA address signals SBHE#, SA1, and SA0 to EISA byte enables BE[3:0]#.

5.4.1 ISA MASTER TO 32-BIT/16-BIT EISA SLAVE

An EISA slave will decode the address to determine if it has been selected. In response to a positive decode, the EISA slave will assert EX32# or EX16#.

The ESC samples these signals to determine if an EISA Slave has been selected. If these signals are asserted, the ESC will perform ISA to EISA cycle translation by driving the EISA control signals.

The ISA Master asserts one of the ISA command signals MRDC#, MWTC#, IORC# or IOWC# depending on whether or not the access is to a memory, an I/O device or an I/O register. The ISA command signals will remain active until the end of the cycle. The ESC will generate the EISA translation by generating the EISA control signals; START#, CMD#, M/IO#, and W/R#.

The EISA slave can add wait states by negating EXRDY. The ESC samples EXRDY and translates it into CHRDY. The ESC will also generate the control signals to steer the data to the appropriate byte lanes for mis-matched cycles.

5.4.2 ISA MASTER TO 16-BIT ISA SLAVE

An ISA Master initiates cycles to ISA slave devices. These cycles are either memory read/write or I/O read/write. The ISA bus Master is assumed to be 16-bit device, and it can access either 8-bit or 16-bit slave devices that reside on the ISA-bus. A 16-bit ISA slave device will respond to a valid address by asserting M16# for memory cycles and IO16# for I/O cycles.

The ESC is inactive during ISA Master cycles where either M16# or IO16# is sampled asserted.

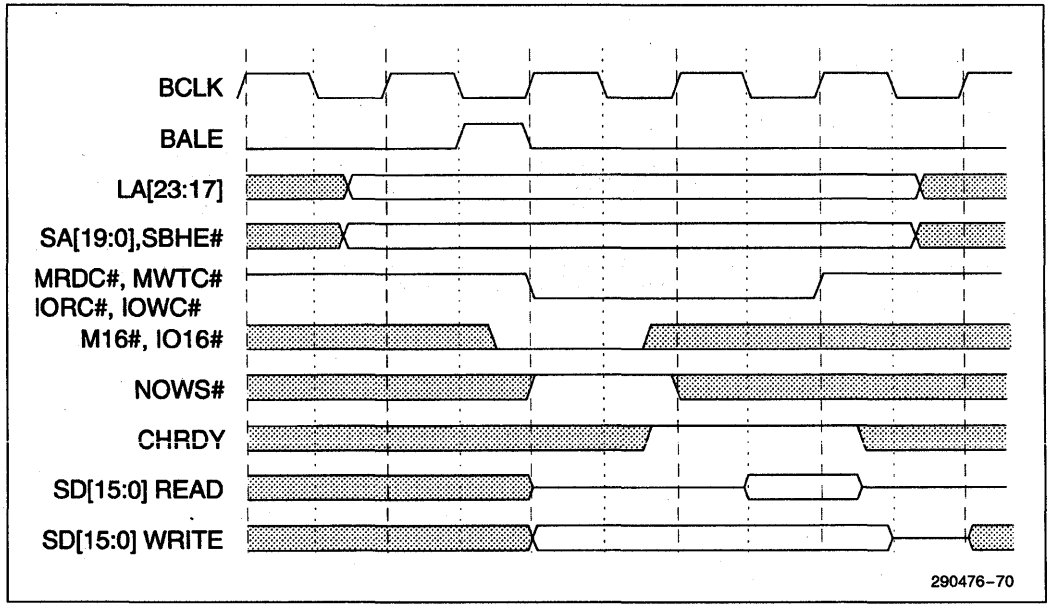


Figure 5-5. ISA Master to 16-Bit ISA Slave Cycles (3 BCLKs)

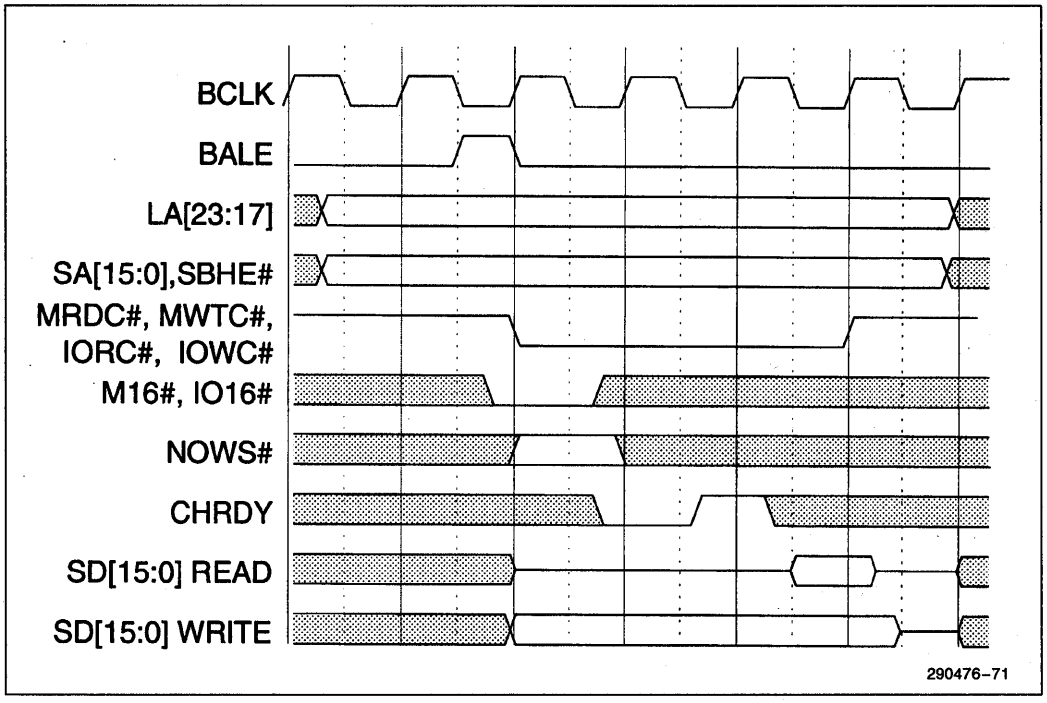


Figure 5-6. ISA Master to 16-Bit ISA Slave Extended Cycle (4 BCLKs)

5.4.3 ISA MASTER TO 8-BIT EISA/ISA SLAVE

An 8-bit slave does not positively acknowledge its selection by asserting any signal. The absence of an asserted EX32#, EX16#, M16#, and IO16# indicate to the ESC that an 8-bit device has been selected. The EISA master is backed-off the bus, and the ESC takes over mastership of the EISA/ISA bus. The ESC will run 8-bit translation cycles on the bus by deriving the EISA control signals and the ISA control signals. A slave can extend the cycles by negating EXRDY or CHRDY signals.

The ESC (Internal Registers) is accessed as an 8-bit slave.

5.4.4 ISA WAIT STATE GENERATION

There are three sources that can affect the generation of wait states for ISA cycles. The first is the default wait states, which determines the standard or default ISA bus cycle in the absence of any response from the slave. The second is cycle extension, which is indicated by the slave pulling the CHRDY signal line inactive (low). The CHRDY is high by default due to a pullup resistor. Thus, the cycle will be extended until the CHRDY is returned to its active high value. The third way to change the number of wait states is when the slave asserts the NOWS# signal which makes the cycle shorter than the default or standard cycle.

ISA Memory slaves (8 bits and 16 bits) and ISA I/O slaves (only 8 bits) can shorten their default cycles by asserting the NOWS# signal lines. A 16-bit I/O slave cannot shorten its default cycles.

When NOWS# is asserted at the same time the CHRDY is negated by the ISA slave device, NOWS# will be ignored and wait states will be added. (i.e.; CHRDY has precedence over NOWS#.)

DMA devices (I/O) cannot add wait states, but memory can.

Table 5-2 shows the number of BCLKs for each cycle type (Memory, I/O DMA), default, wait states added and with NOWS# asserted.

Table 5-2. Number of BCLKs for ISA Master Cycles

Cycle Type	Bus Size	No Wait State NOWS# = 0	Standard CHRDY = 1 NOWS# = 1	One Wait State CHRDY = 0
		Number of BCLKs		
Memory Read/Write	16	2	3	4
Memory Read/Write	8	4, 5	6	7
I/O Read/Write	16	3	3	4
I/O Read/Write	8	4, 5	6	7
DMA Compatible	8/16	8	8	10
DMA Type A*	8/16	NA	6	7
DMA Type B*	8/16	NA	4	5
DMA Type C†	8/16	NA	2	3

NOTES:
* If ISA memory responds, the ESC will extend the cycle by 1 BCLK.
† If ISA memory responds, the ESC will use DMA Type B read cycle timing.

5.5 Mis-Match Cycles

Data size translation is performed by the ESC for all mis-matched cycles. A mis-matched cycle is defined as a cycle in which the bus master and bus slave do not have equal data bus sizes (e.g., a 32-bit EISA master accessing a 16-bit ISA slave). The data size translation is performed in conjunction with the PCEB. The ESC generates the appropriate cycles and data steering control signals for mis-matched cycles. The PCEB uses the data steering control signals from the ESC to latch and redirect the data to the appropriate byte lanes. The ESC will perform one or more of the following operations depending on the master and slave type, transfer direction, and the number of byte enables active.

Table 5-3. Mis-Match Master Slave Combinations

Master Type		Slave Type			
		32-Bit EISA	16-Bit EISA	16-Bit ISA	8-Bit EISA/ISA
32-Bit EISA with 16-Bit Downshift	Standard Burst	Match Match	Mis-Match Match	Mis-Match NA	Mis-Match NA
32-Bit EISA	Standard Burst	Match Match	Mis-Match NA	Mis-Match NA	Mis-Match NA
16-Bit EISA	Standard Burst	Mis-Match Mis-Match	Match Match	Mis-Match NA	Mis-Match NA

NOTE:
NA: Not Applicable. The cycle will never occur.

5.6 Data Swap Buffer Control Logic

For all mis-matched cycles, the ESC is responsible for performing data size translations. The ESC performs these data size translations by either becoming the master of the EISA/ISA Bus (see Section 5.3.4) or by directing the flow of data to the appropriate byte lanes. In both cases, the ESC generates Data Swap Buffer control signals to perform data size translation.

— SDCPYEN[13, 3:1]
— SDCPYUP
— SDOE[2:0]#
— SDLE[3:0]#

The Data Swap Buffers are integrated in the PCEB (see PCEB data sheet Chapter 8.0 for Data Swap Buffer function description).

The data size translation cycles consist of one or combinations of Assembly, Disassembly, Copy Up/Down, and Redrive.

Assembly

This occurs during reads when an EISA master data size is greater than the slave data size. ISA masters are required to perform assemble when accessing 8-bit slaves. Assembly consists of two, three, or four cycles depending on the master data size, slave data size, and number of active byte enables. During the assembly process, the data is latched into the PCEB data latch/buffers. This data is driven or redriven on to the EISA bus during the last cycle. The master after initiating the cycle backs-off the bus (see the EISA master back-off Section for details) when a mis-matched is detected. The ESC becomes the bus master and runs the appropriate number of cycles. At the end of the last cycle, the ESC transfers the control of bus back to the original master.

Disassembly

This occurs during writes when the EISA master data size is greater than the slave data size. ISA masters are required to perform disassemble when accessing 8-bit slaves. Disassembly consists of two, three, or four cycles depending on the master data size, slave data size, and number of active byte enables. During the disassembly process, the data is latched in the PCEB latch/buffers on the first cycle. This data is driven or redriven on to the EISA bus on subsequent cycles. The master after initiating the cycle backs-off the bus (see the EISA master back-off Section for details) when a mis-matched is detected. The ESC becomes the bus master and runs the appropriate number of cycles. At the end of the last cycle, the ESC transfers the control of bus back to the original master.

Copy-Up

This occurs during reads when the master data size is greater than the slave data size and during writes when the master data size is smaller than the slave data size. The copy-up function is used for cycles with and without assembly/disassembly.

Copy-Down

This occurs during writes when the master data size is greater than the slave data size and during reads when the master data size is smaller than the slave data size. The copy-down function is used for cycles with and without assembly/disassembly.

Re-Drive

This occurs during reads and writes when both the master and slave are on the EISA/ISA bus and the PCEB is neither a master nor a slave. The re-drive function is always performed in conjunction with assembly/disassembly. During the assembly process, the last cycle is a re-drive cycle. During disassembly, all the cycles except the first cycle are re-drive cycles.

5.7 Servicing DMA Cycles

The ESC is responsible for performing DMA transfers. If the memory is determined (EX32# or EX16# asserted) to be on the EISA bus, the DMA cycle can be "A", "B", or "C" type. If the memory is determined to be on the ISA bus, then the DMA cycle will run as a compatible cycle. The DMA transfers are described in detail in Section 8.0.

5.8 Refresh Cycles

The ESC support refresh cycles on the EISA/ISA bus. The ESC asserts the REFRESH# signal to indicate when a refresh cycle is in progress. Refresh cycles are generated by two sources: the refresh unit inside the ESC or an external ISA bus masters. The EISA bus controller will enable the address lines LA[15:2] and the BE[3:0]#. The High and Low Page register contents will also be placed on the LA[31:16] bus during refresh. Memory slaves on the EISA/ISA bus must not drive any data onto the data bus during the refresh cycle. Slow memory slaves on the EISA/ISA may extend the refresh cycle by negating the EXRDY or CHRDY signal respectively. The refresh cycles are also described in Section 6.11.

5.9 EISA Slot Support

The ESC support up of 8 EISA slots. The ESC provides support for the 8 slots as follows:

— The ESC address and data output buffers directly drive 240 pF capacitive load on the Bus.

— The ESC generates slot specific AENx signals.

— The ESC supports EISA masters in all 8 slots.

The ESC generates encoded AENs and encoded Master Acknowledge signals for 8 slots and 8 masters. These signals must be decoded on the system board to generate the slot specific AENx signals and MACKx# signals. The ESC can be programmed through Mode Select register bit[1:0] to directly generate these signals for 4 slots and 4 masters.

5.9.1 AEN GENERATION

The ESC directly generates the slot specific AEN signals if the ESC is configured to support 4 AENx.

Table 5-4. AEN Generation

CYCLE	A15:A12	A11:A8	A7:A4	A3:A0	AEN4	AEN3	AEN2	AEN1
DMA	xxxx	xxxx	xxxx	xxxx	1	1	1	1
I/O	0000	xx00	xxxx	xxxx	1	1	1	1
I/O	0001	xx00	xxxx	xxxx	1	1	1	0
I/O	0010	xx00	xxxx	xxxx	1	1	0	1
I/O	0011	xx00	xxxx	xxxx	1	0	1	1
I/O	0100	xx00	xxxx	xxxx	0	1	1	1
I/O	0101 to 1111	xx00	xxxx	xxxx	1	1	1	1
I/O	xxxx	xx01	xxxx	xxxx	0	0	0	0
I/O	xxxx	xx10	xxxx	xxxx	0	0	0	0
I/O	xxxx	xx11	xxxx	xxxx	0	0	0	0
MEM	xxxx	xxxx	xxxx	xxxx	0	0	0	0

If the ESC is programmed to support more than 4 EISA AENx, the ESC will generate Encoded AEN signals. Discrete logic like a F138 is required to generate the slot specific AENs.

Table 5-5. Encoded AEN (AEN) Generation

CYCLE	A15:A12	A11:A8	A7:A4	A3:A0	EAEN4	EAEN3	EAEN2	EAEN1
DMA	xxxx	xxxx	xxxx	xxxx	1	1	1	1
I/O	0000	xx00	xxxx	xxxx	1	1	1	1
I/O	0001	xx00	xxxx	xxxx	0	0	0	1
I/O	0010	xx00	xxxx	xxxx	0	0	1	0
I/O	0011	xx00	xxxx	xxxx	0	0	1	1
I/O	0100	xx00	xxxx	xxxx	0	1	0	0
I/O	0101	xx00	xxxx	xxxx	0	1	0	1
I/O	0110	xx00	xxxx	xxxx	0	1	1	0
I/O	0111	xx00	xxxx	xxxx	0	1	1	1
I/O	1000	xx00	xxxx	xxxx	1	0	0	0
I/O	1001 to 1111	xx00	xxxx	xxxx	1	1	1	1
I/O	xxxx	xx01	xxxx	xxxx	0	0	0	0
I/O	xxxx	xx10	xxxx	xxxx	0	0	0	0
I/O	xxxx	xx11	xxxx	xxxx	0	0	0	0
MEM	xxxx	xxxx	xxxx	xxxx	0	0	0	0

NOTE:
EAEN[4:1] combinations not specified in the table are Reserved.

5.9.2 MACKx# GENERATION

The ESC generates the EISA Master Acknowledge signals if the ESC is configured to directly support 4 masters through the Mode Select Register bit[1:0]. In this case the ESC generates MACKx#s for Master 0–3.

If the ESC is programmed to support more than 4 EISA slots, the ESC will generate Encoded (E)MACKx#s. Discrete logic like a F138 is required to generate the MACKx#s for the Masters.

Table 5-6. Encoded MACK# (EMACKx#) Generation

EMACK[4:1]	MACK7#	MACK6#	MACK5#	MACK4#	MACK3#	MACK2#	MACK1#	MACK0#
0000	1	1	1	1	1	1	1	0
0001	1	1	1	1	1	1	0	1
0010	1	1	1	1	1	0	1	1
0011	1	1	1	1	0	1	1	1
0100	1	1	1	0	1	1	1	1
0101	1	1	0	1	1	1	1	1
0110	1	0	1	1	1	1	1	1
0111	0	1	1	1	1	1	1	1
1111	1	1	1	1	1	1	1	1

NOTE:
EMACK[4:1] combinations 1000–1110 are Reserved.

6.0 DMA CONTROLLER

6.1 DMA Controller Overview

The DMA circuitry incorporates the functionality of two 82C37 DMA controllers with seven independently programmable channels, (Channels 0–3 and Channels 5–7). DMA Channel 4 is used to cascade the two controllers together and will default to cascade mode in the Mode register. In addition to accepting requests from DMA slaves, the DMA also responds to requests that are initiated by software. Software may initiate a DMA service request by setting any DMA Channel Request register bit to a 1. The DMA controller for Channels 0–3 is referred to as "DMA-1" and the controller for Channels 4–7 is "DMA-2".

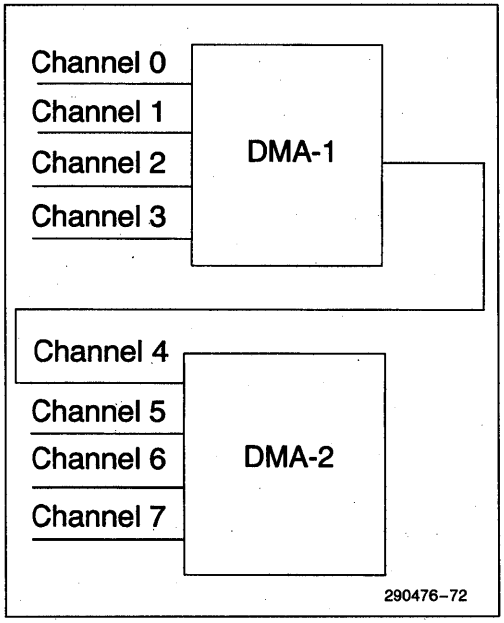


Figure 6-1. Internal DMA Controller

Each DMA channel can be programmed for 8-bit or 16-bit DMA device size. Each channel can also be programmed for compatibility, Type "A", Type "B", or Type "C" (burst transfer) timings. Each DMA channel defaults to the PC-AT compatible settings for DMA device size: channels [3:0] default to 8-bit, count-by-bytes transfers, while channels [7:5] default to 16-bit, count-by-words (address shifted) transfers. The ESC provides the timing control and data size translation necessary for DMA transfers between EISA/ISA agents of mismatched bus sizes.

The DMA Controller supports full 32-bit addressing. Each channel includes a 16-bit ISA compatible Current register which holds the 16 least-significant bits of the 32-bit address, and an ISA compatible Low Page register which contains the eight second most significant bits. An additional High Page register contains the eight most significant bits of the 32-bit address. The address counter can be programmed as either 16-bit compatible address counter or a full 32-bit address counter.

The channels can also be programmed for any of four transfer modes. The transfer modes include single, block, demand, or cascade. Each of the three active transfer modes (single, block, and demand), can perform three different types of transfers (read, write, or verify).

The DMA Controller also features refresh address generation, and auto-initialization following a DMA termination. EISA compatible buffer chaining is included as well as Stop registers to support ring buffer structures.

Scatter-Gather reduces CPU overhead by eliminating reprogramming of the DMA and IO between buffers as well as reducing the number of interrupts.

The DMA Controller includes the EISA Bus arbiter which works with the PCEB's PCI bus arbiter. The arbiter determines which requester from among the requesting DMA slaves, EISA bus masters, the PCI bus, or Refresh should have the bus.

The DMA Controller is at any time either in master mode or slave mode. In master mode, the DMA controller is either servicing a DMA slave's request for DMA cycles, allowing an ISA master to use the bus via a cascaded DREQ signal, or granting the bus to an EISA master via MREQ#/MACK#. In slave mode, the ESC monitors both the EISA bus decoding and responding to I/O read and write commands that address its registers.

When the DMA is in master mode and servicing a DMA slave, it works in conjunction with the ESC EISA bus controller to create bus cycles on the EISA bus. The DMA places addresses onto the internal address bus and the bus controller informs the DMA when to place a new address on the internal bus.

6.2 DMA Transfer Modes

The channels can be programmed for any of four transfer modes. The transfer modes include single, block, demand, or cascade. Each of the three active transfer modes (single, block, and demand), can perform three different types of transfers (read, write, or verify). The ESC does not support memory to memory transfers.

6.2.1 SINGLE TRANSFER MODE

In Single Transfer mode the DMA is programmed to make one transfer only. The byte/word count will be decremented and the address decremented or incremented following each transfer. When the byte/word count "rolls over" from zero to FFFFFFh, or an external EOP is encountered, a Terminal Count (TC) will load a new buffer via Scatter-Gather, buffer chaining or autoinitialize if it is programmed to do so.

DREQ must be held active until DACK becomes active in order to be recognized. If DREQ is held active throughout the single transfer, the bus will be released to the CPU after a single transfer. With the DREQ asserted high, the DMA I/O device will rearbitrate for the bus. Upon winning the bus, another single transfer will be performed. This allows other bus masters a chance to arbitrate for, win, and execute cycles on the EISA Bus.

6.2.2 BLOCK TRANSFER MODE

In Block Transfer mode the DMA is activated by DREQ to continue making transfers during the service until a TC, caused by either a byte/word count going to FFFFFFh or an external EOP, is encountered. DREQ need only be held active until DACK becomes active. If the channel has been programmed for it, a new buffer will be loaded by Scatter-Gather, buffer chaining or Auto-initialization at the end of the service. In this mode, it is possible to lock out other devices for a period of time (including refresh) if the transfer count is programmed to a large number and Compatible timing is selected. Block mode can effectively be used with Type "A", Type "B", or Burst timing since the channel can be interrupted through the 4 μs timeout mechanism, and other devices (or Refresh) can arbitrate for and win the bus. See Chapter 7.0 on the EISA Bus Arbitration for a detailed description of the 4 μs timeout mechanism.

6.2.3 DEMAND TRANSFER MODE

In Demand Transfer mode the DMA channel is programmed to continue making transfers until a TC (Terminal Count) is encountered or an external EOP is encountered, or until the DMA I/O device pulls DREQ inactive. Thus, transfers may continue until the I/O device has exhausted its data capacity. After the I/O device catches up, the DMA service is re-established when the DMA I/O device reasserts the channel's DREQ. During the time between services when the system is allowed to operate, the intermediate values of address and byte/word count are stored in the DMA controller Current Address and Current Byte/Word Count registers. A TC can cause a new buffer to be loaded via Scatter-Gather, buffer chaining or Autoinitialize at the end of the service if the channel has been programmed for it.

6.2.4 CASCADE MODE

This mode is used to cascade more than one DMA controller together for simple system DMA requests for the additional device propagate through the priority network circuitry of the preceding device. The priority chain is preserved and the new device must wait for its turn to acknowledge requests. Within the ESC architecture, Channel 0 of DMA Controller two (DMA-2, Ch 4) is used to cascade DMA Controller one (DMA-1) to provide a total of seven DMA channels. Channel 0 on DMA-2 (labeled Ch 4 overall) connects the second half of the DMA system. This channel is not available for any other purpose.

In Cascade Mode, the DMA Controller will respond to DREQ with DACK, but the ESC will not drive the bus.

Cascade mode is also used to allow direct access of the system by 16-bit bus masters. These devices use the DREQ and DACK signals to arbitrate for the system bus and then they drive the address and command lines to control the bus. The ISA master asserts its ISA master request line (DREQx) to the DMA internal arbiter. If the ISA master wins the arbitration, the ESC responds with an ISA Master Acknowledge (DACKx) signal active. Upon sampling the DACKx line active, the ISA Master asserts MASTER16# signal and takes control of the EISA bus. The ISA Master has control of the EISA Bus, and the ISA Master may run cycles until it negates the MASTER16# signal.

6.3 DMA Transfer Types

Each of the three active transfer modes (Single, Block, or Demand) can perform three different types of transfers. These transfers are Read, Write and Verify.

Write Transfer

Write transfers move data from an EISA/ISA I/O device to memory located on EISA/ISA Bus or PCI Local Bus. The DMA indicates the transfer type to the EISA bus controller. The bus controller will activate IORC# and the appropriate EISA control signals (M/IO# and W/R#) to indicate a memory write.

Read Transfer

Read transfers move data from EISA/ISA or PCI memory to an EISA/ISA I/O device. The DMA indicates the transfer type to the EISA bus controller. The bus controller will activate IOWC# and the appropriate EISA control signals (M/IO# and W/R#) to indicate a memory read.

Verify Transfer

Verify transfers are pseudo transfers. The DMA controller operates as in Read or Write transfers, generating addresses and producing TC, etc. However, the ESC does not assert the memory and I/O control signals. Only the DACK signals are asserted. Internally the DMA controller will count BCLKs so that the DACK signals have a defined pulse width. This pulse width is nine BCLKs long. If Verify transfers are repeated during Block or Demand DMA requests, each additional pseudo transfer will add eight BCLKs. The DACK signals will not be toggled for repeated transfers.

6.4 DMA Timing

The ESC DMA provides four transfer timings. In addition to the compatible timings, the ESC DMA provides Type "A", Type "B", and Type "C" (Burst) timings for I/O slave devices capable of running at faster speeds.

6.4.1 COMPATIBLE TIMINGS

Compatible timing is provided for DMA slave devices. Compatible timing runs at 9 BCLKs (1080 ns/single cycle) and 8 BCLKs (960 ns/cycle) during the repeated portion of a Block or Demand mode transfers.

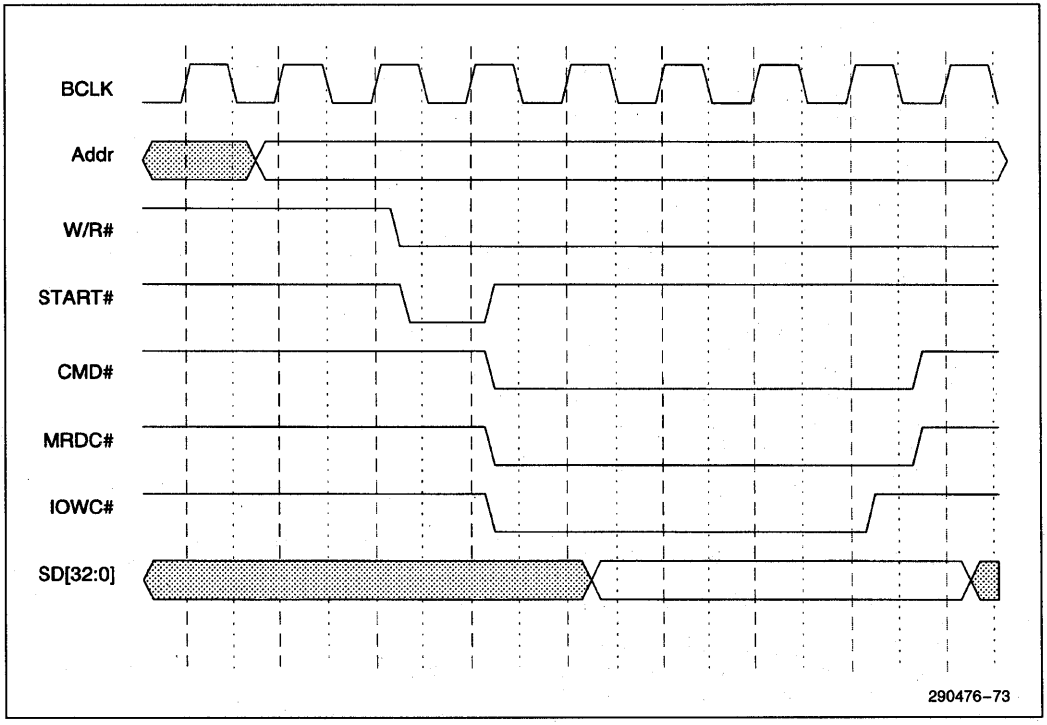


Figure 6-2a. Compatible DMA Read Transfer (8 BCLKs)

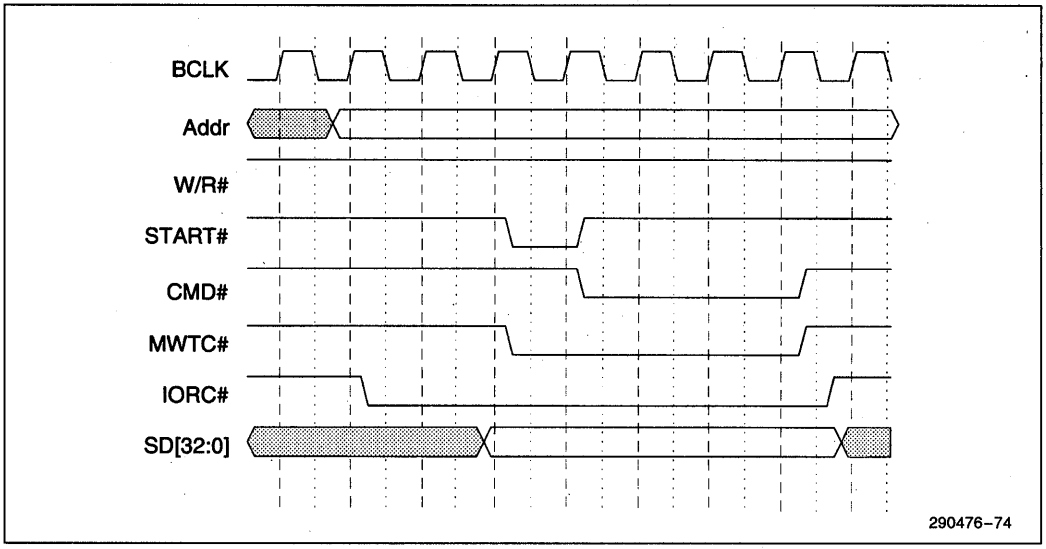


Figure 6-2b. Compatible DMA Write Transfer (8 BCLKS)

6.4.2 TYPE "A" TIMING

Type "A" timing is provided to allow shorter cycles to EISA memory.

NOTE:
Main memory behaves like EISA memory because the PCEB has an EISA slave interface.

Type "A" timing runs at 7 BCLKs (840 ns/single cycle) and 6 BCLKs (720 ns/cycle) during the repeated portion of a Block or Demand mode transfer. Type "A" timing varies from compatible timing primarily in shortening the memory operation to the minimum allowed by system memory. The I/O portion of the cycle (data setup on write, I/O read access time) is the same as with compatible cycles. The actual active command time is shorter, but it is expected that the DMA devices which provide the data access time or write data setup time should not require excess IORC# or IOWC# command active time. Because of this, most DMA devices should be able to use type "A" timing.

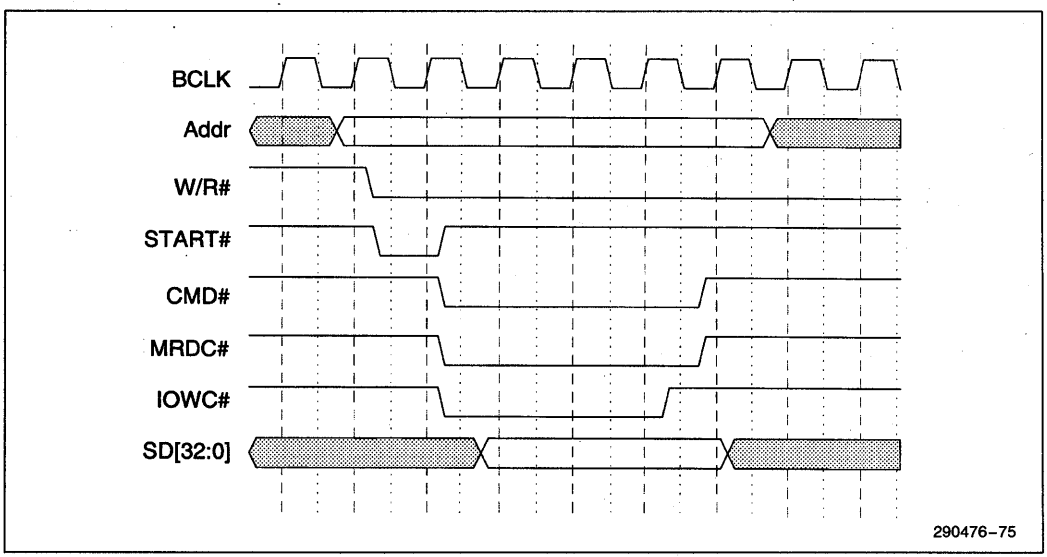


Figure 6-3a. Type "A" DMA Read Transfers (6 BCLKS)

6.4.3 TYPE "B" TIMING

Type "B" timing is provided for 8-bit/16-bit DMA devices which can accept faster I/O timing. Type "B" only works with fast system memory. Type "B" timing runs at 6 BCLKs (720 ns/single cycle) and 4 BCLKs (480 ns/cycle) during the repeated portion of a Block or Demand mode transfer. Type "B" timing requires faster DMA slave devices than compatible timing in that the cycles are shortened so that the data setup time on I/O write cycles is shortened and the I/O read access time is required to be faster. Some of the current ISA devices should be able to support type "B" timing, but these will probably be more recent designs using relatively fast technology.

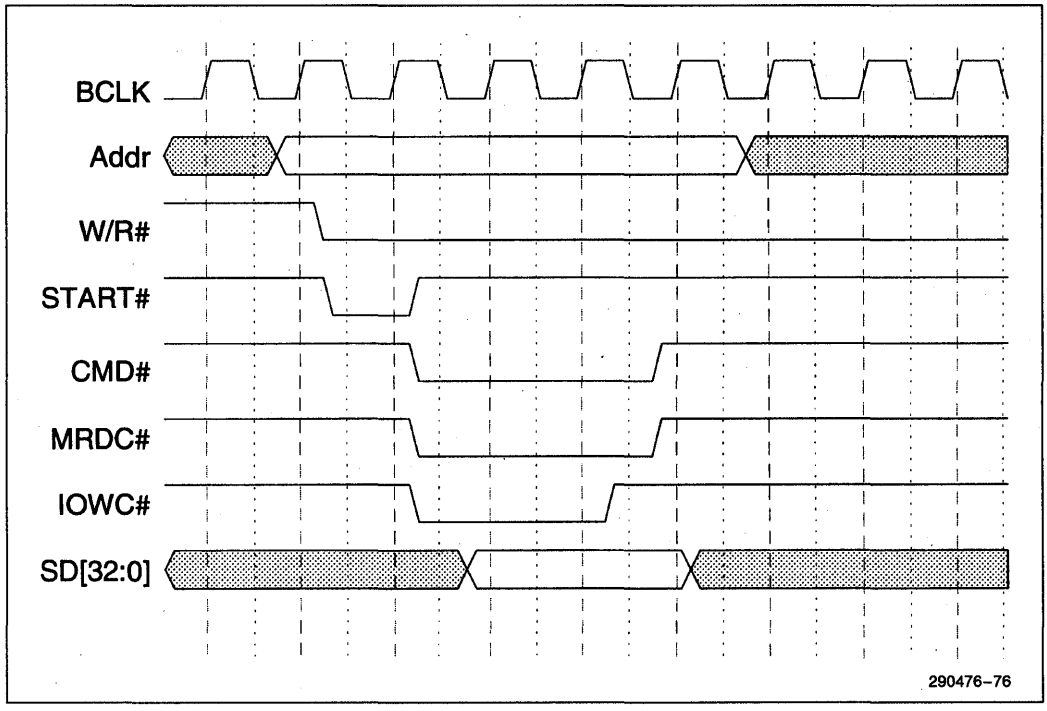


Figure 6-4a. Type "B" DMA Read Transfer (4 BCLKS)

6.4.4 TYPE "C" (BURST) TIMING

Type "C" (burst) timing is provided for EISA DMA devices. The DMA slave device needs to monitor EXRDY and IORC# or IOWC# signals to determine when to change the data (on writes) or sample the data (on reads). This timing will allow up to 33 MBytes per second transfer rate with a 32-bit DMA device and 32-bit memory. Note that 8-bit or 16-bit DMA devices are supported (through the programmable DMA address increment) and that they use the "byte lanes" natural to their size for the data transfer. As with all bursts, the system will revert to two BCLK cycles if the memory does not support burst. When a DMA burst cycle accesses non-burst memory and the DMA cycle crosses a page boundary into burstable memory, the ESC will continue performing standard (non-burst) cycles. This will not cause a problem since the data is transferred correctly.

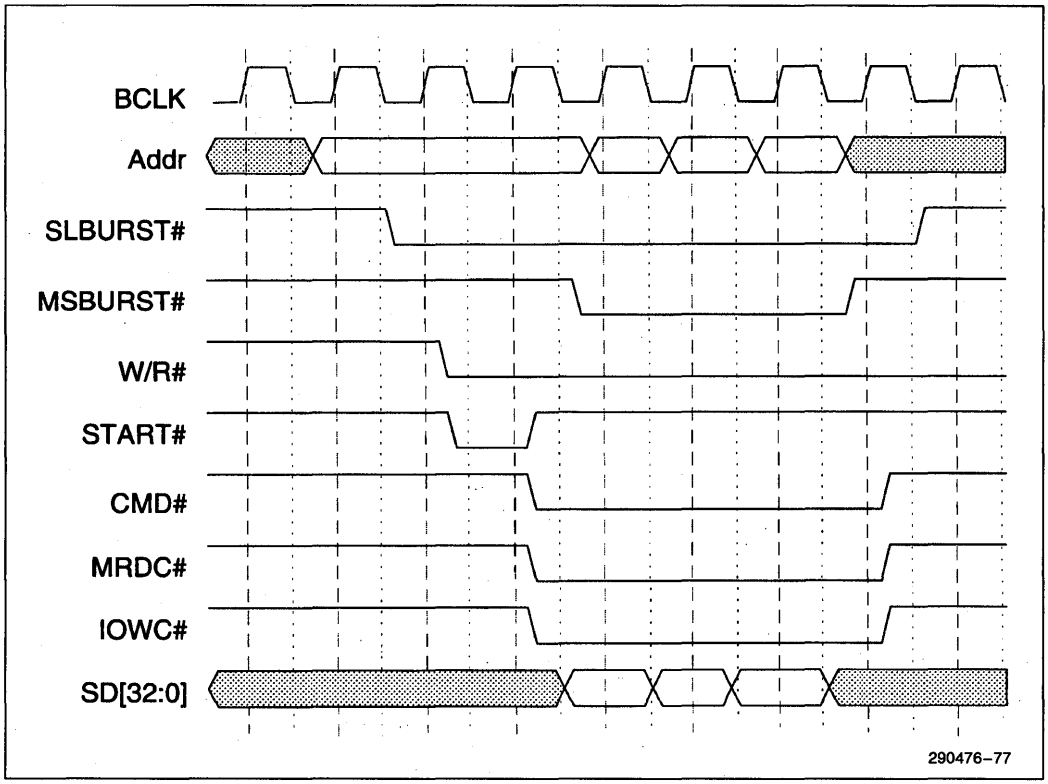


Figure 6-5a. Type "C" (Burst) DMA Read Transfers (1 BCLK)

6.5 Channel Priority

For priority resolution the DMA consists of two logical channel groups-channels 0–3 and channels 4–6. Each group may be in either Fixed or Rotate mode, as determined by the Command register.

For arbitration purposes, the source of the DMA request is transparent. DMA I/O slaves normally assert their DREQ line to arbitrate for DMA service. However, a software request for DMA service can be presented through each channel's DMA Request register. A software request is subject to the same prioritization as any hardware request. Please see the detailed register description in Section 3.2 for Request Register programming information.

Fixed Priority

The initial fixed priority structure is as follows:

Table 6-1. Initial Fixed Priority Structure

High Priority	Low Priority
(0, 1, 2, 3)	5, 6, 7

The fixed priority ordering is 0, 1, 2, 3, 5, 6, and 6. In this scheme, Channel 0 has the highest priority, and Channel 7 has the lowest priority. Channels [3:0] of DMA-1 assume the priority position of Channel 4 in DMA-2, thus taking priority over Channels 5, 6, and 7.

Rotating Priority

Rotation allows for "fairness" in priority resolution. The priority chain rotates so that the last channel serviced is assigned the lowest priority in the Channel group (0–3, 5–7).

Channels 0–3 rotate as a group of 4. They are always placed between Channel 5 and Channel 7 in the priority list.

Channel 5–7 rotate as part of a group of 4. That is, Channels (5–7) form the first three partners in the rotation, while Channel group (0–3) comprises the fourth position in the arbitration.

Table 6-2 demonstrates rotation priority:

Table 6-2. Rotating Priority Example

Programmed Mode	Action	Priority
		High . . . Low
Group (0–3) is in rotation mode.	1. Initial Setting	(0, 1, 2, 3), 5, 6, 7
Group (4–7) is in fixed mode.	2. After servicing channel 2	(3, 0, 1, 2), 5, 6, 7
	3. After servicing channel 3	(0, 1, 2, 3), 5, 6, 7
Group (0–3) in rotation mode.	1. Initial Setting	(0, 1, 2, 3), 5, 6, 7
Group (4–7) is in rotation mode.	2. After servicing channel 0	5, 6, 7, (1, 2, 3, 0)
	3. After servicing channel 5	6, 7, (1, 2, 3, 0), 5
(note that the first servicing of channel 0 caused double rotation).	4. After servicing channel 6	7, (1, 2, 3, 0), 5, 6
	5. After servicing channel 7	(1, 2, 3, 0), 5, 6, 7

6.6 Scatter-Gather Functional Description

Scatter-Gather provides the capability of transferring multiple buffers between memory and I/O without CPU intervention. In Scatter-Gather, the DMA can read the memory address and word count from an array of buffer descriptors called the Scatter-Gather Descriptor (SGD) Table. This allows the DMA to sustain DMA transfers until all buffers in the Scatter-Gather Descriptior Table are transferred.

The Scatter-Gather Command register and Scatter-Gather Status register are used to control the operational aspect of Scatter-Gather transfers (see Section 3.2 for details of these registers). The Scatter-Gather Descriptior Next Link register holds the address of the next buffer descriptor in the Scatter-Gather Descriptor Table.

The next buffer descriptor is fetched from the Scatter-Gather Descriptor Table by a DMA read transfer. DACK# will not be asserted for this transfer because the I/O device is the DMA itself and the DACK is internal to the ESC. The ESC will assert IOWC# for these bus cycles like any other DMA

transfer. The ESC will behave as an 8-bit I/O slave and will run type "B" timings for a Scatter-Gather buffer descriptor transfer. EOP will be asserted at the end of the transfer.

To initiate a typical Scatter-Gather transfer between memory and I/O device following steps involved:

Software prepares a Scatter-Gather Descriptor (SGD) Table in system memory. Each Scatter-Gather descriptor is 8 bytes long and consists of an address pointer to the starting address and the transfer count of the memory buffer to be transferred. In any given SGD Table, two consutive SGDs are offset by 8 bytes and are aligned on a 4-byte boundary.

Each Scatter-Gather Descriptor for the linked list must contain following information:

a) Memory Address (buffer start) 4 bytes
b) Byte Count (buffer size) 3 bytes
c) End of Link List 1 bit (MSB)

Initialize DMA Mode and Extended Mode registers with transfer specific information like 8-bit/16-bit I/O device, Transfer Mode, Transfer Type, etc.

Software provides the starting address of the Scatter-Gather Descriptor Table by loading the Scatter-Gather Descriptor Table Pointer register.

Engage the Scatter-Gather machine by writing a Start command to the Scatter-Gather Command register.

The Mask register should be cleared as the last step of programming the DMA register set. This is to prevent DMA from starting a transfer with a partially loaded command description.

Once the register set is loaded and the channel is unmasked, the DMA will generate an internal request to fetch the first buffer from the Scatter-Gather Descriptor Table.

The DMA will then respond to DREQ or software requests. The first transfer from the first buffer will move the memory address and word count from the Base register set to the Current register set. As long as Scatter-Gather is active and the Base register set is not loaded and the last buffer has not been fetched, the channel will generate a request to fetch a reserve buffer into the Base register set. The reserve buffer is loaded to minimize latency problems going from one buffer to another. Fetching a reserve buffer has a lower priority than completing DMA for the channel.

The DMA controller will terminate a Scatter-Gather cycle by detecting an End of List (EOL) bit in the SGD. After the EOL bit is detected, the channel will transfer the buffers in the Base and Current register sets if they are loaded. At Terminal Count the channel will assert EOP or IRQ13 depending on its programming and set the Terminate bit in the Scatter-Gather Status register. The Active bit in the Scatter-Gather Status register will be reset and the channel's Mask bit will be set.

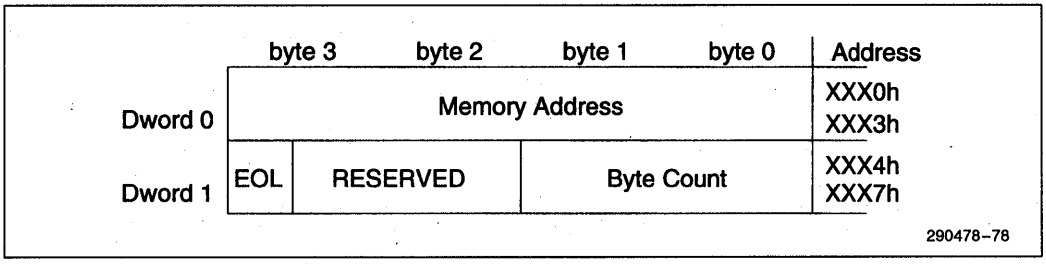


Figure 6-6. Scatter-Gather Descriptor Format

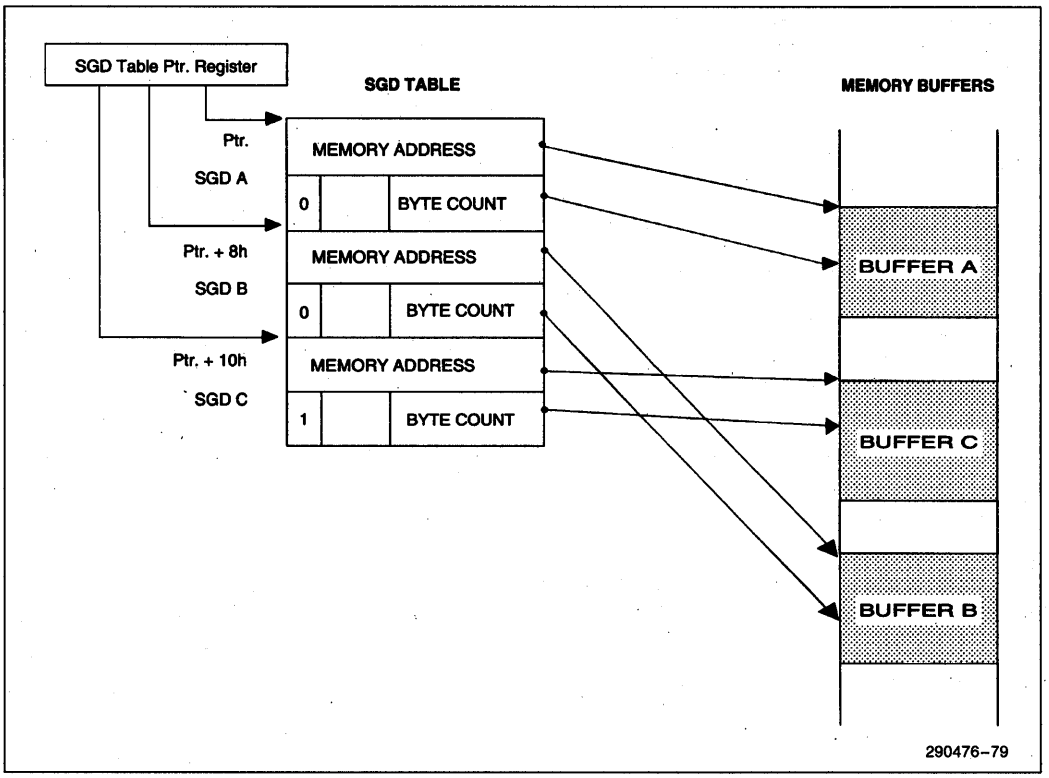


Figure 6-7. Link List Example

6.7 Register Functionality

Please see Section 3.2 for detailed information on register programming, bit definitions, and default values/functions after a reset.

DMA Channel 4 is used to cascade the two DMA controllers together and should not be programmed for any mode other than cascade. The Mode register for Channel 4 will default to cascade mode. Special attention should also be taken when programming the Command and Mask registers as related to Channel 4 (refer to the Command and Mask register descriptions, Section 3.2).

6.7.1 ADDRESS COMPATIBILITY MODE

Whenever the DMA is operating in Address Compatibility mode, the addresses do not increment or decrement through the High and Low Page registers, and the high page register is set to 00h. This is compatible with the 82C37 and Low Page register implementation used in the PC AT. This mode is set when any of the lower three address bytes of a channel are programmed. If the upper byte of a channel's address is programmed last, the channel will go into Extended Address Mode. In this mode, the high byte may be any value and the address will increment or decrement through the entire 32-bit address.

After reset is negated all channels will be set to Address Compatibility Mode. The DMA Master Clear command will also reset the proper channels to Address Compatibility Mode. The Address Compatibility Mode bits are stored on a per channel basis.

6.7.2 SUMMARY OF THE DMA TRANSFER SIZES

Table 6-3 lists each of the DMA device transfer sizes. The column labeled "Word Count Register" indicates that the register contents represents either the number of bytes to transfer or the number of 16-bit words to transfer. The column labeled "Current Address Register Increment/Decrement" indicates the number added to or taken from the Current Address register after each DMA transfer cycle. The Mode Register determines if the Current Address register will be incremented or decremented.

Table 6-3. DMA Transfer Size

DMA Device Date Size and Word Count	Word Count Register	Current Address Increment/ Decrement
8-Bit I/O, Count By Bytes	Bytes	1
16-Bit I/O,Count By Words (Address Shifted)	Words	1
16-Bit I/O,Count By Bytes	Bytes	2
32-Bit I/O,Count By Bytes	Bytes	4

6.7.3 ADDRESS SHIFTING WHEN PROGRAMMED FOR 16-BIT I/O COUNT BY WORDS

To maintain compatibility with the implementation of the DMA in the PCAT which used the 82C37, the DMA will shift the addresses when the Extended Mode register is programmed for, or defaulted to, transfers to/from a 16-bit device count-by-words. Note that the least significant bit of the Low Page register is dropped in 16-bit shifted mode. When programming the Current Address register while the DMA channel is in this mode, the Current Address must be programmed to an even address with the address value shifted right by one bit. The address shifting is shown in Table 6-4.

6.7.4 STOP REGISTERS (RING BUFFER DATA STRUCTURE)

To support a common data communication data structure, (the ring buffer), a set of DMA registers have been provided. These registers are called Stop registers. Each channel has 22 bits of register location associated with it. The 22 bits are distributed between three different registers (one six-bit and two eight-bit). The Stop registers can be enabled or disabled by writing to the channel's corresponding Extended Mode register.

The ring buffer data structure reserves a fixed portion of memory, on doubleword boundaries, to be used for a DMA channel. Consecutively received frames or other data structures are stored sequentially within the boundaries of the ring buffer memory.

The beginning and end of the ring buffer area is defined in the Base Address register and the Base Address register + the Base Byte/Transfer Count. The incoming frames (data) are deposited in sequential locations of the ring buffer. When the DMA reaches the end of the ring buffer, indicating the byte count has expired, the DMA controller (if so programmed) will Autoinitialize. Upon autoinitialization, the Current Address register will be restored from the Base Address register, taking the process back to the start of the ring buffer. The DMA will then be available to begin depositing the incoming bytes in the ring buffers sequential locations, providing that the CPU has read the data that was previously placed in those locations. The DMA determines that the CPU has read certain data by the value that the CPU writes into the Stop register.

Table 6-4. Address Shifting in 16-Bit I/O DMA Transfers

Output Address	8-Bit I/O Programmed Address	16-Bit I/O Programmed Address (Shifted)	16-Bit I/O Programmed Address (No Shift)	32-Bit I/O Programmed Address (No Shift)
A0 A[16:1] A[31:17]	A0 A[16:1] A[31:17]	"0" A[15:0] A[31:17]	A0 A[16:01] A[31:17]	A0 A[16:01] A[31:17]

NOTE:
The least significant bit of the Low Register is dropped in 16-bit shifted mode.

Once the data of a frame is read by the CPU, the memory location it occupies becomes available for other incoming frames. The Stop register prevents the DMA from over writing data that has not yet been read by the CPU. After the CPU has read a frame from memory it will update the Stop register to point to the location that was last read. The DMA will not deposit data into any location beyond that pointed to by the top register. The last address transferred before the channel is masked is the first address that matches the Stop register.

For example:

If the stop register = 00001Ch, the last three transfers will be:

Table 6-5. Stop Register Functionality Example

	By Bytes	By Words	By Words
Increment	XX00001Ah	XX000018h	XX000018h
	XX00001Bh	XX00001Ah	XX00001Ah
	XX00001Ch	XX00001Ch	XX00001Ch
Decrement	XX000021h	XX000023h	XX000023h
	XX000020h	XX000021h	XX000021h
	XX00001Fh	XX00001Fh	XX00001Fh

The Stop registers store values to compare against LA[23:2] only, so the size of the ring buffer is limited to 16 Megabytes.

6.7.5 BUFFER CHAINING MODE AND STATUS REGISTERS

The Chaining Mode registers are used to implement the buffer chaining mode of a channel. The buffer chaining mode is useful when transferring data from a peripheral to several different areas of memory with one continuous transfer operation. Four registers are used to implement this function: the Chaining Mode register, the Chaining Mode Status Register, the Channel Interrupt Status register, and the Chain Buffer Expiration Control register.

The Chaining Mode register controls the buffer chaining initialization. Buffer chaining mode can be enabled or disabled. A Chaining Mode bit is used to indicate if Base register programming is complete and chaining can begin, or to hold off chaining because the Base registers still need programming. Another bit dictates the buffer expiration response by indicating whether an IRQ13 or EOP should be issued when the buffer needs reprogramming.

The Chaining Mode Status Register indicates whether each channel's chaining mode is enabled or disabled.

The Channel Interrupt Status Register indicates the channel source of a DMA chaining interrupt on IRQ13. The CPU can read this register to determine which channel asserted IRQ13 following a buffer expiration.

The Chain Buffer Expiration Control Register is a read only register that reflects the outcome after the expiration of a chain buffer. If a channel bit is set to 0, IRQ13 will be activated following the buffer expiration. If a channel bit is set to 1, EOP will be asserted following the buffer expiration.

6.7.6 AUTOINITIALIZE

By programming a bit in the Mode register, a channel may be set up as an Autoinitialize channel. During Autoinitialize initialization, the original values of the Current page, Current address and Current Byte/Word Count registers are automatically restored from the Base Address, and Word count registers of that channel following TC. The Base registers are loaded simultaneously with the Current registers by the microprocessor and remain unchanged throughout the DMA service. The mask bit is not set when the channel is in Autoinitialize. Following Autoinitialize the channel is ready to perform another DMA service, without CPU intervention, as soon as a valid DREQ is detected.

NOTE:
Autoinitialize will not function if the channel is also programmed for Scatter-Gather or buffer chaining. Only one of these features should be enabled at a time.

6.8 Software Commands

These are additional special software commands which can be executed in the Program Condition. They do not depend on any specific bit pattern on the data bus. The three software commands are:

1. Byte Pointer Flip-Flop
2. Master Clear, and
3. Clear Mask Register.

6.8.1 CLEAR BYTE POINTER FLIP-FLOP

This command is executed prior to writing or reading new address or word count information to the DMA. This initializes the flip-flop to a known state so that subsequent accesses to register contents by the microprocessor will address upper and lower bytes in the correct sequence.

When the CPU is reading or writing DMA registers, two Byte Pointer Flip-Flops are used; one for Channels 0–3 and one for Channels 4–6. Both of these act independently. There are separate software commands for clearing each of them (0Ch for Channels 0–3, 0D8h for Channels 4–7).

An additional Byte Pointer Flip-Flop has been added for use when EISA masters are reading and writing DMA registers. (The arbiter state will be used to determine the current master of the bus.) This Flip-Flop is cleared when an EISA Master performs a write to either 00Ch or 0D8h. There is one Byte Pointer Flip-Flop per eight DMA channels. This Byte Pointer was added to eliminate the problem of the CPU's byte pointer getting out of synchronization if an EISA Master takes the bus during the CPU's DMA programming.

6.8.2 DMA MASTER CLEAR

This software instruction has the same effect as the hardware Reset. The Command, Status, Request, and Internal First/Last Flip-Flop registers are cleared and the Mask register is set. The DMA Controller will enter the idle cycle.

There are two independent Master Clear Commands, 0Dh which acts on Channels 0–3, and 0DAh which acts on Channels 4–6.

6.8.3 CLEAR MASK REGISTER

This command clears the mask bits of all four channels, enabling them to accept DMA requests. I/O port 00Eh is used for Channels 0–3 and I/O port 0DCh is used for Channels 4–6.

6.9 Terminal Count/EOP Summary

This is a summary of the events that will happen as a result of a terminal count or external EOP when running DMA in various modes.

Table 6-6. Terminal Count/EOP Summary Table

Conditions				
AUTOINIT	No		Yes	
Event				
Word Counter Expired	Yes	X	Yes	X
EOP Input	X	Asserted	X	Asserted
Result				
Status TC	set	set	set	set
Mask	set	set	—	—
SW Request	clr	clr	clr	clr
Current Register	—	—	load	load

NOTES:
load = Load Current From Base
"—" = No Change
X = Don't Care
clr = Clear

6.10 Buffer Chaining

The buffer chaining mode of a channel is useful for transferring data from a peripheral to several different areas of memory within one transfer operation (from the DMA device's viewpoint). This is accomplished by causing the DMA to interrupt the CPU for more programming information while the previously programmed transfer is still in progress. Upon completion of the previous transfer, the DMA controller will then load the new transfer information automatically. In this way, the entire transfer can be completed without interrupting the operation of the DMA device. This mode is most useful for DMA single-cycle or demand modes where the transfer process allows time for the CPU to execute the interrupt routine.

The buffer chaining mode of a channel may be entered by programming the address and count of a transfer as usual. After the initial address and count is programmed, the Base registers are selected via the Chaining Mode register Chaining Mode Enabled bit. The address and count for the second transfer and both the Chaining Mode Enabled and the Program Complete bit of the Chaining Mode register should be programmed at this point, before starting the DMA process. When, during the DMA process, the Current Buffer is expired, the Base address, Page, and Count registers will be transferred to the Current registers and a signal that the buffer has been expired is sent to the programming master.

This signal will be an IRQ13 if the master is the CPU, or a TC if the programming master is an EISA Master device. The type of programming master is indicated in the DMA's Chaining Mode Register, Bit 4. If the CPU is the programming master for the Channel, TC will be generated only if the Current buffer expires and there is no Next Buffer stored in the Base registers.

Upon the expiration of a Current Buffer, the new Base register contents should be programmed and both the Chaining Mode Enabled and Program complete bits of the Chaining Mode register should be set. This resets the interrupt, if the CPU was the programming master, and allows for the next Base register to Current register transfer. If the Program Complete bit is not set before the current transfer reaches TC, then the DMA controller will set the Mask Bit and the TC bit in the Status register and stop transferring data. In this case, an over-run is likely to occur. To determine if this has, a read of either Status register or the Mask register can be done (the Mask register has been made readable). If the channel is masked or has registered a TC, the DMA channel has been stopped and the full address, count, and chaining mode must be programmed to return to normal operation.

Note that if the CPU is the programming master, an interrupt will only be generated if a Current Buffer expires and chaining mode is enabled. It will not occur during initial programming. The Channel Interrupt Status register will indicate pending interrupts only. That is, it will indicate an empty Base register with Chaining Mode enabled. When Chaining mode is enabled, only the Base registers are written by the processor, and only the Current registers can be read. The Current registers are only updated on a TC.

6.11 Refresh Unit

The ESC provides an EISA Bus compatible refresh unit that provides 14 bits of refresh address for EISA/ISA bus DRAMS that do not have their own local refresh units. The refresh system uses the combined functions of the Interval Timers, the DMA Arbiter, DMA address counter, and EISA Bus Controller. Functionally the Refresh unit is a sub-section of the ESC DMA unit. The DMA Address Counter is used to increment the Refresh Address register following each refresh cycle. Interval Counter 1, Timer 1 generates an internal refresh request. The DMA Arbiter detects a Refresh signal from either the Counter/Timer or the REFRESH# input and determines when the refresh will be done. The DMA drives the refresh address out onto the LA address bus. The cycle is decoded and driven onto the EISA address bus by the EISA Bus Controller. The ESC EISA Bus Controller is responsible for generating the EISA cycle control signals. Timer 1 Counter 1 should be programmed to provide a refresh request about every 15 μs.

Requests for refresh cycles are generated by two sources: the ESC (Timer 1 Counter 1), and 16-bit masters that activate REFRESH# when they own the EISA bus.

If a 16-bit ISA bus master holds the bus longer than 15 μs, it must initiate memory refresh cycles. If the ISA Master initiates a Refresh cycle while it owns the bus, it floats the address lines and cycle control signals and asserts REFRESH# to the ESC. The ESC EISA Bus Controller generates the cycle control signals and the ESC DMA Refresh unit supplies the refresh address. The ISA Master must then wait one BCLK after MRDC# is negated before floating REFRESH# and driving the address lines and control signals.

Typically, the refresh cycle length is five BCLK's. The I/O slave can insert one wait state to extend the cycle to six BCLK's by asserting CHRDY. The ESC EISA Bus Controller, upon seeing REFRESH#, knows to run refresh cycles instead of DMA cycles.

7.0 EISA BUS ARBITRATION

The ESC receives requests for EISA Bus ownership from several different sources; from DMA devices, from the Refresh counter, from EISA masters and from PCI agents. PCI agents requesting the EISA Bus request the EISA Bus through the PCEB. Additionally, 16-bit ISA Masters may request the bus through a cascaded DMA channel (see the Cascade mode description in Section 6.2.4).

7.1 Arbitration Priority

At the top level of the arbiter, the ESC uses a three way rotating priority arbitration method. On a fully loaded bus, the order in which the devices are granted bus access is independent of the order in which they assert a bus request, since devices are serviced based on their position in the rotation. The arbitration scheme assures that DMA channels and EISA masters are able to access the bus with minimal latency.

The PCEB and EISA Masters share one of the slots in the three way rotating priority scheme. This sharing is a two way rotation between the CPU and EISA Masters as a group. In this arbitration scheme the PCEB acts on behalf of the CPU and all other PCI masters.

EISA Masters have a rotating priority structure which can handle up to eight master requests.

The next position in the top level arbiter is occupied by the DMA. The DMA's DREQ lines can be placed in either fixed or rotating priority. The default mode is fixed and by programming the DMA Command registers, the priority can be modified to rotating priority mode.

7.2 Preemption

An EISA compatible arbiter ensures that minimum latencies are observed for both EISA DMA devices, and EISA Masters.

7.2.1 PCEB EISA BUS ACQUISITION AND PCEB PREEMPTION

EISA Bus arbitration is intended to be optimized for CPU access the EISA bus. Since the CPU accesses the EISA Bus through the PCEB, the PCEB is assumed to be the default owner of the EISA bus. The arbitration interface between the PCEB and the ESC is implemented as a HOLD/HLDA (EISAHOLD/EISAHLDA) pair.

If a PCI cycle requires access to the EISA Bus while EISAHLDA signal is asserted (EISA Bus busy) the PCI cycle is retried, and the PCEB requests the EISA bus by asserting PEREQ#. The ESC after sampling PEREQ# asserted preempts the current owner of the EISA Bus. The ESC grants the EISA Bus to by negating EISAHOLD signal.

The ESC asserts EISAHOLD to the PCEB when the ESC needs to acquire the ownership of the EISA bus. While EISAHOLD is asserted the arbitration process is dynamic and may change i.e., the ESC is still accepting EISA Bus requests. When the PCEB returns EISAHLDA, the arbiter freezes the arbitration process and determines the winner. If the new winner is an EISA Master or DMA channel, the ESC will assert NMFLUSH#. The ESC tristates the NMFLUSH# output driver on the following clock. The PCEB holds NMFLUSH# asserted until all buffers are flushed. After all buffers are flushed, the PCEB negates NMFLUSH# and then tristate the output buffer. After sampling NMFLUSH# negated, the ESC resumes driving NMFLUSH# on the next PCI clock. The way the ESC does not assert MACK# or DACK# until the PCEB acknowledges that all line buffers have been flushed.

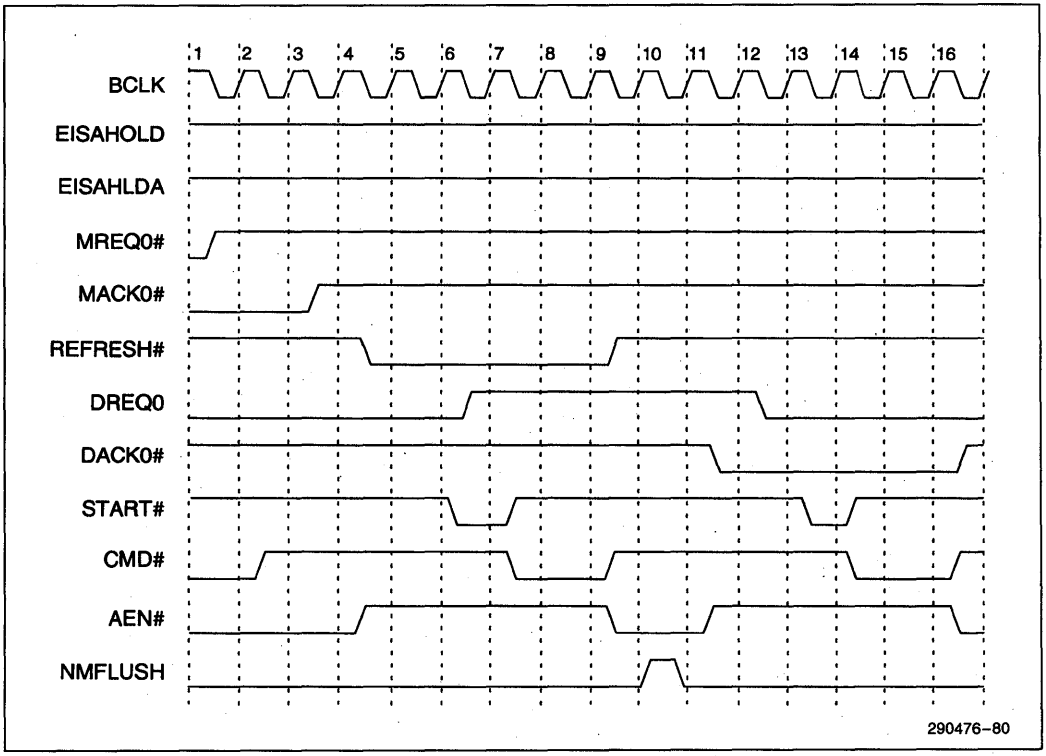


Figure 7-1. EISA Arbitration

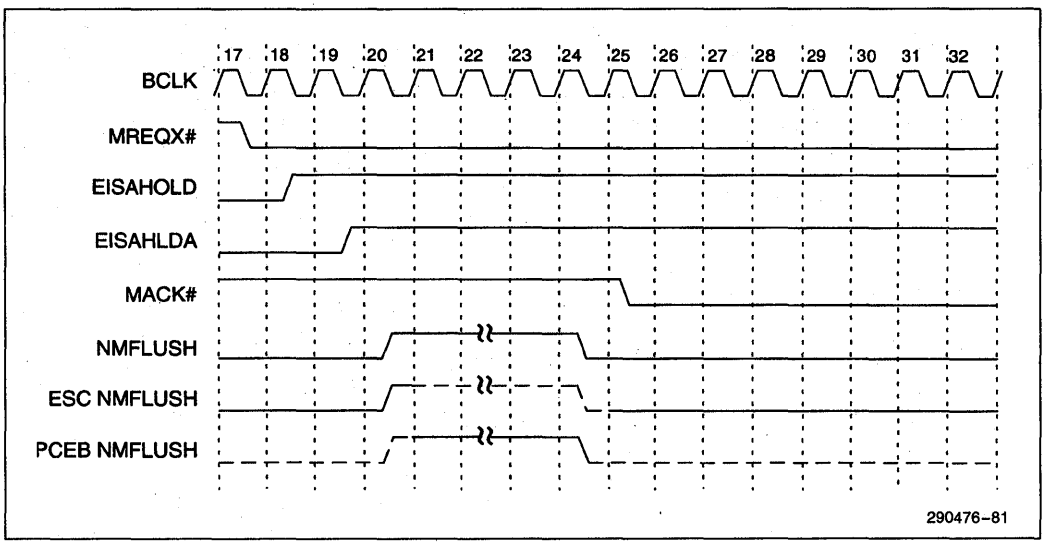


Figure 7-2. PCEB Preemption

7.2.2 EISA MASTER PREEMPTION

EISA specification requires that EISA Masters must release the bus within 64 BCLKs (8 μs) after the ESC negates MACKx#. If the bus master attempts to start a new bus cycle after this timeout period, a bus timeout (NMI) is generated and the RSTDRV is asserted to reset the offending bus master.

7.2.3 DMA PREEMPTION

A DMA slave device that is not programmed for compatible timing is preempted from the EISA Bus by another device that requests use of the bus. This will occur regardless of the priority of the pending request. For DMA devices not using compatible timing mode, the DMA controller stops the DMA transfer and releases the bus within 32 BCLK (4 μs) of a preemption request. Upon the expiration of the 4 μs timer, the DACK is negated after the current DMA cycle has completed. The EISA Bus then arbitrated for and granted to the highest priority requester. This feature allows flexibility in programming the DMA for long transfer sequences in a performance timing mode while guaranteeing that vital system services such as Refresh are allowed access to the expansion bus.

The 4 μs timer is not used in compatible timing mode. It is only used for DMA channels programmed for Type "A", Type "B", or Type "C" (Burst) timing. The 4 μs timer is also not used for 16-bit ISA masters cascaded through the DMA DREQ lines.

If the DMA channel that was preempted by the 4 μs timer is operating in Block mode, an internal bit will be set so that the channel will be arbitrated for again, independent of the state of DREQ.

7.3 Slave Timeouts

A slave which does not release EXRDY or CHRDY can cause the CMD# active time to exceed 256 BCLKs (32 μs). The ESC does not monitor EXRDY or CHRDY for this timeout. Typically this function is provided in a system through a third party add-in card. The add-in cards which monitor EXRDY or CHRDY assert IOCHK signal when the 256 BCLK count expires. The ESC in response asserts NMI.

The only way that a 16-bit ISA Master can be preempted from the EISA bus is if it exceeds the 256 BCLK (32 μs) limit on CMD# active.

7.4 Arbitration During Non-Maskable Interrupts

If a non-maskable interrupt (NMI) is pending at the PCEB, and the PCEB is requesting the bus, the DMA and EISA Masters will be bypassed each time they come up for rotation. This gives the PCEB the EISA Bus bandwidth on behalf of the CPU to process the interrupt as fast as possible.

8.0 INTERVAL TIMERS

The ESC contains five counter/timers that are equivalent to those found in the 82C54 programmable interval timer. The five counters are contained in two separate ESC timer units, referred to as Timer 1 and Timer 2. The ESC uses the Timers to implement key EISA system functions. Timer 1 contains three counters, and Timer 2 contains two counters. EISA systems do not use the middle counter on Timer 2.

Interval Timer 1, Counter 0 is connected to the interrupt controller IRQ0 and provides a system timer interrupt for a time-of-day, diskette time-out, or other system timing functions. Counter 1 generates a refresh-request signal and Counter 2 generates the tone for the speaker.

Interval Timer 2, Counter 0 implements a fail safe timer. Counter 0 generates NMI at regular intervals, thus preventing the system from locking up. Counter 1 is not used. Counter 2 is used to slow down the CPU by means of pulse-width modulation. The output of Timer-2 Counter 2 is tied to the SLOWH# signal.

Table 8-1. Interval Timer Functions

Interval Timer Functions		
Function	**Counter 0 System Timer**	**Counter 0 Fail-Safe Timer**
Gate	Always On	Always On
Clock In	1.193 MHz (OSC/12)	0.298 MHz (OSC/48)
Out	INT-1 IRQ0	NMI Interrupt
	Counter 1 Refresh Request	
Gate	Always On	
Clock In	1.193 MHz (0SC/12)	
Out	Refresh Request	
	Counter 2	**Counter 2**
Gate	Programmable Port 61h	Refresh Request
Clock In	1.193 MHz (OSC/12)	8 MHz (BCLK)
Out	Speaker	CPU Speed Control (SLOWH#)

8.1 Interval Timer Address Map

The following table shows the I/O address map of the interval timer counters:

Table 8-2. Interval Timer I/O Address Map

I/O Port Address	Register Description
040h	Timer 1, System Timer (Counter 0)
041h	Timer 1, Refresh Request (Counter 1)
042h	Timer 1, Speaker Tone (Counter 2)
043h	Timer 1, Control Word Register
048h	Timer 2, Fail-Safe Timer (Counter 0)
049h	Timer 2, Reserved
04Ah	Timer 2, CPU Speed Control (Counter 2)
04Bh	Timer 2, Control Word Register

Timer 1—Counter 0, System Timer

This counter functions as the system timer by controlling the state of IRQ[0] and is typically programmed for Mode 3 operation. The counter produces a square wave with a period equal to the product of the counter period (838 ns) and the initial count value. The counter loads the initial count value one counter period after software writes the count value to the counter I/O address. The counter initially asserts IRQ[0] and decrements the count value by two each counter period. The counter negates IRQ[0] when the count value reaches 0. It then reloads the initial count value and again decrements the initial count value by two each counter period. The counter then asserts IRQ[0] when the count value reaches "0", reloads the initial count value, and repeats the cycle, alternately asserting and negating IRQ[0].

Timer 1—Counter 1, Refresh Request Signal

This counter provides the Refresh Request signal and is typically programmed for Mode 2 operation. The counter negates Refresh Request for one counter period (833 ns) during each count cycle. The initial count value is loaded one counter period after being written to the counter I/O address. The counter initially asserts Refresh Request, and negates it for 1 counter period when the count value reaches 1. The counter then asserts Refresh Request and continues counting from the initial count value.

Timer 1—Counter 2, Speaker Tone

This counter provides the speaker tone and is typically programmed for Mode 3 operation. The counter provides a speaker frequency equal to the counter clock frequency (1.193 MHz) divided by the initial count value. The speaker must be enabled by a write to port 061h (see Section 3.7 on the NMI Status and Control ports).

Timer 2—Counter 0, Fail-Safe Timer

This counter functions as a fail-safe timer by preventing the system from locking up. This counter generates an interrupt on the NMI line as the count expires by setting bit 7 on Port 0461. Software routines can avoid the Fail-Safe NMI by resetting the counter before the timer count expires.

Timer 2—Counter 2, CPU Speed Control

This counter generates the SLOWH# to the CPU and is typically programmed for Mode 1 operation. The counter is triggered by the refresh request signal generated by Timer 1—Counter 1 only. If the counter is programmed, the counters SLOWH# output will stop the CPU for the programmed period of the one-shot every time a refresh request occurs. This counter is not configured or programmed until a speed reduction in the system is required.

8.2 Programming the Interval Timer

The counter/timers are programmed by I/O accesses and are addressed as though they are contained in two separate 82C54 interval timers. Timer 1 contains three counters and Timer 2 contains two counters. Each Timer is controlled by a separate Control Word register.

The interval timer is an I/O-mapped device. Several commands are available:

- The Control Word Command specifies:
  —which counter to read or write
  —the operating mode
  —the count format (binary or BCD)
- The Counter Latch Command latches the current count so that it can be read by the system. The countdown process continues.
- The Read Back Command reads the count value, programmed mode, the current state of the OUT pins, and the state of the Null Count Flag of the selected counter.

The Read/Write Logic selects the Control Word register during an I/O write when address lines A1, A0 = 11. This condition occurs during an I/O write to port addresses 043h and 04Bh, the addresses for the Control Word Register on Timer 1 and Control Word Register on Timer 2 respectively. If the CPU writes to port 043h or port 04Bh, the data is stored in the respective Control Word Register and is interpreted as a Control Word used to define the operation of the Counters.

The Control Word Register is write-only. Counter Status information is available with the Read-Back Command.

The following Table lists the six operating modes for the interval counters. Section 8.4 describes each mode's function in detail.

Table 8-3. Counter Operating Modes

Mode	Function
0	Out signal on end of count (=0)
1	Hardware retriggerable one-shot
2	Rate generator (divide by n counter)
3	Square wave output
4	Software triggered strobe
5	Hardware triggered strobe

Because the timer counters wake up in an unknown state after power up, multiple refresh requests may be queued up. To avoid possible multiple refresh cycles after power up, program the timer counter immediately after power up.

Write Operations

Programming the interval timer is a simple process:

1. Write a control word.
2. Write an initial count for each counter.
3. Load the least and/or most significant bytes (as required by Control Word Bits 5, 4) of the 16-bit counter.

The programming procedure for the ESC timer units is very flexible. Only two conventions need to be observed. First, for each Counter, the Control Word must be written before the initial count is written. Second, the initial count must follow the count format specified in the Control Word (least significant byte only, most significant byte only, or least significant byte and then most significant byte).

Since the Control Word Register and the three Counters have separate addresses (selected by the A1, A0 inputs), and each Control Word specifies the Counter it applies to (SC0, SC1 bits), no special instruction sequence is required. Any programming sequence that follows the conventions above is acceptable.

A new initial count may be written to a Counter at any time without affecting the Counter's programmed Mode in any way. Counting will be affected as described in the Mode definitions. The new count must follow the programmed count format.

If a Counter is programmed to read/write two-byte counts, the following precaution applies: A program must not transfer control between writing the first and second byte to another routine which also writes into that same Counter. Otherwise, the Counter will be loaded with an incorrect count.

Interval Timer Control Word Format

The Control Word specifies the counter, the operating mode, the order and size of the COUNT value, and whether it counts down in a 16-bit or binary-coded decimal (BCD) format. After writing the control word, a new count may be written at any time. The new value will take effect according to the programmed mode.

If a counter is programmed to read/write two-byte counts, the following precaution applies: A program must not transfer control between writing the first and second byte to another routine which also writes into that same counter. Otherwise, the counter will be loaded with an incorrect count. The count must always be completely loaded with both bytes.

Read Operations

It is often desirable to read the value of a Counter without disturbing the count in progress. This is easily done in the ESC timer units.

There are three possible methods for reading the counters: a simple read operation, the Counter Latch Command, and the Read-Back Command.

Counter I/O Port Read

The first method is to perform a simple read operation. To read the Counter the CLK input of the selected Counter must be inhibited by using either the GATE input or external logic. Otherwise, the count may be in the process of changing when it is read, giving an undefined result. When reading the count value directly, follow the format programmed in the control register: read LSB, read MSB, or read LSB then MSB. Within the ESC timer unit, the GATE input on Timer 1 Counter 0, Counter 1 and Timer 2 Counter 0 are tied high. Therefore, the direct register read should not be used on these two counters. The GATE input of Timer 1 Counter 2 is controlled through I/O port 061h. If the GATE is disabled through this register, direct I/O reads of port 042h will return the current count value.

Counter Latch Command

The Counter Latch command latches the count at the time the command is received. This command is used to insure that the count read from the counter is accurate (particularly when reading a two-byte count). The count value is then read from each counter's Count register as was programmed by the Control register.

The selected Counter's output latch (OL) latches the count at the time the Counter Latch Command is received. This count is held in the latch until it is read by the CPU (or until the Counter is reprogrammed). The count is then unlatched automatically and the OL returns to "following" the counting element (CE). This allows reading the contents of the Counters "on the fly" without affecting counting in progress. Multiple Counter Latch Commands may be used to latch more than one Counter. Each latched Counter's OL holds its count until it is read. Counter Latch Commands do not affect the programmed Mode of the Counter in any way. The Counter Latch Command can be used for each counter in the ESC timer unit.

If a Counter is latched and then, some time later, latched again before the count is read, the second Counter Latch Command is ignored. The count read will be the count at the time the first Counter Latch Command was issued.

With either method, the count must be read according to the programmed format; specifically, if the Counter is programmed for two byte counts, two bytes must be read. The two bytes do not have to be read one right after the other; read, write, or programming operations for other Counters may be inserted between them.

Another feature of the ESC timer unit is that reads and writes of the same Counter may be interleaved. For example, if the Counter is programmed for two byte counts, the following sequence is valid:

1. Read least significant byte.
2. Write new least significant byte.
3. Read most significant byte.
4. Write new most significant byte.

One precaution is worth noting. If a Counter is programmed to read/write two-byte counts, a program must not transfer control between reading the first and second byte to another routine which also reads from that same Counter. Otherwise, an incorrect count will be read.

Read Back Command

The third method uses the Read-Back command. The Read-Back command is used to determine the count value, programmed mode, and current states of the OUT pin and Null Count flag of the selected counter or counters. The Read-Back command is written to the Control Word register, which causes the current states of the above mentioned variables to be latched. The value of the counter and its status may then be read by I/O access to the counter address.

The read-back command may be used to latch multiple counter output latches (OL) by setting the COUNT# bit D5 = 0 and selecting the desired counter(s). This single command is functionally equivalent to several counter latch commands, one for each counter latched. Each counter's latched count is held until it is read (or the counter is reprogrammed). Once read, a counter is automatically unlatched. The other counters remain latched until they are read. If multiple count read-back commands are issued to the same counter without reading the count, all but the first are ignored; i.e., the count which will be read is the count at the time the first read-back command was issued.

The read-back command may also be used to latch status information of selected counter(s) by setting STATUS# bit D4 = 0. Status must be latched to be read. The status of a counter is accessed by a read from that counter's I/O port address.

If multiple counter status latch operations are performed without reading the status, all but the first are ignored. The status returned from the read is the counter status at the time the first status read-back command was issued.

Both count and status of the selected counter(s) may be latched simultaneously by setting both the COUNT# and STATUS# bits [5:4] = 00b. This is functionally the same as issuing two consecutive, separate read-back commands. The above discussions apply here also. Specifically, if multiple count and/or status read-back commands are issued to the same counter(s) without any intervening reads, all but the first are ignored.

If both count and status of a counter are latched, the first read operation from that counter will return the latched status, regardless of which was latched first. The next one or two reads (depending on whether the counter is programmed for one or two type counts) return the latched count. Subsequent reads return unlatched count.

9.0 INTERRUPT CONTROLLER

The ESC provides an EISA compatible interrupt controller which incorporates the functionality of two 82C59 interrupt controllers. The two controllers are cascaded so that 14 external and two internal interrupts are possible. The master interrupt controller provides IRQ [7:0] and the slave interrupt controller provides IRQ [15:8] (see Figure 9-1). The two internal interrupts are used for internal functions only and are not available at the chip periphery. IRQ2 is used to cascade the two controllers together and IRQ0 is used as a system timer interrupt and is tied to Interval Timer 1, Counter 0. The remaining 14 interrupt lines (IRQ1, IRQ3–IRQ15) are available for external system interrupts. Edge or level sense selection is programmable on a by-controller basis.

The Interrupt Controller consists of two separate 82C59 cores. Interrupt Controller 1 (CNTRL-1) and Interrupt Controller 2 (CNTRL-2) are initialized separately, and can be programmed to operate in different modes. The default settings are: 80x86 Mode, Edge Sensitive (IRQ0-15) Detection, Normal EOI, Non-Buffered Mode, Special Fully Nested Mode disabled, and Cascade Mode. CNTRL-1 is connected as the Master Interrupt Controller and CNTRL-2 is connected as the Slave Interrupt Controller.

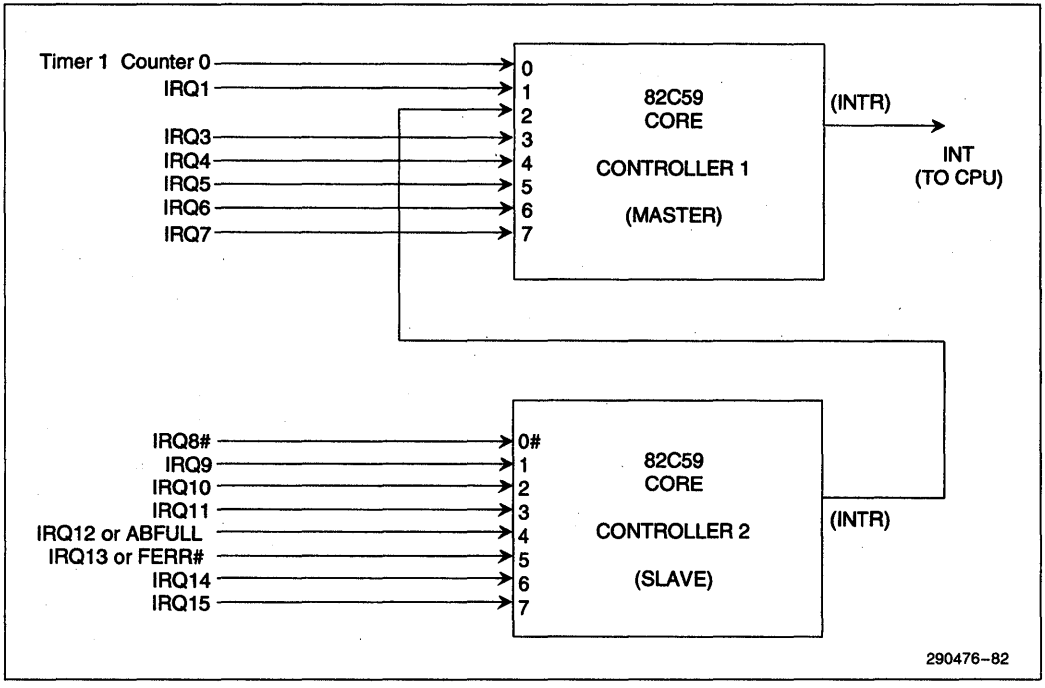


Figure 9-1. Block Diagram of The Interrupt Controller

The following Table lists the I/O port address map for the interrupt registers:

Table 9-1. I/O Address Map

Interrupts	I/O Address	# of Bits	Register
IRQ[7:0]	0020h	8	CNTRL-1 Control Register
IRQ[7:0]	0021h	8	CNTRL-1 Mask Register
IRQ[7:0]	04D0h	8	CNTRL-1 Edge/ Level Control Register
IRQ[15:8]	00A0h	8	CNTRL-2 Control Register
IRQ[15:8]	00A1h	8	CNTRL-2 Mask Register
IRQ[15:8]	04D1h	8	CNTRL-2 Edge/ Level Control Register

IRQ0, and IRQ2 are connected to the interrupt controllers internally. The other interrupts are always generated externally. IRQ12 and IRQ13 may be generated internally through the ABFULL and FERR# signals. respectively.

Table 9-2. Typical Interrupt Functions

Priority	Label	Controller	Typical Interrupt Source
1	IRQ0	1	Interval Timer 1, Counter 0 OUT
2	IRQ1	1	Keyboard
3–10	IRQ2	1	Interrupt from Controller 2
3	IRQ8#	2	Real Time Clock
4	IRQ9	2	Expansion Bus Pin B04
5	IRQ10	2	Expansion Bus Pin D03
6	IRQ11	2	Expansion Bus Pin D04
7	IRQ12	2	Expansion Bus Pin D05
8	IRQ13	2	Coprocessor Error, Chaining
9	IRQ14	2	Fixed Disk Drive Controller Expansion Bus Pin D07
10	IRQ15	2	Expansion Bus Pin D06
11	IRQ3	1	Serial Port 2, Expansion Bus B25
12	IRQ4	1	Serial Port 1, Expansion Bus B24
13	IRQ5	1	Parallel Port 2, Expansion Bus B23
14	IRQ6	1	Diskette Controller, Expansion Bus B22
15	IRQ7	1	Parallel Port 1, Expansion Bus B21

9.1 Interrupt Controller Internal Registers

Several registers are contained internally within each 82C59. The interrupts at the IRQ input lines are handled by two registers in cascade, the Interrupt Request Register (IRR) and the In-Service Register (ISR). The IRR is used to store all the interrupt levels which are requesting service and the ISR is used to store all the interrupt levels which are being serviced.

Internal circuitry determines the priorities of the bits set in the IRR. The highest priority is selected and strobed into the corresponding bit of the ISR during Interrupt Acknowledge Cycles.

The Interrupt Mask Register (IMR) stores the bits which mask the incoming interrupt lines. The IMR operates on the IRR. Masking of a higher priority input will not affect the interrupt request lines of lower priority inputs.

9.2 Interrupt Sequence

The powerful features of the Interrupt Controller in a microcomputer system are its programmability and the interrupt routine addressing capability. The latter allows direct or indirect jumping to the specific interrupt routine requested without any polling of the interrupting devices. The following shows the interrupt sequence for an x86 type system (the 8080 mode of the interrupt controller must never be selected when programming the ESC).

Note that externally, the interrupt acknowledge cycle sequence appears different than in a traditional discrete 82C59 implementation. However, the traditional interrupt acknowledge sequence is generated within the ESC and it is an EISA compatible implementation.

1. One or more of the Interrupt Request (IRQ[x]) lines are raised high, setting the corresponding IRR bit(s).
2. The Interrupt Controller evaluates these requests, and sends an INT to the CPU, if appropriate.
3. The CPU acknowledges the INT and responds with an interrupt acknowledge cycle. This cycle is translated into a PCI bus command. This PCI command is broadcast over the PCI bus as a single cycle as opposed to the two cycle method typically used.
4. Upon receiving an interrupt acknowledge cycle from the CPU over the PCI, the PCEB converts the single cycle into an INTA# pulse to the ESC. The ESC uses the INTA# pulse to generate the two cycles that the internal 8259 pair can respond to with the expected interrupt vector. The cycle conversion is performed by a functional block in the ESC Interrupt Controller Unit. The internally generated interrupt acknowledge cycle is completed as soon as possible as the PCI bus is held in wait states until the interrupt vector data is returned. Each cycle appears as an interrupt acknowledge pulse on the INTA# pin of the cascaded interrupt controllers. These two pulses are not observable at the ESC periphery.
5. Upon receiving the first internally generated interrupt acknowledge, the highest priority ISR bit is set and the corresponding IRR bit is reset. The Interrupt Controller does not drive the Data Bus during this cycle. On the trailing edge of the first cycle pulse, a slave identification code is broadcast by the master to the slave on a private, internal three bit wide bus. The slave controller uses these bits to determine if it must respond with an interrupt vector during the second INTA# cycle.
6. Upon receiving the second internally generated interrupt acknowledge, the Interrupt Controller releases an 8-bit pointer (the interrupt vector) onto the Data Bus where it is read by the CPU.
7. This completes the interrupt cycle. In the AEOI mode the ISR bit is reset at the end of the second interrupt acknowledge cycle pulse. Otherwise, the ISR bit remains set until an appropriate EOI command is issued at the end of the interrupt subroutine.

If no interrupt request is present at step four of either sequence (i.e., the request was too short in duration) the Interrupt Controller will issue an interrupt level 7.

9.3 80x86 Mode

When initializing the control registers of the 82C59, an option exists in Initialization Control Word Four (ICW4) to select either an 80x86 or an MSC-85 microprocessor based system. The interrupt acknowledge cycle is different in an MSC-85 based system than in the 80x86 based system: the interrupt acknowledge takes three INTA# pulses with the MSC-85, rather than the two pulses with the 80x86. The ESC is used only in an 80x86 based system. You must program each interrupt controller's ICW4 bit-0 to a "1" to indicate that the interrupt controller is operating in an 80x86 based system. This setting ensures proper operation during an interrupt acknowledge.

9.4 ESC Interrupt Acknowledge Cycle

As discussed, the CPU generates an interrupt acknowledge cycle that is translated into a single PCI command and broadcast across the PCI bus to the PCEB. The PCEB pulses the INTA# signal to the ESC. The ESC Interrupt Unit translates the INTA# signal into the two INTA# pulses expected by the interrupt controller subsystem. The Interrupt Controller uses the first interrupt acknowledge cycle to internally freeze the state of the interrupts for priority resolution. The first controller (CNTRL-1), as a master, issues a three bit interrupt code on the cascade lines to CNTRL-2 (internal to the ESC) at the end of the INTA# pulse. On this first cycle the interrupt controller block does not issue any data to the processor and leaves its data bus buffers disabled. CNTRL-2 decodes the information on the cascade lines, compares the code to the byte stored in Initialization Command Word Three (ICW3), and determines if it will have to broadcast the interrupt vector during the second interrupt acknowledge cycle. On the second interrupt acknowledge cycle, the master (CNTRL-1) or slave (CNTRL-2), will send a byte of data to the processor with the acknowledged interrupt code composed as follows:

Table 9-3. Content of Interrupt Vector Byte for 80x86 System Mode

	D7	D6	D5	D4	D3	D2	D1	D0
IRQ7, 15	T7	T6	T5	T4	T3	1	1	1
IRQ6, 14	T7	T6	T5	T4	T3	1	1	0
IRQ5, 13	T7	T6	T5	T4	T3	1	0	1
IRQ4, 12	T7	T6	T5	T4	T3	1	0	0
IRQ3, 11	T7	T6	T5	T4	T3	0	1	1
IRQ2, 10	T7	T6	T5	T4	T3	0	1	0
IRQ1, 9	T7	T6	T5	T4	T3	0	0	1
IRQ0, 8	T7	T6	T5	T4	T3	0	0	0

NOTE:
T7-T3 represent the interrupt vector address (refer to Section 3.7, ICW2 register description).

The byte of data released by the interrupt unit onto the data bus is referred to as the "interrupt vector". The format for this data is illustrated on a per-interrupt basis in Table 9-3.

9.5 Programming the Interrupt Controller

The Interrupt Controller accepts two types of command words generated by the CPU or bus master:

1. Initialization Command Words (ICWs): Before normal operation can begin, each Interrupt Controller in the system must be initialized. In the 82C59, this is a two to four byte sequence. However, for the ESC, each controller must be initialized with a four byte sequence. This four byte sequence is required to configure the interrupt controller correctly for the ESC implementation. This implementation is EISA-compatible.

The four initialization command words are referred to by their acronyms: ICW1, ICW2, ICW3, and ICW4.

The base address for each interrupt controller is a fixed location in the I/O memory space, at 0020h for CNTRL-1 and at 00A0h for CNTRL-2.

An I/O write to the CNTRL-1 or CNTRL-2 base address with data bit 4 equal to 1 is interpreted as ICW1. For ESC-based EISA systems, three I/O writes to "base address + 1" (021h for CNTRL-1 and 0A0h for CNTRL-2) must follow the ICW1. The first write to "base address + 1" (021h/0A0h) performs ICW2, the second write performs ICW3, and the third write performs ICW4.

ICW1 starts the initialization sequence during which the following automatically occur:

1. Following initialization, an interrupt request (IRQ) input must make a low-to-high transition to generate an interrupt.
2. The Interrupt Mask Register is cleared.
3. IRQ7 input is assigned priority 7.
4. The slave mode address is set to 7.
5. Special Mask Mode is cleared and Status Read is set to IRR.

ICW2 is programmed to provide bits [7:3] of the interrupt vector that will be released onto the data bus by the interrupt controller during an interrupt acknowledge. A different base [7:3] is selected for each interrupt controller. Suggested values for a typical EISA system are listed in Table 9-4.

ICW3 is programmed differently for CNTRL-1 and CNTRL-2, and has a different meaning for each controller.

For CNTRL-1, the master controller, ICW3 is used to indicate which IRQx input line is used to cascade CNTRL-2, the slave controller. Within the ESC interrupt unit, IRQ2 on CNTRL-1 is used to cascade the INT output of CNTRL-2. Consequently, bit-2 of ICW3 on CNTRL-1 is set to a 1, and the other bits are set to 0's.

For CNTRL-2, ICW3 is the slave identification code used during an interrupt acknowledge cycle. CNTRL-1 broadcasts a code to CNTRL-2 over three internal cascade lines if an IRQ[x] line from CNTRL-2 won the priority arbitration on the master controller and was granted an interrupt acknowledge by the CPU. CNTRL-2 compares this identification code to the value stored in ICW3, and if the code is equal to bits [2:0] of ICW3, CNTRL-2 assumes responsibility for broadcasting the interrupt vector during the second interrupt acknowledge cycle pulse.

ICW4 must be programmed on both controllers. At the very least, bit 0 must be set to a 1 to indicate that the controllers are operating in an 80x86 system.

2. Operation Command Words (OCWs): These are the command words which dynamically reprogram the Interrupt Controller to operate in various interrupt modes.

Any interrupt lines can be masked by writing an OCW1. A 1 written in any bit of this command word will mask incoming interrupt requests on the corresponding IRQx line.

OCW2 is used to control the rotation of interrupt priorities when operating in the rotating priority mode and to control the End of Interrupt (EOI) function of the controller.

OCW3 is used to set up reads of the ISR and IRR, to enable or disable the Special Mask Mode (SMM), and to set up the interrupt controller in polled interrupt mode.

The OCWs can be written into the Interrupt Controller any time after initialization. Table 9-4 shows an example of typical values programmed by the BIOS at power-up for the ESC interrupt controller.

Table 9-4. Suggested Default Values for Interrupt Controller Registers

Port	Value	Description of Contents
020h	11h	CNTLR-1, ICW1
021h	08h	CNTLR-1, ICW2 Vector Address for 000020h
021h	04h	CNTLR-1, ICW3 Indicates Slave Connection
021h	01h	CNTLR-1, ICW3 ICW4 8086 Mode
021h	B8h	CNTLR-1, Interrupt Mask (may vary)
4D0h	00h	CNTLR-1, Edge/Level Control Register
0A0h	11h	CNTLR-2, ICW1
0A1h	70h	CNTLR-2, ICW2 Vector Address for 0001C0h
0A1h	02h	CNTLR-2, ICW3 Indicates Slave ID
0A1h	01h	CNTL4-2, ICW4 8086 Mode
4D1h	00h	CNTLR-2, Edge/Level Control Register
0A1h	BDh	CNTLR-2, Interrupt Mask (may vary)

The following flow chart illustrates the sequence software must follow to load the interrupt controller Initialization Command Words (ICWs). The sequence must be executed for CNTRL-1 and CNTRL-2. After writing ICW1, ICW2, ICW3, and ICW4 must be written in order. Any divergence from this sequence, such as an attempt to program an OCW, will result in improper initialization of the interrupt controller and unexpected, erratic system behavior. It is suggested that CNTRL-2 be initialized first, followed by CNTRL-1.

In the ESC, it is required that all four Initialization Command Words (ICWs) be initialized. Also, as shown in Figure 9-3, all ICWs must be programmed prior to programming the OCWs.

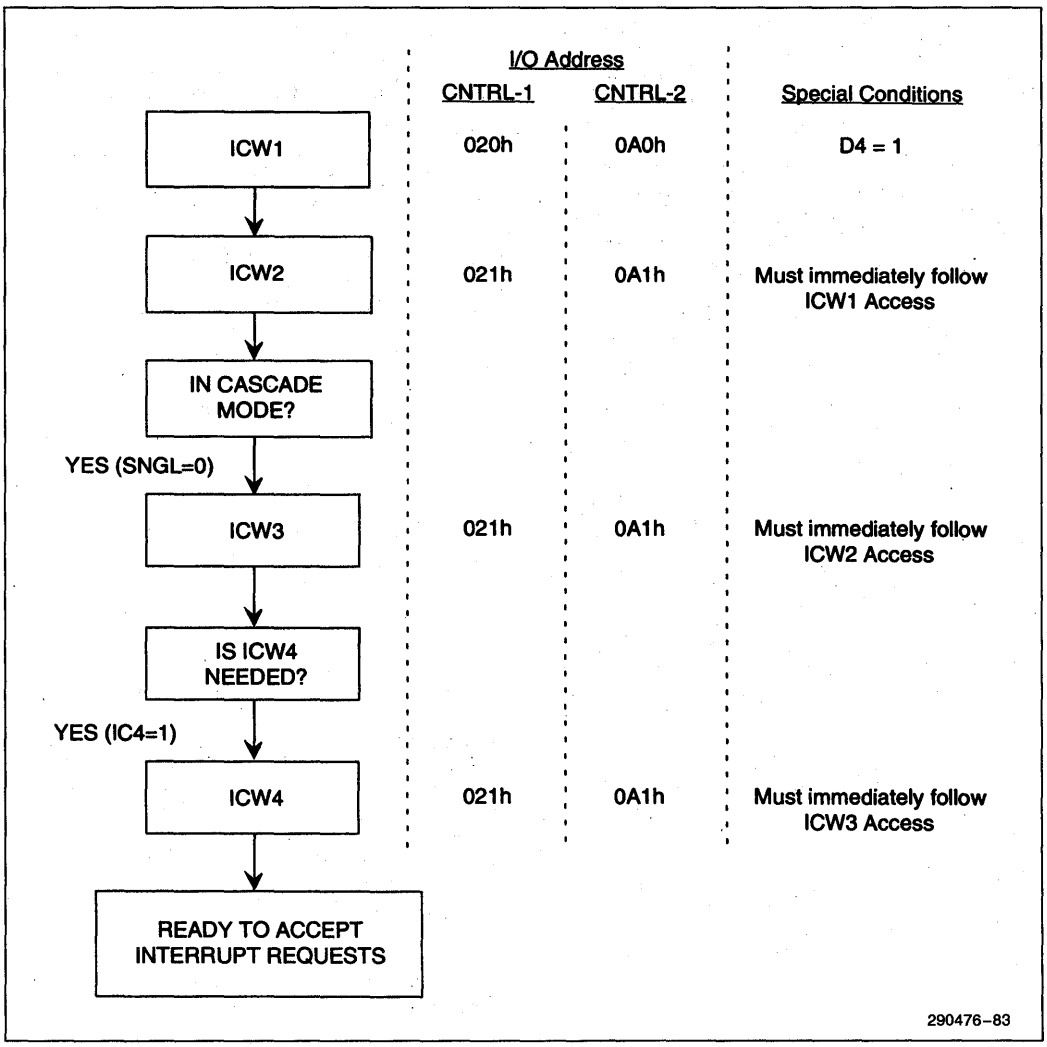


Figure 9-2. Initialization Sequence for ESC Initialization Command Words (ICWs)

9.6 End-of-Interrupt Operation

9.6.1 END OF INTERRUPT (EOI)

The In Service (IS) bit can be reset either automatically following the trailing edge of the second internal INTA# pulse (when AEOI bit in ICW1 is set) or by a command word that must be issued to the Interrupt Controller before returning from a service routine (EOI command). An EOI command must be issued twice with this cascaded interrupt controller configuration, once for the master and once for the slave.

There are two forms of EOI commands: Specific and Non-Specific. When the Interrupt Controller is operated in modes which preserve the fully nested structure, it can determine which IS bit to reset on EOI. When a Non-Specific EOI command is issued the Interrupt Controller will automatically reset the highest IS bit of those that are set, since in the fully nested mode the highest IS level was necessarily the last level acknowledged and serviced. A non-specific EOI can be issued with OCW2 (EOI = 1, SL = 0, R = 0).

When a mode is used which may disturb the fully nested structure, the Interrupt Controller may no longer be able to determine the last level acknowledged. In this case a Specific End of Interrupt must be issued which includes as part of the command the IS level to be reset. A specific EOI can be issued with OCW2 (EOI = 1, SL = 1, R = 0, and LO–L2 is the binary level of the IS bit to be reset).

It should be noted that an IS bit that is masked by an IMR bit will not be cleared by a non-specific EOI if the Interrupt Controller is in the Special Mask Mode.

9.6.2 AUTOMATIC END OF INTERRUPT (AEOI) MODE

If AEOI = 1 in ICW4, then the Interrupt Controller will operate in AEOI mode continuously until reprogrammed by ICW4. Note that reprogramming ICW4 implies that ICW1, ICW2, and ICW3 must be reprogrammed first, in sequence. In this mode the Interrupt Controller will automatically perform a non-specific EOI operation at the trailing edge of the last interrupt acknowledge pulse. Note that from a system standpoint, this mode should be used only when a nested multilevel interrupt structure is not required within a single Interrupt Controller. The AEOI mode can only be used in a master Interrupt Controller and not a slave (on CNTRL-1 but not CNTRL-2).

9.7 Modes of Operation

9.7.1 FULLY NESTED MODE

This mode is entered after initialization unless another mode is programmed. The interrupt requests are ordered in priority from 0 through 7 (0 being the highest). When an interrupt is acknowledged the highest priority request is determined and its vector placed on the bus. Additionally, a bit of the Interrupt Service register (IS[0:7]) is set. This IS bit remains set until the microprocessor issues an End of Interrupt (EOI) command immediately before returning from the service routine. Or, if the AEOI (Automatic End of Interrupt) bit is set, this IS bit remains set until the trailing edge of the second internal INTA#. While the IS bit is set, all further interrupts of the same or lower priority are inhibited, while higher levels will generate an interrupt (which will be acknowledged only if the microprocessor internal Interrupt enable flip-flop has been re-enabled through software).

After the initialization sequence, IRQ0 has the highest priority and IRQ7 the lowest. Priorities can be changed, as will be explained, in the rotating priority mode.

9.7.2 THE SPECIAL FULLY NESTED MODE

This mode will be used in the case of a big system where cascading is used, and the priority has to be conserved within each slave. In this case the special fully nested mode will be programmed to the master (using ICW4). This mode is similar to the normal nested mode with the following exceptions:

- When an interrupt request from a certain slave is in service, this slave is not locked out from the master's priority logic and further interrupt requests from higher priority IRQ's within the slave will be recognized by the master and will initiate interrupts to the processor. (In the normal nested mode a slave is masked out when its request is in service and no higher requests from the same slave can be serviced.)
- When exiting the Interrupt Service routine the software has to check whether the interrupt serviced was the only one from that slave. This is done by sending a non-specific End of Interrupt (EOI) command to the slave and then reading its In-Service register and checking for zero. If it is empty, a non-specific EOI can be sent to the master too. If not, no EOI should be sent.

9.7.3 AUTOMATIC ROTATION (EQUAL PRIORITY DEVICES)

In some applications there are a number of interrupting devices of equal priority. Automatic rotation mode provides for a sequential 8-way rotation. In this mode a device receives the lowest priority after being serviced. In the worst case, a device requesting an interrupt will have to wait until each of seven other devices are serviced at most once. Figure 9-3 shows an example of automatic rotation.

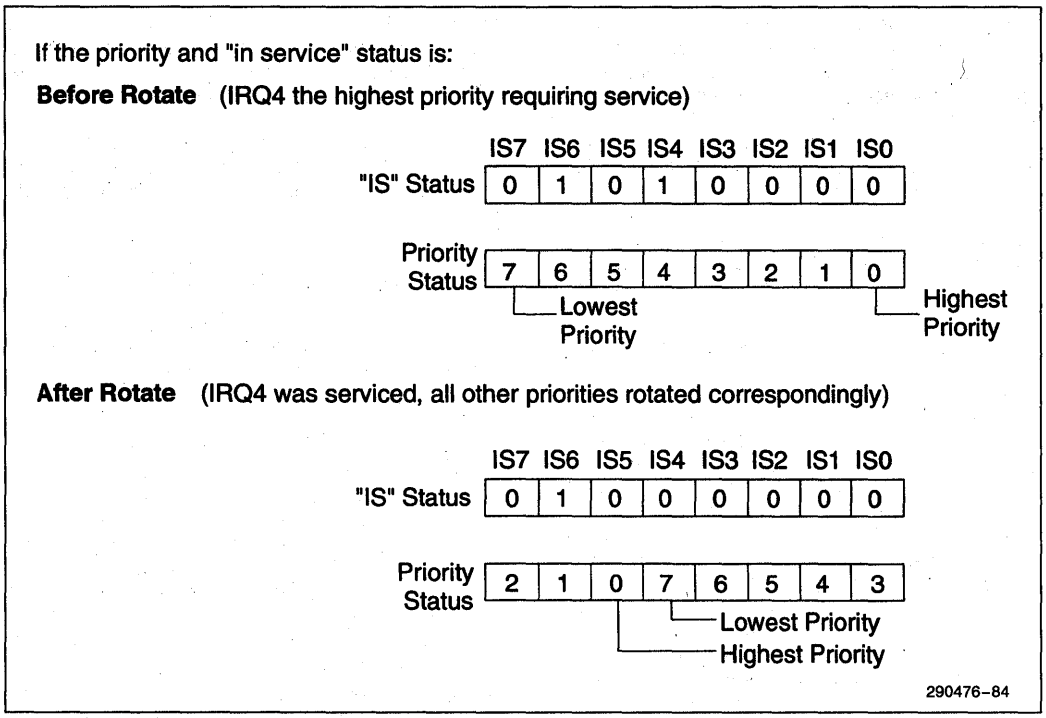


Figure 9-3. Automatic Rotation Mode Example

There are two ways to accomplish Automatic Rotation using OCW2, the Rotation on Non-Specific EOI Command (R = 1, SL = 0, EOI = 1) and the Rotate in Automatic EOI Mode which is set by (R = 1, SL = 0, EOI = 0) and cleared by (R = 0, SL = 0, EOI = 0).

9.7.4 SPECIFIC ROTATION (SPECIFIC PRIORITY)

The programmer can change priorities by programming the bottom priority and thus fixing all other priorities. For example, if IRQ5 is programmed as the bottom priority device, then IRQ6 will be the highest priority device.

The Set Priority command is issued in OCW2 where: R = 1, SL = 1; LO–L2 is the binary priority level code of the bottom priority device. See the register description for the bit definitions.

Note that in this mode internal status is updated by software control during OCW2. However, it is independent of the End of Interrupt (EOI) command (also executed by OCW2). Priority changes can be executed during an EOI command by using the Rotate on Specific EOI command in OCW2 (R = 1, SL = 1, EOI = 1 and LO–L2 = IRQ level to receive bottom priority).

9.7.5 POLL COMMAND

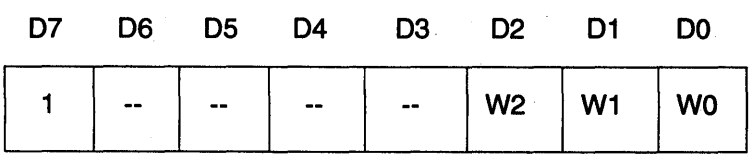
The Polled Mode can be used to conserve space in the interrupt vector table. Multiple interrupts that can be serviced by one interrupt service routine do not need separate vectors if the service routine uses the poll command.

The Polled Mode can also be used to expand the number of interrupts. The polling interrupt service routine can call the appropriate service routine, instead of providing the interrupt vectors in the vector table.

In this mode the INT output is not used and the microprocessor internal Interrupt Enable flip-flop is reset, disabling its interrupt input. Service to devices is achieved by software using a Poll command.

The Poll command is issued by setting P = 1 in OCW3. The Interrupt Controller treats the next I/O read pulse to the Interrupt Controller as an interrupt acknowledge, sets the appropriate IS bit if there is a request, and reads the priority level. Interrupts are frozen from the I/O write to the I/O read.

The word enabled onto the data bus during I/O read is:



NOTES:
W0–W2: Binary code of the highest priority level requesting service.
Bit D7–1: Equal to "1" if there is an interrupt.

Figure 9-4. Polled Mode

This mode is useful if there is a routine command common to several levels so that the INTA# sequence is not needed (saves ROM space).

9.7.6 CASCADE MODE

The Interrupt Controllers in the ESC system are interconnected in a cascade configuration with one master and one slave. This configuration can handle up to 15 separate priority levels.

The master controls the slaves through a three line internal cascade bus. When the master drives 010b on the cascade bus, this bus acts like a chip select to the slave controller.

In a cascade configuration, the slave interrupt outputs are connected to the master interrupt request inputs. When a slave request line is activated and afterwards acknowledged, the master will enable the corresponding slave to release the interrupt vector address during the second INTA# cycle of the interrupt acknowledge sequence.

Each Interrupt Controller in the cascaded system must follow a separate initialization sequence and can be programmed to work in a different mode. An EOI command must be issued twice: once for the master and once for the slave.

9.7.7 EDGE AND LEVEL TRIGGERED MODES

There are two ELCR registers one for each 82C59 bank. They are located at I/O ports 04D0h (for the Master Bank, IRQ[0:1,3:7]) and 04D1h (for the Slave Bank, IRQ[8#:15]). They allow the edge and level sense selection to be made on an interrupt by interrupt basis instead of on a complete bank. Only the interrupts that connect to the EISA bus may be programmed for level sensitivity. That is IRQ (0,1,2,8#,13) must be programmed for edge sensitive operation. The LTIM bit is disabled in the ESC. The default programming is equivalent to programming the LTIM bit (ICW1 bit 3) to a 0.

If an ELCR bit is equal to "0", an interrupt request will be recognized by a low to high transition on the corresponding IRQ input. The IRQ input can remain high without generating another interrupt.

If an ELCR bit is equal to "1", an interrupt request will be recognized by a "low" level on the corresponding IRQ input, and there is no need for an edge detection. For level triggered interrupt mode, the interrupt request signal must be removed before the EOI command is issued or the CPU interrupt must be disabled. This is necessary to prevent a second interrupt from occurring.

In both the edge and level triggered modes the IRQ inputs must remain active until after the falling edge of the first INTA#. If the IRQ input goes inactive before this time a DEFAULT IRQ7 will occur when the CPU acknowledges the interrupt. This can be a useful safeguard for detecting interrupts caused by spurious noise glitches on the IRQ inputs. To implement this feature the IRQ7 routine is used for "clean up" simply executing a return instruction, thus ignoring the interrupt. If IRQ7 is needed for other purposes a default IRQ7 can still be detected by reading the ISR. A normal IRQ7 interrupt will set the corresponding ISR bit, a default IRQ7 won't. If a default IRQ7 routine occurs during a normal IRQ7 routine, however, the ISR will remain set. In this case it is necessary to keep track of whether or not the IRQ7 routine was previously entered. If another IRQ7 occurs it is a default.

IRQ13 still appears externally to be an edge sensitive interrupt even though it is shared internally with the Chaining interrupt. The Chaining interrupt is ORed after the edge sense logic.

9.8 Register Functionality

For a detailed description of the Interrupt Controller register set, please see Section 3.4, Interrupt Unit Register description.

9.8.1 INITIALIZATION COMMAND WORDS

Four initialization command words (ICWs) are used to initialize each interrupt controller. Each controller is initialized separately. Following this initialization sequence, the interrupt controller is ready to accept interrupts.

9.8.2 OPERATION CONTROL WORDS (OCWS)

After the Initialization Command Words (ICWs) are programmed into the Interrupt Controller, the chip is ready to accept interrupt requests at its input lines. However, Interrupt Controller operation can be dynamically modified to fit specific software/hardware expectations. Different modes of operation are dynamically selected following initialization through the use of Operation Command Words (OCWs).

9.9 Interrupt Masks

9.9.1 MASKING ON AN INDIVIDUAL INTERRUPT REQUEST BASIS

Each Interrupt Request input can be masked individually by the Interrupt Mask register (IMR). This register is programmed through OCW1. Each bit in the IMR masks one interrupt channel if it is set to a "1". Bit 0 masks IRQ0, Bit 1 masks IRQ1 and so forth. Masking an IRQ channel does not affect the other channel's operation, with one notable exception. Masking IRQ[2] on CNTRL-1 will mask off all requests for service from CNTRL-2. The CNTRL-2 INT output is physically connected to the CNTRL-1 IRQ[2] input.

9.9.2 SPECIAL MASK MODE

Some applications may require an interrupt service routine to dynamically alter the system priority structure during its execution under software control. For example, the routine may wish to inhibit lower priority requests for a portion of its execution but enable some of them for another portion.

The difficulty here is that if an Interrupt Request is acknowledged and an End of Interrupt command did not reset its IS bit (i.e., while executing a service routine), the Interrupt Controller would have inhibited all lower priority requests with no easy way for the routine to enable them.

The Special Mask Mode enables all interrupts not masked by a bit set in the Mask Register. Interrupt service routines that require dynamic alteration of interrupt priorities can take advantage of the Special Mask Mode. For example, a service routine can inhibit lower priority requests during a part of the interrupt service, then enable some of them during another part.

In the Special Mask Mode, when a mask bit is set in OCW1, it inhibits further interrupts at that level and enables interrupts from all other levels (lower as well as higher) that are not masked.

Thus, any interrupts may be selectively enabled by loading the Mask register with the appropriate pattern.

Without Special Mask Mode, if an interrupt service routine acknowledges an interrupt without issuing an EOI to clear the IS bit, the interrupt controller inhibits all lower priority requests. The Special Mask Mode provides an easy way for the interrupt service routine to selectively enable only the interrupts needed by loading the Mask register.

The special Mask Mode is set by OCW3 where: SSMM = 1, SMM = 1, and cleared where SSMM = 1, SMM = 0.

9.10 Reading the Interrupt Controller Status

The input status of several internal registers can be read to update the user information on the system. The Interrupt Request Register (IRR) and In-Service Register (ISR) can be read via OCW3, as discussed in Section 3.7. The Interrupt Mask Register (IMR) is read via a read of OCW1, as discussed in Section 3.7. Here are brief descriptions of the ISR, the IRR, and the IMR.

Interrupt Request Register (IRR): 8-bit register which contains the status of each interrupt request line. Bits that are clear indicate interrupts that have not requested service. The Interrupt Controller clears the IRR's highest priority bit during an interrupt acknowledge cycle. (Not affected by IMR).

In-Service Register (ISR): 8-bit register indicating the priority levels currently receiving service. Bits that are set indicate interrupts that have been acknowledged and their interrupt service routine started. Bits that are cleared indicate interrupt requests that have not been acknowledged, or interrupt request lines that have not been asserted. Only the highest priority interrupt service routine executes at any time. The lower priority interrupt services are suspended while higher priority interrupts are serviced. The ISR is updated when an End of Interrupt Command is issued.

Interrupt Mask Register (IMR): 8-bit register indicating which interrupt request lines are masked.

The IRR can be read when, prior to the I/O read cycle, a Read Register Command is issued with OCW3 (RR = 1, RIS = 0).

The ISR can be read when, prior to the I/O read cycle, a Read Register Command is issued with OCW3 (RR = 1, RIS = 1).

The interrupt controller retains the ISR/IRR status read selection following each write to OCW3. Therefore, there is no need to write an OCW3 before every status read operation, as long as the current status read corresponds to the previously selected register. For example, if the ISR is selected for status read by an OCW3 write, the ISR can be read over and over again without writing to OCW3 again. However, to read the IRR, OCW3 will have to be reprogrammed for this status read prior to the OCW3 read to check the IRR. This is not true when poll mode is used. Polling Mode overrides status read when P = 1, RR = 1 in OCW3.

After initialization the Interrupt Controller is set to read the IRR.

As stated, OCW1 is used for reading the IMR. The output data bus will contain the IMR status whenever I/O read is active the address is 021h or 061h (OCW1).

9.11 Non-Maskable Interrupt (NMI)

An NMI is an interrupt requiring immediate attention and has priority over the normal interrupt lines (IRQx). The ESC indicates error conditions by generating a non-maskable interrupt.

The ESC generates NMI interrupts based on the following Hardware and Software events.

Hardware Events:

1. Mother Board Parity Errors.

   Memory parity errors for the mother board memory. These errors are reported to the ESC through the PERR# signal line.

2. System Errors.

   System error on the mother board. The system board uses the SERR# signal to indicate system errors to the ESC.

3. Add-In Board Parity Errors.

   Parity errors on the add-in memory boards on the EISA expansion bus. IOCHK# signal on the EISA bus is driven low by the add-in board logic when this error occurs.

4. Fail-Safe Timer Timeout.

   Fail-Safe Timer (Timer 2, Counter 0) count expires. If this counter has been set and enabled, and the count expires before a software routine can reset the counter.

5. Bus Timeout.

   An EISA bus Master or Slave exceeds the allocated time on the bus. A bus timeout occurs if an EISA Master does not relinquish the bus (MREQ# negated) within 64 BCLKS after it has been preempted (MACK# negated). A bus timeout also occurs if a memory slave extends the cycle (CHRDY negated) long enough to keep CMD# asserted for more than 256 BCLKS. The DMA controller does not cause a bus timeout. The ESC asserts RESDRV when a bus timeout occurs.

Software Events:

1. Software Generated NMI.

   If an I/O Write access to Port 0462h occurs. The data value for this write is a don't care.

The NMI logic incorporates four different 8-bit registers. These registers are used by the CPU to determine the source of the interrupt and to enable or disable/clear the interrupts. Please see the Section 3.4, Interrupt Unit Register description, for the register details.

Table 9-5. NMI Register I/O Address Map

I/O Port Address	Register Description
0061h	NMI Status Register
0070h	NMI Enable Register
0461h	Extended NMI Register
0462h	Software NMI Register

Table 9-6. NMI Source Enable/Disable and Status Port Bits

NMI Source	I/O Port Bit for Status Reads	I/O Port Bit for Enable/Disable
PERR#	Port 0061h, Bit 7	Port 0061h, Bit 2
IOCHK#	Port 0061h, Bit 6	Port 0061h, Bit 3
Fail-Safe	Port 0461h, Bit 7	Port 0461h, Bit 2
Bus Timeout	Port 0461h, Bit 6	Port 0461h, Bit 3
Write to Port 0462h	Port 0461h, Bit 5	Port 0461h, Bit 1

The individual enable/disable bits clear the NMI detect flip-flops when disabled.

All NMI sources can be enabled or disabled by setting Port 070h bit [7]. This disable function does not clear the NMI detect Flip-Flops. This means, if NMI is disabled then enabled via Port 070h, then an NMI will occur when Port 070h is re-enabled if one of the NMI detect Flip-Flops had been previously set.

To ensure that all NMI requests are serviced, the NMI service routine software needs to incorporate a few very specific requirements. These requirements are due to the edge detect circuitry of the host microprocessor, 80386 or 80486. The software flow would need to be the following:

1. NMI is detected by the processor on the rising edge of the NMI input.
2. The processor will read the status stored in port 061h and 0461h to determine what sources caused the NMI. The processor may then reset the register bits controlling the sources that it has determined to be active. Between the time the processor reads the NMI sources and resets them, an NMI may have been generated by another source. The level of NMI will then remain active. This new NMI source will not be recognized by the processor because there was no edge on NMI.
3. The processor must then disable all NMI's by writing bit [7] of port 070H high and then enable all NMI's by writing bit [7] of port 070H low. This will cause the NMI output to transition low then high if there are any pending NMI sources. The CPU's NMI input logic will then register a new NMI.

10.0 PCEB/ESC INTERFACE

The PCEB/ESC interface (Figure 10-1) 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 buffer), and interrupt acknowledge cycles.

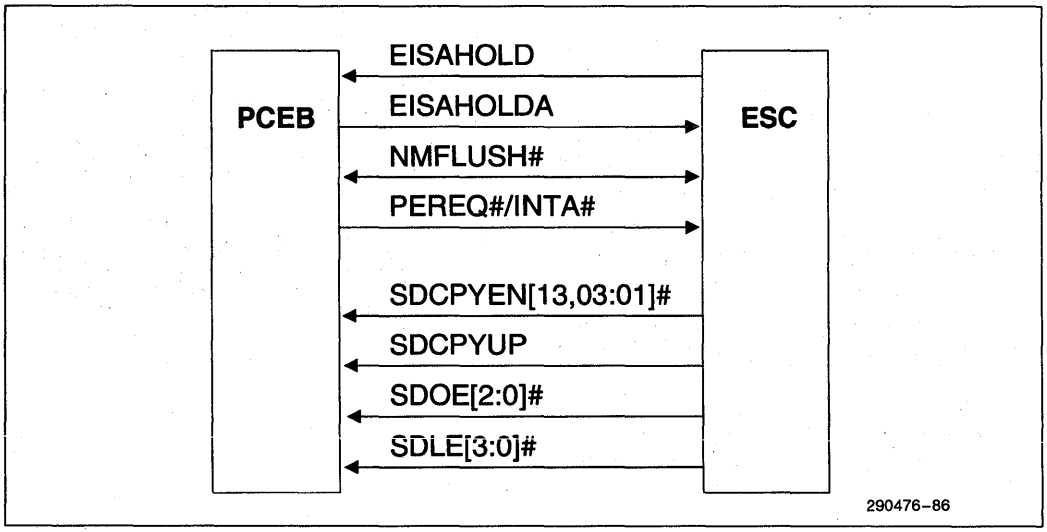


Figure 10-1. PCEB/ESC Interface Signals

10.1 Arbitration Control Signals

The PCEB contains the arbitration circuitry for the PCI Local 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 Local Bus. The assertion of NMFLUSH# also instructs the PCEB to initiate flushing and to temporarily disable system buffers on the PCI Local 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 Local Bus. Buffer flushing also prevents dead-lock conditions between the PCI Local Bus and EISA Bus. Since the ESC/PCEB does not know ahead of time, whether the new master is going to access the PCI Local Bus or a device on the EISA Bus, buffers pointing to the PCI Local 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 Local 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.

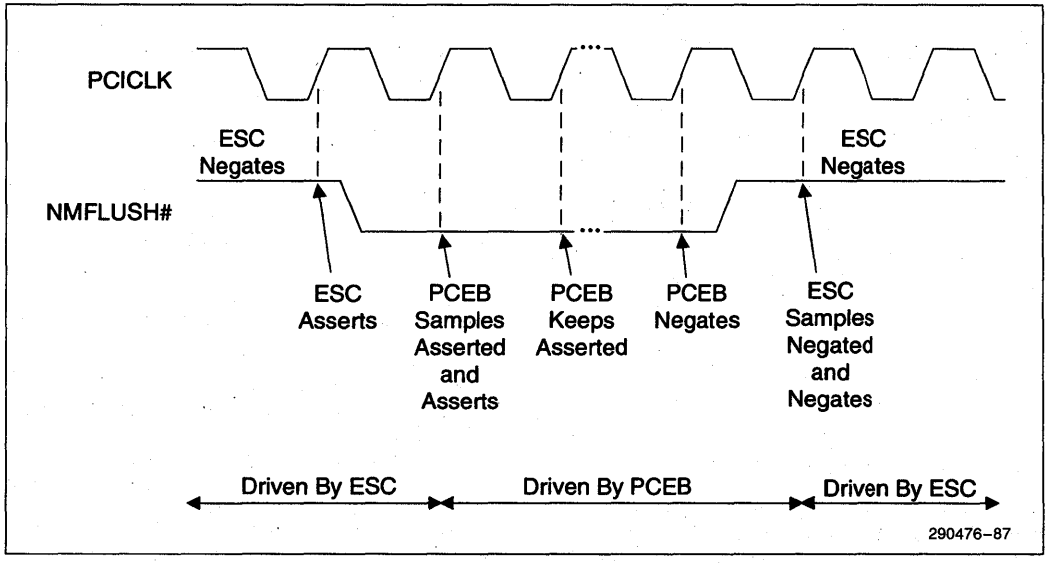


Figure 10-2. NMFLUSH# Protocol

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 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 the PCEB data sheet. 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 are inputs to the PCEB.

- SDCPYEN[13,3:1]#
- SDCPYUP
- SDOE[2:0]#
- SDLE[3:0]#

Copy Enable Outputs (SDCPYEN[13,3:1]#)

These signals enable the byte copy operations between data byte lanes 0, 1, 2 and 3 as shown in the Table 10-1. ISA master cycles do not perform assembly/disassembly operations. Thus, these cycles use SDCPYEN[13,3:1]# to perform the byte routing and byte copying between lanes. EISA master cycles however, can have assembly/disassembly operations. These cycles use SDCPYEN[13,3:1]# in conjunction with SDCPYUP and SDLE[3:0]#.

Table 10-1. 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 10-2). 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 10-2. 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 Local 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.

10.3 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 INTEGRATED SUPPORT LOGIC

The ESC integrates support logic for assorted functions for a typical EISA system board. The following functions are directly supported by the ESC.

- EISA Address Buffer Control
- Coprocessor Interface
- BIOS Interface
- Keyboard Controller Interface
- Real Time Clock Interface
- Floppy Disk Controller Interface
- Configuration RAM Interface
- X-Bus and IDE Decode

11.1 EISA Address Buffer Control

The EISA Bus consists of unlatch addresses (LA[31:2]) and latched addresses (SA[19:2]). EISA devices generate or monitor LA addresses, and ISA devices generate or monitor SA addresses. Three Discrete F543s are used to generate the SA address from LA and LA addresses from SA addresses (see Figure 11-1). The ESC generates the control signals SALE#, LASAOE#, and SALAOE# for the F543s. These signals control the direction of the address flow. For EISA master, DMA, and Refresh cycles, the LA addresses are generated by the master device, and the SA addresses are driven by the F543s. For ISA master devices, the SA addresses are generated by the master device, and the LA addresses are driven by the F543s.

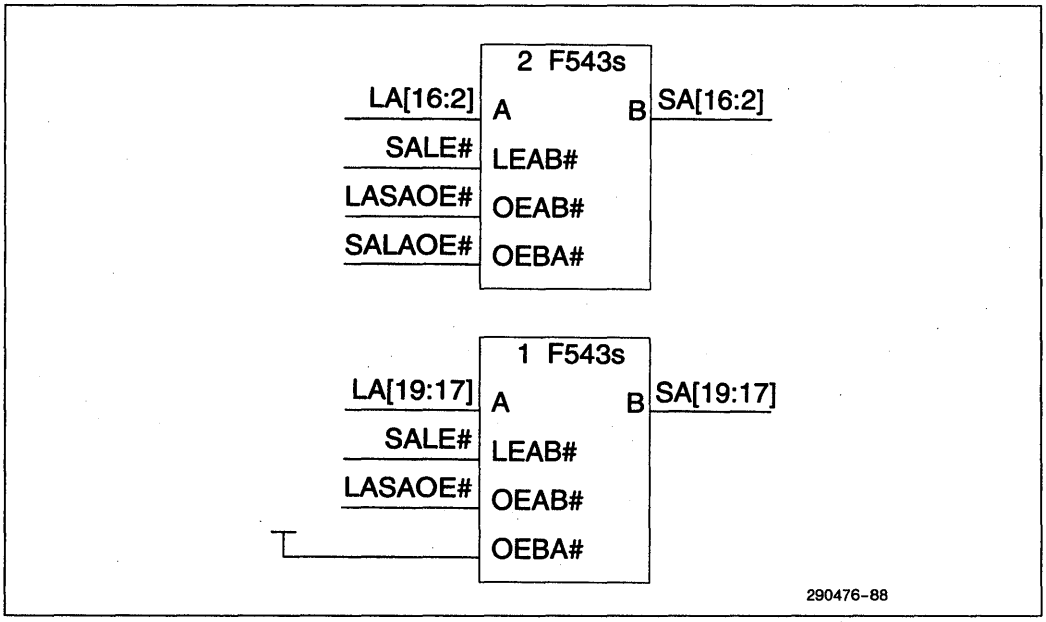


Figure 11-1. EISA Address Buffers

Table 11-1. EISA Address Buffer Control Function

Signal	Cycle Type			
	EISA Master	ISA Master	DMA	Refresh
SALE#	Pulses	Low	Pulses	Pulses
LASAOE#	Low	High	Low	Low
SALAOE#	High	Low	High	High

11.2 Coprocessor Interface

The numeric coprocessor interface is designed to support PC/AT compatible numeric coprocessor exception handling. The EISA Clock Divisor configuration register bit 5 needs to be set to a 1 in order to enable the coprocessor error support in the ESC. The coprocessor interface consists of FERR# signal and IGNNE# signal. The FERR# signal and IGNNE# signals are connected directly to the Floating Point Error pin and Ignore Floating Point Error pin of the CPU respectively.

Whenever an error during computation is detected, the CPU asserts the FERR# signal to the ESC. The ESC internally generates an interrupt on the IRQ13 line of the integrated Interrupt Controller. The result is a asserted INT signal to the CPU.

When the ESC detects an I/O write to the internal port 00F0h, the ESC deasserts the internal IRQ13 line to the integrated Interrupt Controller. At the same time the ESC asserts the IGNNE# signal. The ESC keeps the IGNNE# signal asserted until the FERR# signal is negated by the CPU.

If the coprocessor error support is enabled in the EISA Clock Divisor configuration register then the ESC IRQ13 pins cannot be used, and this pin should be tied to ground.

11.3 BIOS Interface

The ESC supports a total of 512 Kbytes of BIOS memory. The ESC asserts the LBIOSCS# signal for EISA or ISA memory cycles decoded to be in the BIOS space. The 512 Kbytes of BIOS includes the conventional 128 Kbytes of BIOS and 384 Kbytes of enlarged BIOS. The 128 Kbytes of conventional BIOS is divided into multiple regions. Each region can be independently enabled or disabled by setting the appropriate bits in the BIOS Chip Select A register and BIOS Chip Select B register. The 128 Kbytes of conventional BIOS is also aliased at different locations within the memory space. Refer to Section 4.1, BIOS Memory Space, for details.

The ESC generates the LBIOSCS# signal by internally latching the output of the BIOS address decode with BALE signal. The ESC asserts the LBIOSCS# for all read cycles in the enabled BIOS memory space. The ESC will assert LBIOSCS# signal for write cycles in the enabled BIOS memory space only if the BIOS Chip Select B register bit 3 is set to 1 (BIOS write enable).

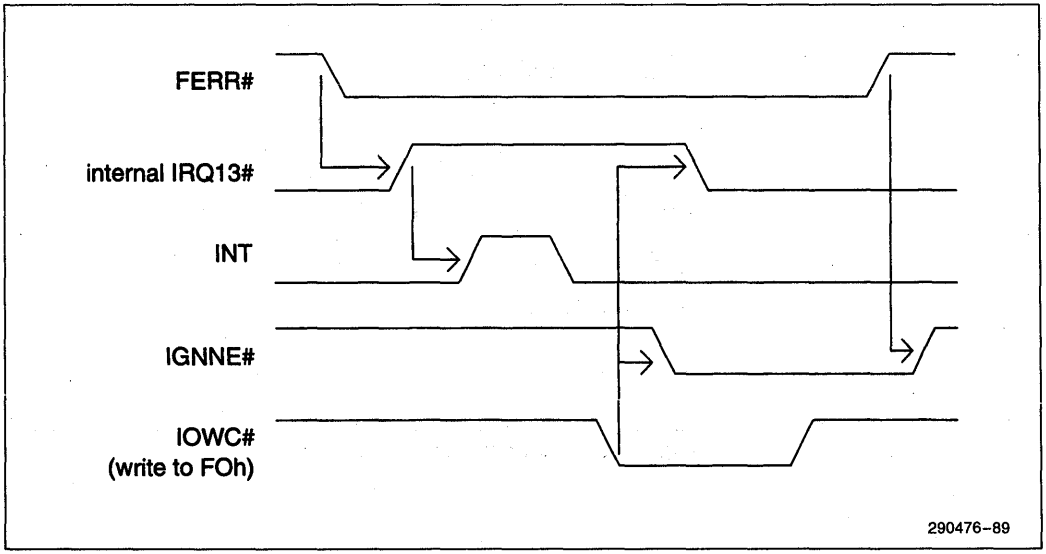


Figure 11-2. Coprocessor Interface Waveform

11.4 Keyboard Controller Interface

The ESC provides a complete inferface to a glueless interface to a 8x42 Keyboard Controller. The ESC Keyboard Controller interface consists of Keyboard Controller Chip Select (KYBDCS#) signal, Mouse interrupt (ABFULL) signal. The ESC also supports the fast Keyboard commands for CPU reset (ALTRST#) and address A20 enable (ALTA20) by integrating Port 92h.

The ESC asserts the KYBDCS# signal for I/O cycles to addresses 60h and 64h if the Peripheral Chip Select A register bit 1 is set to 1. The ESC uses the ABFULL signal to internally generate an interrupt request to the integrated Interrupt Controller on the IRQ12 line if EISA Clock Divisor register bit 4 is set to 1 (Mouse Interrupt Enable). A low to high transition on the ABFULL signal is internally latched by the ESC. The high level on this latch remains until a write to I/O port 60h is detected or the ESC is reset.

The ALTRST# is used to reset the CPU under software control. The ESC ALTRST# signal needs to be AND'ed externally with the reset signal from the keyboard controller. A write to the System Control register (092h) bit 0 to set the bit to a 1 from a 0 causes the ESC to pulse the ALTRST# signal. The ALTRST# signal is asserted for approximately 4 BCLKs. The ESC will not pulse the ALTRST# signal if bit 0 has previously been set to a 1.

11.5 Real Time Clock

The ESC provides a glueless interface for the Real Time Clock in the system. The ESC provides a Real Time Clock Address Latch Enable signal (RTCALE), a Real Time Clock read Strobe (RTCRD#), and a Real Time Clock Write strobe (RTCWR#). The ESC pulses the RTCALE signal asserted for one and a half BCLKs when an I/O write to address 70h is detected. The ESC asserts RTCRD# signal and RTCWR# signal for I/O read and write accesses to address 71h respectively.

The ESC also supports the power on password protection through the Real Time Clock. The power on password protection is enabled by setting the System Control register 092h bit 3 to a 1. The ESC does not assert RTCRD# signal or RTCWR# signal for I/O cycles to 71h if the accesses are addressed to Real Time Clock addresses (write to 70h) 36h to 3Fh if the power on password protection is enabled.

11.6 Floppy Disk Control Interface

The ESC support interface to the 82077(SL) floppy disk controller chip. The ESC provides a Floppy Disk Controller Chip Select signal (FDCCS#). The ESC also provides a buffered Drive Interface (DSKCHG#) signal. In addition, the ESC generates the control for the disk light.

The ESC supports both the primary address range (03F0h-03F7h) and secondary address range (0370h-0377h) of the Floppy Disk Controller. The state of Peripheral Chip Select A register bit 5 determines which address range is decoded by the ESC as access to Floppy Disk Controller. If bit 5 is set to 0, the ESC will decode the primary Floppy Disk Controller address range. If bit 5 is set to 1, the ESC will decode the secondary Floppy Disk Controller address range.

The ESC supports the Drive Interface signal. During I/O accesses to address 03F7h (primary) or 0377h (secondary), the ESC drives the inverted state of the DSKCHG# signal on to the SD7 data line. The ESC uses the DSKCHG# signal to determine if the Floppy Disk Controller is present on the X-Bus. If the DSKCHG# signal is sampled low during reset, the ESC will disable Floppy Disk Controller support.

The ESC also supports the Disk Light function by generating the DLIGHT# signal. If System Control 092h register bit 6 or bit 7 is set to a 1, the ESC will assert the DLIGHT# signal.

11.7 Configuration RAM Interface

The ESC provides the control signals for 8 Kbytes of external configuration RAM. The configuration RAM is used for storing EISA configuration system parameters. The configuration RAM is I/O mapped between location 0800h–08FFh. Due to the I/O address constraint (256 byte addresses for 8 Kbytes of RAM), the configuration RAM is organized in 32 pages of 256 bytes each. The I/O port 0C00h is used to store the configuration RAM page address. The ESC integrates this port as Configuration RAM Page register. During a read or a write to the configuration RAM address space 0800h-08FFh, the ESC drives the configuration RAM page address by placing the content of the Configuration RAM Page Address register bits[4:0] on the EISA Address line LA[31:27]#. The ESC will also assert the

CRAMRD# signal or the CRAMWR# signal for I/O read and write accesses to I/O address 0800h-08FFh. The ESC will only generate the configuration RAM page address and assert the CRAMRD# signal and CRAMWR# signal if the Peripheral Chip Select B register bit 7 is set to 1.

11.8 General Purpose Peripherals, IDE, Parallel Port, and Serial Port Interface

The ESC provides three dual function pins (GPCS[2:0]#, ECS[2:0]). The functionality of these pins is selected through the configuration Mode Select register bit 4. If Mode Select register bit 4 is set to 0 the general purpose chip select functionality is selected. If Mode Select register bit 4 is set to 1, the encoded chip select functionality is selected.

In general purpose chip select mode, the ESC generates three general purpose chip selects (GPCS[2:0]#). The decode for each general purpose chip selects is programmed through a set of three configuration registers; General Purpose Chip Select x Base Low Address register, General Purpose Chip Select x Base High Address register, and General Purpose Chip Select x Mask register. Each General Purpose Peripheral can be mapped anywhere in the 64 Kbytes of I/O address. The general purpose peripheral address range is programmable from 1 byte to 256 bytes with 2^n granuality.

In encoded chip select mode (ESC[2:0]), in addition to decoding the general purpose chip select 0 address and general purpose chip select 1 address, the ESC also decodes IDE, Parallel Ports, and Serial Ports addresses. The encoded chip select mode requires an external decoder like a F138 to generate the device chip selects from the ESC[2:0] signals.

The ESC generates encoded chip selects for two Serial Ports, COMACS# (ECS[2:0]=000) and COMBCS# (ESC[2:0]=001). The ESC supports Serial Port COM1 and Serial Port COM2. Accesses to Serial Port COM1 or Serial Port COM2 are individually programmed through Peripheral Chip Select B register bits [0:3] to generate an encoded chip select for COMACS# or COMBCS#.

Table 11-2. Encoded Chip Select Decode

ESC2	ESC1	ESC0	PERIPHERAL CS
0	0	0	COMACS#
0	0	1	COMBCS#
0	1	0	LPTCS#
0	1	1	IDECS0#
1	0	0	IDECS1#
1	0	1	GPCS0#
1	1	0	GPCS1#
1	1	1	Idle State

Refer to Section 4.5 for the address decode of the peripheral chip selects.

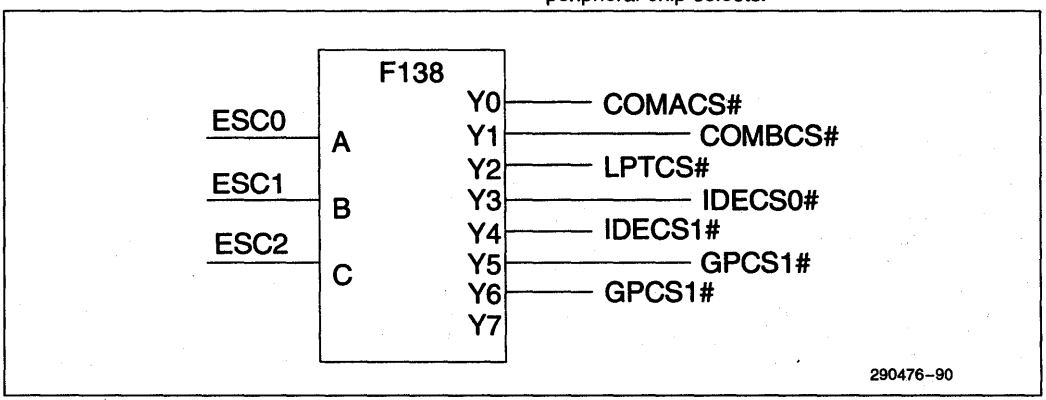


Figure 11-3. Encoded Chip Select Decoder Logic

11.9 X-Bus Control and General Purpose Decode

The X-Bus is a secondary data bus buffered from the EISA Bus. The X-Bus is used to interface with peripheral devices that do not require a high speed interface. Typically a discrete buffer device like a F245 is used to buffer the EISA Bus from the X-Bus. The ESC provides two control signals, XBUSTR# and XBUSOE#, for the discrete F245 buffer.

The XBUSTR# signal controls the direction of the data flow of the F245. When the XBUSTR# signal is high, the data direction of the F245 buffer is from the XD[7:0] bus to the SD[7:0] bus. The ESC drives the XBUSTR# signal high during EISA master I/O read cycles, ISA master I/O read cycles, DMA write cycles (write to memory), and memory read cycles decoded to be in the X-Bus BIOS address space. The ESC also drives the XBUSTR# signal high for DMA reads (reads from memory/writes to I/O) from the X-Bus BIOS address space. The X-Bus BIOS address space is defined as the enabled regions and enabled aliases of the BIOS memory space. See Section 4.1, BIOS Memory Space, for detailed description of the BIOS memory map and the configuration bits.

The XBUSOE# signal controls outputs of the F245. When the XBUSOE# signal is asserted, the F245 drives its A buffers or B buffers depending on the state of the XBUSTR# signal. The ESC asserts the XBUSOE# signal for I/O cycles decoded to be in the address range of the peripherals supported by the ESC if these peripherals are enabled in the Peripheral Chip Select A register and Peripheral Chip Select B register.

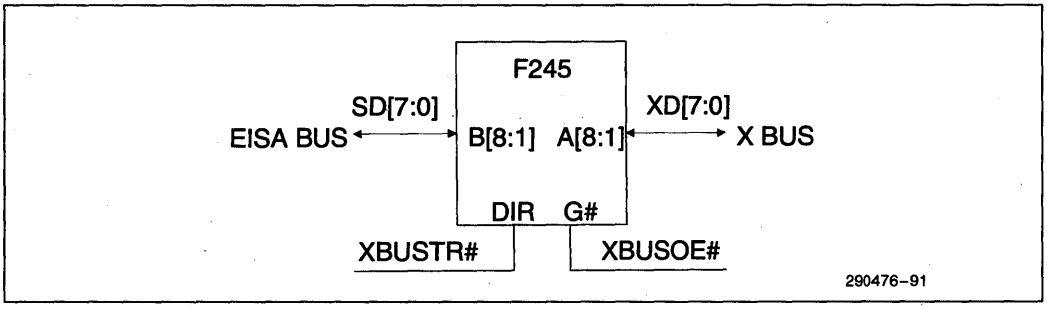


Figure 11-4. X-Bus Data Buffer

12.0 ELECTRICAL CHARACTERISTICS

12.1 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 . . . . . 0.70W Fully Loaded
. . . . . 0.55W with Four Slots

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.

**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.*

12.2 D.C. Characteristics

12.2.1 JUNCTION TEMPERATURE SPECIFICATIONS

The junction temperature for the ESC is +93°C with a case temperature of 85°C. To guarantee device operation at +85°C case temperature the ambient temperature allowable is shown in Table 12-1.

Table 12-1. ESC Maximum Allowable Ambient Temperature/Air Flow Rates

EISA Loading	Still	100 lfpm	200 lfpm	400 lfpm
4 Slots	63°C	66°C	69°C	72°C
8 Slots	56°C	60°C	63°C	67°C

12.2.2 EISA BUS D.C. SPECIFICATIONS

EISA Signals

BCLKOUT(out), BCLK(in), LA[31:27]#/CPG[4:0](t/s), LA[26:2](t/s), BE[3:0]#(t/s), M/IO#(t/s), W/R#(t/s), EX32#(o/d), EX16#(o/d), START#(t/s), CMD#(out), EXRDY(o/d), SLBURST#(in), MSBURST#(t/s), MASTER16#(in), SD[7:0](t/s)

ISA Signals

BALE(out), SA[1:0](t/s), SBHE#(t/s), M16#(o/d), IO16#(o/d), MRDC#(t/s), MWTC#(t/s), SMRDC#(out), SMWTC#(out), IORC#(t/s), IOWC#(t/s), CHRDY(o/d), IOCHK#(in), NOWS#(o/d), OSC(in), RSTDRV(out), REFRESH#(t/s), AEN#(out), AEN[4:1]/EAEN[4:1](out)

DMA and Arbitration Signals

DREQ[3:0,7:5](in), DACK[3:0,7:5]#(out), EOP(t/s), MREQ[3:0]#(in), MREQ[7:4]#/PIRQ[0:3]#(in), MACK[3:0]#/EMACK[3:0](out)

X-Bus and Integrated Logic Signals

SPKR(out), SLOWH#(out), IRQ[15:1](in), INT(out), NMI(out), SALE#(out), LASAOE#(out), SALAOE#(out), FERR#(in), IGNNE#(out), LBIOSCS#(out), KYBDCS#(out), ALTRST#(out), ALTA20(out), ABFULL(in), RTCALE(out), RTCRD#(out), RTCWR#(out), FDCCS#(out), DSKCHG(in), DLIGHT#(out), CRAMRD#(out), CRAMWR#(out), XBUSTR#(out), XBUSOE#(out), GPCS[2:0]#/ECS[2:0](out)

Interchip Signals

EISAHOLD(out), EISAHLDA(in), PEREQ#/INTA#(in), NMFLUSH#(t/s), SDCPYEN[13:01]#(out), SDCPYUP(out), SDOE[2:0]#(out), SDLE[3:0]#(out)

Table 12-2. E/ISA Bus D.C. Specifications (V_{DD} = 5V ±5%, T_{case} = 0°C to +85°C)

Symbol	Parameter	Min	Max	Test Conditions	Notes
V_{IL1}	Input Low Voltage		0.8V		
V_{IH1}	Input High Voltage	2.0V			
V_{IL2}	Input Low Voltage		0.8V		1
V_{IH2}	Input High Voltage	V_{CC} − 0.8V			1
V_{OL1}	Output Low Voltage		0.45V	I_{OL} = 24 mA	2
V_{OH1}	Output High Voltage	2.4V		I_{OH} = −5.0 mA	2
V_{OL2}	Output Low Voltage		0.45V	I_{OL} = 1 mA	3
V_{OH2}	Output High Voltage	V_{CC} − 0.45V		I_{OH} = −1 mA	3
V_{OL3}	Output Low Voltage		0.45V	I_{OL} = 8.0 mA	4
V_{OH3}	Output High Voltage	2.4V		I_{OH} = −2.0 mA	4
I_{LI}	Input Leakage Current		±15 μA	$0V < V_{IN} < V_{CC}$	
I_{LO}	Output Leakage Current		±15 μA	$0.45V < V_{IN} < V_{CC}$	
C_{IN}	Capacitance Input		8 pF		
C_{OUT}	Capacitance Output		15 pF	@1 MHz	
I_{CC}	V_{CC} Supply Current		TBD	TBD	

NOTES:
1. All EISA Bus signals use V_{IL1}, V_{IH1} for input levels except for the Interchip signals: SDCPYEN#, SDCPYUP, SDOE#, SDLE#, EISAHOLD, EISAHLDA, PEREQ#/INTA#, INTCHIP0, NMFLUSH#.
2. BALE, BCLKOUT, BE[3:0]#, CHRDY, CMD#, EOP, EX16#, EX32#, EXRDY, IO16#, IORC#, IOWC#, LA[31:2], M16#, M/IO#, MRDC#, MSBURST#, MWTC#, REFRESH#, RSTDRV, SA[1:0], SBHE#, SD[7:0], SMRDC#, SMWTC#, START#, W/R#, SPKR#.
3. ALTA20, AEN[4:1], ALTRST#, CRAMRD#, CRAMWT#, INT, DLIGHT, EISAHOLD, FDCCS#, IDECS#, IGNNE#, KEYBDCS#, LASAOE#, LBIOSCS#, NMFLUSH#, RTCALE, RTCRD#, RTCWR#, SALAOE#, SALE#, SDCPYEN[13:01]#, SDCPYUP, SDLE[3:0]#, SDOE[2:0]#, SLOWH#, XBUSOE#, XBUSTR#.
4. DACK[7:5,3:0], MACK[7:0]#.

12.2.3 PCI LOCAL BUS D.C. SPECIFICATIONS

PCI System Signals

PCICLK(in), RESET(in)

PCI Shared Signals

PERR#(in), SERR#(in)

Table 12-3. PCI Local Bus D.C. Specifications (V_{DD} = 5V ±5%, T_{case} = 0°C to +85°C)

Symbol	Parameter	Min	Max	Test Conditions
V_{IL}	Input Low Voltage		0.8V	
V_{IH}	Input High Voltage	2.0V	5.5V	
I_{IL}	Low-Level Input Current		−70 μA	V_{IN} = 0.5V
I_{IH}	High-Level Input Current		−70 μA	V_{IN} = 2.7V
$C_{I/O}$	Input/Output Capacitance		10 pF	@ 1 MHz
C_{CLK}	PCICLK Signal Input Capacitance		17 pF	@ 1 MHz
I_{CC}	V_{CC} Supply Current		TBD	TBD

12.3 A.C. Characteristics

The Symbol column shows the timing variable used in the A.C. timing waveforms. The parameter column contains the description of the timing and its reference signal. If the timing is for a particular bus cycle, the cycles will be listed in parenthesis. Burst cycles include standard timings for their requirements. The Min column lists either the minimum delay time, set-up time, or hold time requirement in nano-seconds unless stated otherwise. The Max column lists the maximum delay time also in nano-seconds. The Figure column shows what A.C. timing waveforms the parameter can be found. The Note column may contain a number to refer to a specific note found at the end of the table.

The A.C. specifications are based upon the specified capacitive loading values in Table 12-4. The minimum capacitive loading value is 50 pF for all signals.

Table 12-4. Capacitive Loading Table

Signals	Loading
BALE, BCLKOUT, BE[3:0]#, CHRDY, CMD#, EOP, EX16#, EX32#, EXRDY, IO16#, IORC#, IOWC#, LA[31:2], M16#, M/IO#, MRDC#, MSBURST#, MWTC#, REFRESH#, RSTDRV, SA[1:0], SBHE#, SD[7:0], SMRDC#, SMWTC#, START#, W/R#	240 pF
ALTA20, AEN[4:1], ALTRST#, CRAMRD#, CRAMWT#, INT, DLIGHT, FDCCS#, IDECS#, IGNNE#, KEYBDCS#, LASAOE#, LBIOSCS#, NMFLUSH#, RTCALE, RTCRD#, RTCWR#, SALAOE#, SALE#, SLOWH#, SPKR, XBUSOE#, XBUSTR#, PCICLK, RESET, PERR#, SERR#	50 pF
DACK[7:5,3:0], MACK[7:0]#	120 pF
EISAHOLD, SDCPYEN[13:1]#, SDCPYUP, SDLE[3:0]#, SDOE[2:0]#	30 pF

12.3.1 CLOCK SIGNALS A.C. SPECIFICATIONS

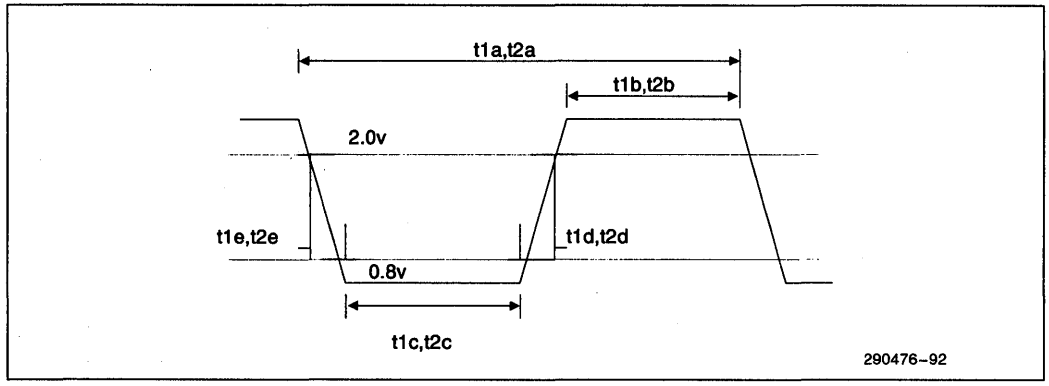


Figure 12-1. Clock Timings

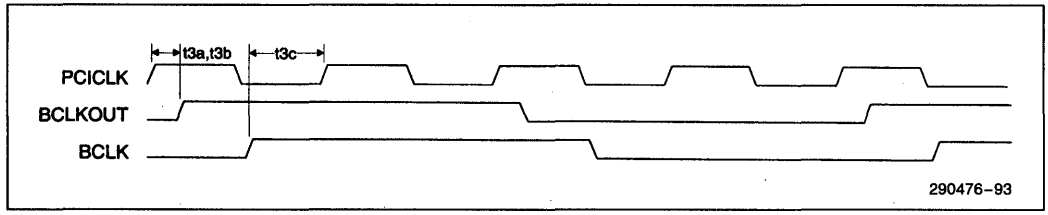


Figure 12-2. BCLKOUT and BCLK

Table 12-5. Clock Signals A.C. Specifications (V_{DD} = 5V ±5%, T_{case} = 0°C to +85°C)

Symbol	Parameter	Min	Max	Figures	Notes
PCICLK(1)					
t1a	Cycle Time	30		12-1	
t1b	High Time (at 2.0V)	40% * Tcyc		12-1	
t1c	Low Time (at 0.8V)	40% * Tcyc		12-1	
t1d	Rise Time (0.8V to 2.0V)	3		12-1	
t1e	Fall Time (2.0V to 0.8V)	3		12-1	
BCLK, BCLKOUT					
t2a	Clock Period (0.8V to 0.8V)	120		12-1	
t2b	Low Time (at 0.8V)	55		12-1	
t2c	High Time (at 2.0V)	56		12-1	
t2d	Rise Time (0.8V to 2.0V)		7	12-1	
t2e	Fall Time (2.0V to 0.8V)		6	12-1	
t3a	BCLKOUT Delay from PCICLK Rising (240 pF)	5	17	12-2	
t3b	BCLKOUT Delay from PCICLK Rising (30 pF)	2	9	12-2	
t3c	BCLK Setup to PCICLK Rising	4		12-2	
t3d	BCLK Hold from PCICLK Rising	4		12-2	
OSC					
t2a	Clock Period (0.8V to 0.8V)	65	70	12-1	
t2b	Low Time (0.8V)	20		12-1	
t2c	High Time (2.0V)	20		12-1	

12.3.2 A.C. SPECIFICATIONS

Table 12-6. EISA Interface A.C. Specifications (V_{DD} = 5V ±5%, T_{case} = 0°C to +85°C)

Symbol	Parameter	Min	Max	Figures	Notes
EISA MASTER					
BE[3:0]#					
t4a	Delay from BCLK Rising/Falling (Assembly, Disassembly)	1	30	12-6,7,8	
t4b	Setup to BCLK Rising	80		12-3	
t4c	Setup to BCLK Falling (Burst)	30		12-9	
t4d	Hold from BCLK Rising	20		12-3	
t4e	Hold from BCLK Falling (Burst)	2		12-9	
SA0, SA1, SBHE#					
t5	Valid Delay from BCLK Falling		30	12-3	
M/IO#					
t6a	Setup to BCLK Rising	80		12-3	
t6b	Hold from BCLK Rising	20		12-3	
W/R#					
t7a	Setup to BCLK Falling	25		12-3	
t7b	Hold from BCLK Rising	20		12-3	
MASTER16#					
t8a	Setup to BCLK Rising	17		12-10	
t8b	Hold from BCLK Rising	0		12-10	
MRDC#, MWTC#, IORC#, IOWC#, SMRDC#, SMWTC#					
t9a	Delay from BCLK Rising/Falling	2	30	12-11	
t9b	IOWC# Delay from BCLK Rising/Falling	3.5	25	12-11	
START#					
t10a	Delay from BCLK Rising	1	25	12-4,6,7	
t10b	Setup to BCLK Rising	23		12-3	
t10c	Hold from BCLK Rising	0		12-3	
t10d	Pulsewidth	Tper-5		12-4,6,7	
t10e	START# Setup to PCICLK Rising	9			
CMD#					
t11	Delay from BCLK Rising	1	30	12-3	
BALE					
t12	Delay from BCLK Rising/Falling	1	20	12-3	
MSBURST#					
t13a	Setup to BCLK Falling	12		12-12	
t13b	Hold from BCLK Falling	20		12-12	

Table 12-6. EISA Interface A.C. Specifications (V_{DD} = 5V ±5%, T_{case} = 0°C to +85°C) (Continued)

Symbol	Parameter	Min	Max	Figures	Notes
EISA MASTER (continued)					
EX32#, EX16#					
t14a	Valid Delay from BCLK Falling (End of Backoff)	3	32	12-6	
t14b	Setup to BCLK Rising	25		12-3	
t14c	Hold from BCLK Rising	55		12-3	
t14d	Float Delay from BCLK Falling (End of Backoff)	2	19	12-6	
IO16#					
t15a	Setup to BCLK Falling	20		12-13	
t15b	Hold from BCLK Falling	20		12-13	
M16#					
t16a	Setup to BCLK Rising	18		12-14	
t16b	Hold from BCLK Falling	0		12-14	
NOWS#					
t17a	Setup to BCLK Falling	10		12-15	5
t17b	Hold from BCLK Falling	20		12-15	5
CHRDY					
t18a	Negate Setup to BCLK Falling	7		12-18	
t18b	Assert Setup to BCLK Rising	10		12-18	
t18c	Negated Pulsewidth	10		12-18	
EXRDY#					
t19a	Setup to BCLK Falling	13		12-19	
t19b	Hold from BCLK Falling	0		12-19	
ISA MASTER					
BE[3:0]#					
t20	Valid Delay from SA[1:0], SBHE# Valid		60	12-4	
M/IO, W/R					
t21	Delay from IORC#, IOWC#, MRDC# Asserted		40	12-4	
SMRDC#, SMWTC#					
t22	Delay from MRDC#, MWTC# Asserted	0	25	12-4	
START#					
t23a	Delay from BCLK Rising	1	25	12-4	
t23b	Pulsewidth	Tper-5		12-4	

Table 12-6. EISA Interface A.C. Specifications (V_{DD} = 5V ±5%, T_{case} = 0°C to +85°C) (Continued)

Symbol	Parameter	Min	Max	Figures	Notes
ISA MASTER (Continued)					
CMD#					
t24a	Delay from BCLK Rising	1	30	12-4	
t24b	Delay from MRDC#, IORC# Rising (Read)	0	30	12-4	
IO16#, M16#					
t25	Delay from EX32#, EX16# Asserted		50	12-4	
CHRDY					
t26a	Negate Delay from MRDC#, MWTC#, IORC#, IOWC#		60	12-4	
t26b	Float Delay from BCLK Falling		15	12-4	
DMA CYCLES					
LA[31:2]					
t27a	Delay from BCLK Falling (DMA "C")	2	30	12-19	
t27b	Delay from BCLK Falling (DMA "A, B", Refresh)		50	12-18	
BE[3:0]#					
t28a	Delay from BCLK Falling (DMA "C")	2	30	12-19	
t28b	Delay from BCLK Falling (DMA "A, B", Refresh)		50	12-18	
SA0, SA1, SBHE#					
t29	Delay from BCLK Falling (DMA "A, B", Refresh)		50	12-18	
REFRESH#					
t30a	Delay from BCLK Rising		50	12-25	
t30b	Setup to BCLK Rising	18		12-25	
t30c	Hold from BCLK Rising	3		12-25	

Table 12-6. EISA Interface A.C. Specifications (V_{DD} = 5V ±5%, T_{case} = 0°C to +85°C) (Continued)

Symbol	Parameter	Min	Max	Figures	Notes
DMA CYCLES (Continued)					
AEN#, AEN[4:1]/EAEN[4:1]					
t31	Delay from BCLK Falling		50	12-18	
M/IO, W/R					
t32	Delay from BCLK Falling (DMA "A, B", Refresh)	1	50	12-18	
MRDC#, MWTC#, IORC#, IOWC#, SMRDC#, SMWTC#					
t33a	Delay from BCLK Rising/Falling	2	30	12-11	
t33b	IOWC# Delay from BCLK Rising	3	25	12-11	
START#					
t34a	Delay from BCLK Rising	1	25	12-18	
t34b	Pulsewidth	Tper-5		12-18	
CMD#					
t35	Delay from BCLK Rising/Falling	1	30	12-18	
SLBURST#					
t36a	Setup to BCLK Rising (DMA "C")	15		12-19	
t36b	Hold from BCLK Rising (DMA "C")	25		12-19	
MSBURST#					
t37	Delay from BCLK Falling (DMA "C")	1	30	12-19	
EXRDY#					
t38a	Valid Delay from BCLK Rising	1	32	12-20	
t38b	Setup to BCLK Falling	13		12-17	
t38c	Hold from BCLK Falling	0		12-17	
t38d	Float Delay from BCLK Falling	2	30	12-20	
EOP					
t39a	Delay from BCLK (DMA "A, B")		40	12-21	
t39b	Setup to BCLK Rising (EOPIN Mode)	15		12-21	
t39c	Hold from BCLK Rising (EOPIN Mode)	15		12-21	
t39d	Float Delay from DACK# Rising (DMA "A, B")		30	12-21	
SLAVE ACCESS					
SD[7:0]					
t40a	Valid Data Delay from BCLK Rising (Read)		60	12-22	
t40b	Setup to BCLK Rising (Write)	15		12-22	
t40c	Hold from BCLK Rising (Write)	5		12-22	
t40d	Float Delay from CMD# Negated (Read)		30	12-22	
NOWS#					
t41	Asserted from LA Valid		60	12-23	
CPG[4:0]#					
t42	Valid Delay from LA Valid		60	12-23	4

Table 12-7. Data Swap Logic A.C. Specifications (V_{DD} = 5V ±5%, T_{case} = 0°C to +85°C)

Symbol	Parameter	Min	Max	Figures	Notes
SDCPYEN[13:1]#					
t43a	Delay from BCLK Rising/Falling (Standard Mismatch, Assembly)	5	25	12-5,6	
t43b	Negate Delay from BCLK Rising (Disassembly)		25	12-7	
t43c	Assert Delay from BCLK Falling (Disassembly)		19	12-7	
t43d	Delay from BCLK Rising/Falling (Burst Mismatch)	5	15	12-9	
t43e	Valid Delay from IO16# Asserted (ISA 16-Bit Slave)		20	12-24	
t43f	SDCPYUP Setup to SDCPYEN[13:1]# Valid	0		12-26	
t43g	Delay from MRDC#, MWTC#, IORC#, IOWC# Asserted/Negated (ISA Mismatch)	2	20	12-25	
SDCPYUP					
t44a	Delay from BCLK Rising/Falling (Standard Mismatch, Assembly, Disassembly)	2	25	12-5, 6	
t44b	Delay from MWTC#, IOWC# Asserted/Negated (ISA Mismatch)	2	20	12-26	
SDLE[3:0]#					
t45a	Assert Delay from BCLK Rising (Assembly)	1	25	12-6	
t45b	Deassert Setup to CMD# Negated (Assembly)	2		12-6	
t45c	Deassert Setup to SDCPYEN[13:01]# Negated (Assembly)	2		12-6	
t45d	Delay from BCLK Falling (Disassembly)		25	12-8	
SDOE[2:0]#					
t46a	Delay from BCLK Rising/Falling (Redrive)	1	12.5	12-6	3
t46b	Delay from BCLK Rising/Falling (Disassembly)	1	25	12-8	
t46c	Assert Delay from BCLK Falling (Disassembly)	1	21	12-8	

Table 12-8. Arbitration, Timer, Interrupt A.C. Specifications (V_{DD} = 5V ±5%, T_{case} = 0°C to +85°C)

Symbol	Parameter	Min	Max	Figures	Notes
MREQ[7:0]#					
t47a	Setup to BCLK Rising	17		12-27	
t47b	Hold from BCLK Rising	15		12-27	
MACK[3:0]#/EMACK[3:0]					
t48a	Delay from BCLK Rising		40	12-27	
t48c	Delay from MREQx# Rising	240		12-27	
DREQ[7:5,3:0]					
t49a	Setup to BCLK Rising	15		12-4,27	
t49b	Hold from BCLK Rising	15		12-4,27	
DACK[7:5,3:0]#					
t50a	Delay from BCLK Rising		50	12-4,27	
t50b	Delay from DREQ Falling	240		12-4,27	

Table 12-8. Arbitration, Timer, Interrupt A.C. Specifications
(V_{DD} = 5V ±5%, T_{case} = 0°C to +85°C) (Continued)

Symbol	Parameter	Min	Max	Figures	Notes
INT					
t51a	Delay from PIRQ[0:3], IRQ[15:1], OSC, BCLK, FERR#		200	12-28	
t51b	Delay from ABFULL Asserted		100	12-28	
NMI					
t52	Delay from IOCHK, SERR# Asserted		200	12-29	
RSTDRV					
t53	Delay from RESET# Asserted/Negated		100	12-29	
SLOWH#					
t54	Delay from BCLK Falling		200	12-28	
SPKR					
t55a	Delay from BCLK Falling		100	12-28	
t55b	Delay from OSC Rising		200	12-28	

Table 12-9. Interchip Signals A.C. Specifications (V_{DD} = 5V ±5%, T_{case} = 0°C to +85°C)

Symbol	Parameter	Min	Max	Figures	Notes
EISAHOLD					
t56	Delay from PCICLK Rising	2	15	12-30	
EISAHOLDA					
t57a	Setup to PCICLK Rising	12		12-30	
t57b	Hold from PCICLK Rising	0		12-30	
PEREQ#/INTA#					
t58a	Setup to PCICLK Rising	12		12-30	
t58b	Hold from PCICLK Rising	0		12-30	
NMFLUSH#					
t59a	Delay from PCICLK Rising (Request from ESC)	2	13	12-31	
t59b	Setup from PCICLK Rising (Acknowledge from PCEB)	12		12-31	
t59c	Hold from PCICLK Rising (Acknowledge from PCEB)	2		12-31	

Table 12-10. Integrated Logic Support Signals
A.C. Specifications (V_{DD} = 5V ±5%, T_{case} = 0°C to +85°C)

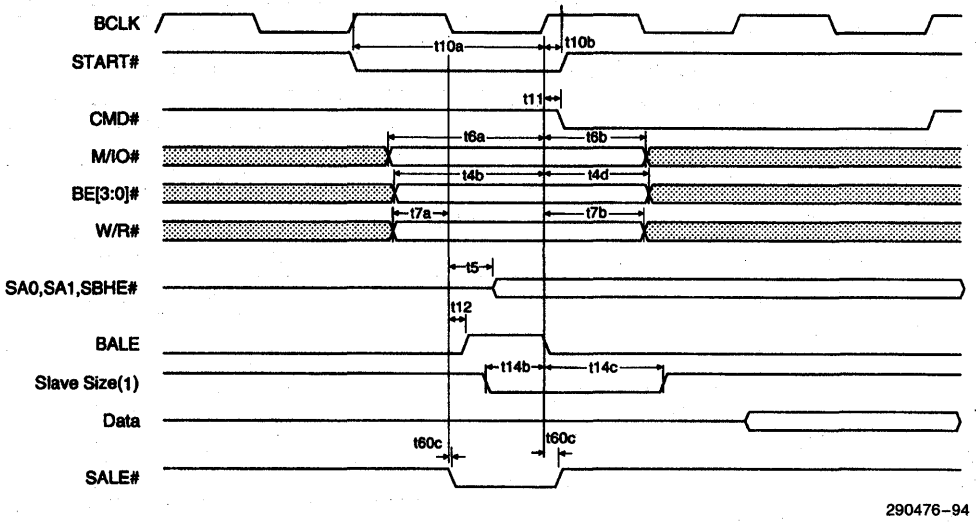
Symbol	Parameter	Min	Max	Figures	Notes
EISA ADDRESS BUFFER CONTROL (SALE#, LASAOE#, SALAOE#)					
t60a	SALE# Negate Delay from BCLK Rising (DMA Master)	1	38	12-32	2
t60b	SALE# Assert Delay from BCLK Rising (DMA Master)	1	38	12-32	
t60c	SALE# Delay from BCLK Falling/Rising (EISA Master, DMA Assembly/Disassembly)	1	14	12-3	
t60d	SALE# Delay from BCLK Rising (ISA Master)	1	38	12-4	
t61a	LASAOE# Delay from BCLK Rising		30	12-4	
t61b	SALAOE# Delay from BCLK Rising		35	12-4	
t61c	SALAOE# Negate Delay from REFRESH# Asserted		35	12-25	

Table 12-10. Integrated Logic Support Signals
A.C. Specifications (V_{DD} = 5V ±5%, T_{case} = 0°C to +85°C) (Continued)

Symbol	Parameter	Min	Max	Figures	Notes
COPROCESSOR INTERFACE (FERR#, IGNNE#)					
t62a	Asserted Delay from IOWC# Asserted		100	12-33	
t62b	Deassert Delay from FERR# Negated		100	12-33	
CHIP SELECTS (LBIOSCS#, KYBDCS#, FDCCS#, GPCS[2:0]#/ECS[2:0]#)					
t63	Delay from LA Valid and BALE Asserted		60	12-34	
KEYBOARD CONTROLLER (ALTRST#, ALTA20, ABFULL)					
t64a	ALTRST# Asserted Pulsewidth	480		12-35	
t64b	ALTA20 Delay from BCLK Rising		50	12-35	
REAL TIME CLOCK (RTCRD#, RTCWR#, RTCALE)					
t65a	RTCALE Delay from BCLK Rising		40	12-36	
t65b	RTCRD# Assert Delay from XBUSTR# Negated		50	12-36	
t65c	RTCRD# Deassert Delay from IORC# Negated		30	12-36	
t65d	RTCWR# Assert Delay from IOWC# Asserted		50	12-36	
t65e	RTCWR# Deassert Delay from BCLK Rising		50	12-36	
CONFIGURATION RAM (CRAMRD#, CRAMWR#)					
t66a	CRAMRD# Assert Delay from XBUSTR# Negated		50	12-36	
t66b	CRAMRD# Deassert Delay from IORC# Negated		30	12-36	
t66c	CRAMWR# Assert Delay from IOWC# Asserted		50	12-36	
t66d	CRAMWR# Deassert Delay from BCLK Rising		50	12-36	
X-BUS CONTROL (UBUSTR#, UBUSOE#)					
t67a	XBUSTR# Delay from IORC#, MWTC#		15	12-36,37	
t67b	XBUSOE# Delay from IORC#, IOWC#, MRDC#, MWTC#		21	12-37	
FLOPPY DISK CONTROLLER (DSKCHG)					
t68a	DSKCHG Valid to SD7 Valid		25	12-38	
t68b	IORC# Asserted to SD7 Driven		25	12-38	
t68c	IORC# Negated to SD7 Floated		25	12-38	

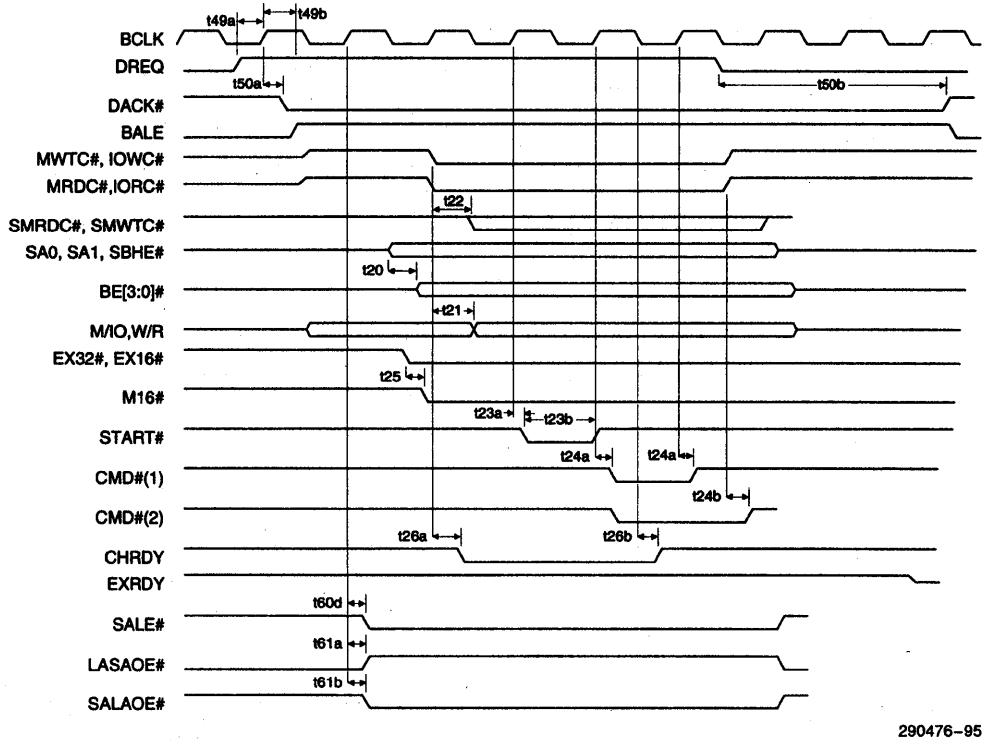
NOTES:
1. PCICLK input must meet PCI Specification requirements for clock skew.
2. For DMA type "A, B" read cycles SALE# is asserted for one BCLK. For DMA compatible read, type "B" write, type "C" read and write SALE# is asserted for two BCLKs. For DMA compatible write, and type "A" write SALE# is asserted for three BCLKs.
3. For EISA redrive cycles SDOE is negated on the first rising edge of BCLK. For DMA type "B" redrive cycles SDOE is negated on the second falling edge of BCLK. For DMA type "C" redrive cycles SDOE is negated on the third falling edge of BCLK.
4. The CPG[4:0] signals are shared with LA[31:27]# and are outputs only during I/O access to 0800h-08FFh.
5. For an 8-bit ISA slave, NOWS# is sampled starting on the second falling edge of BCLK after CMD# is asserted. 16-bit ISA slaves are sampled starting on the first falling edge of BCLK after CMD# is asserted.

12.3.3 A.C. TIMING WAVEFORMS



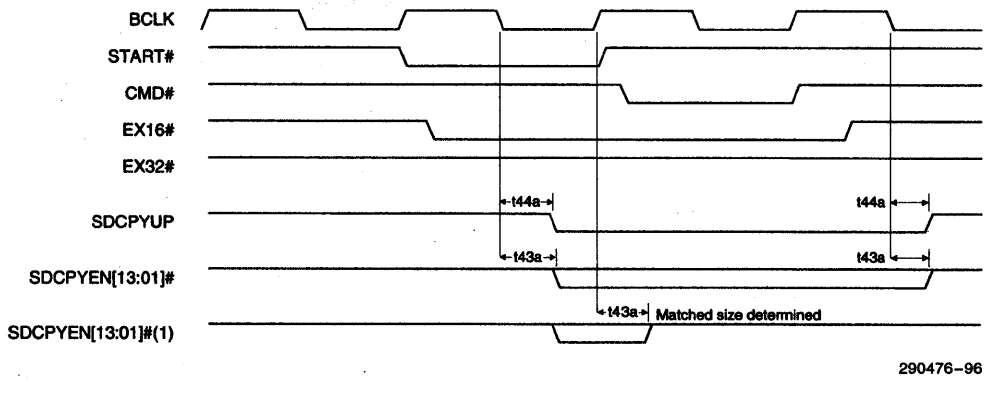
NOTE:
1. EX32#, EX16#

Figure 12-3. EISA Master Cycle



NOTES:
1. CMD# negation during write cycles is triggered off of BCLK rising.
2. CMD# negation during read cycles is triggered off of MRDC#, IORC#.

Figure 12-4. ISA Master Cycle



NOTE:
1. The copy buffers are enabled until the ESC can determine that the master and slave sizes match. At this time the copy buffers are disabled.

Figure 12-5. Mismatch

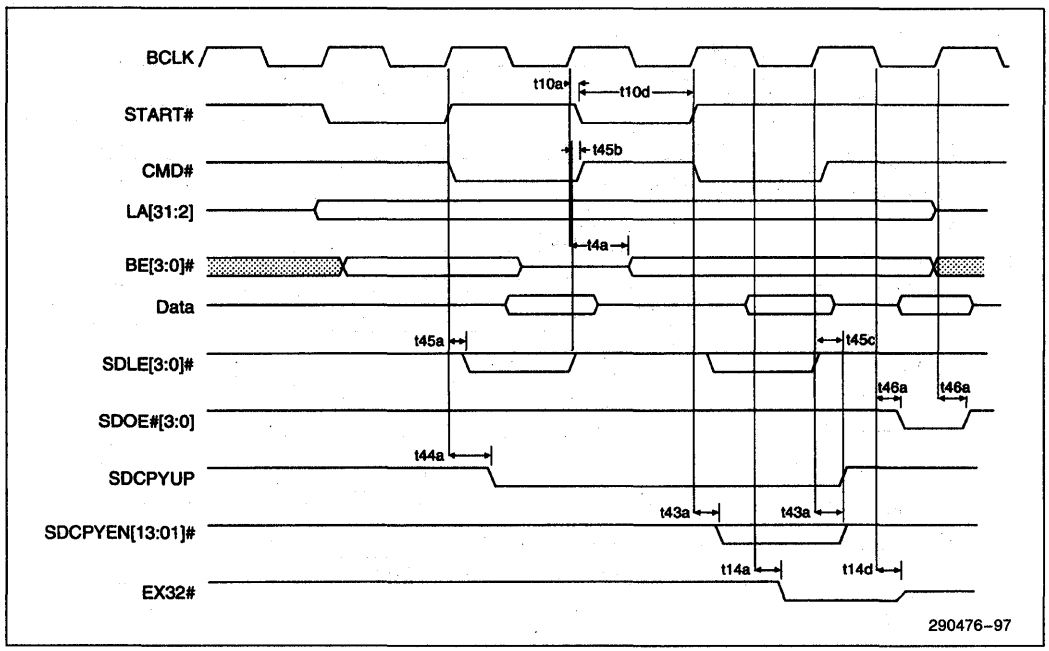


Figure 12-6. Assembly with Redrive

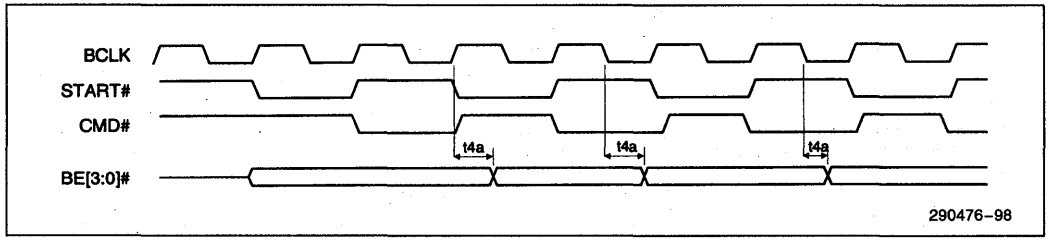


Figure 12-7. DMA Assembly, Disassembly

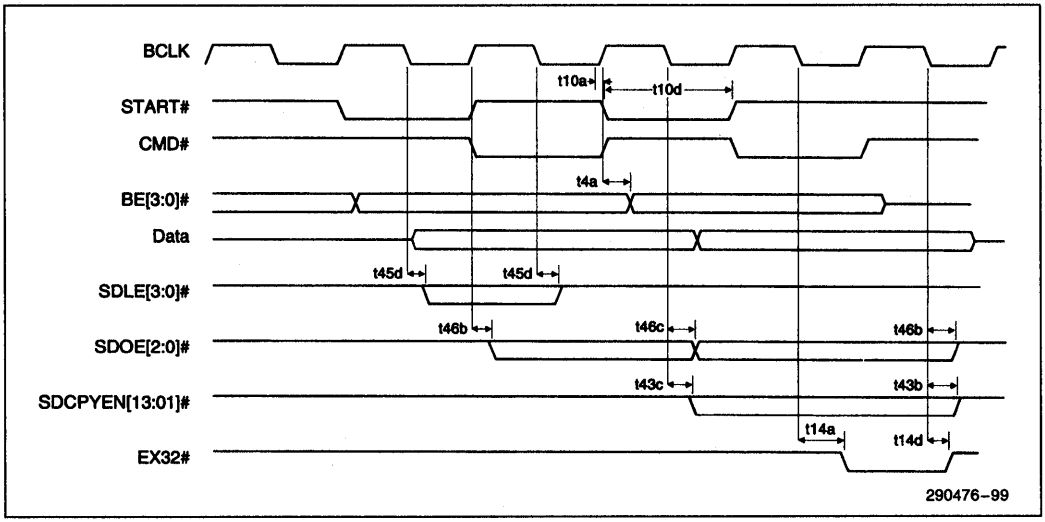


Figure 12-8. Disassembly

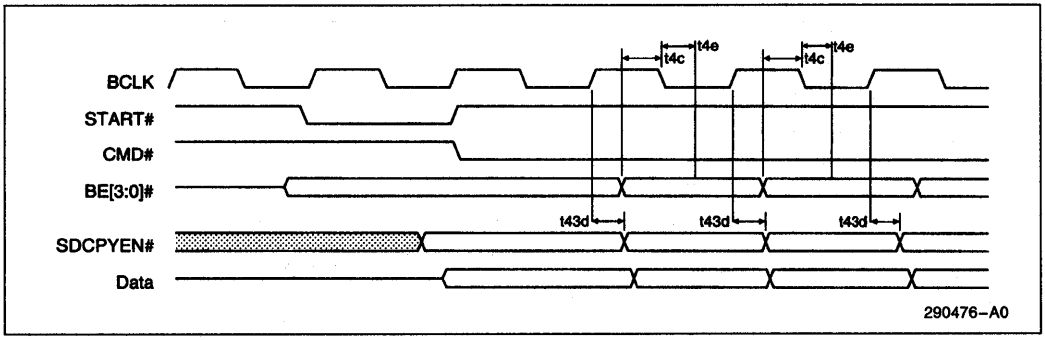


Figure 12-9. Burst Mismatch

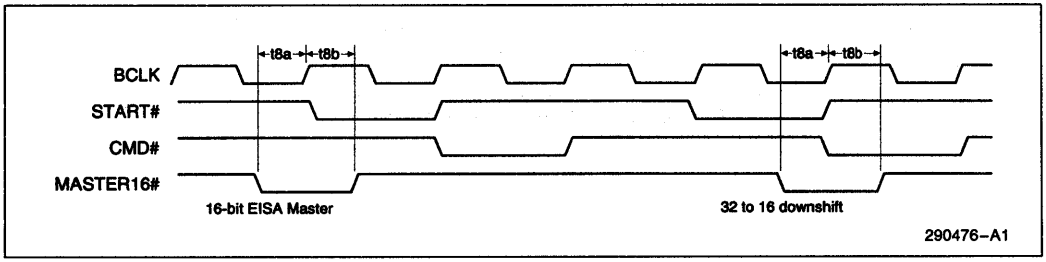


Figure 12-10. MASTER16# Signal

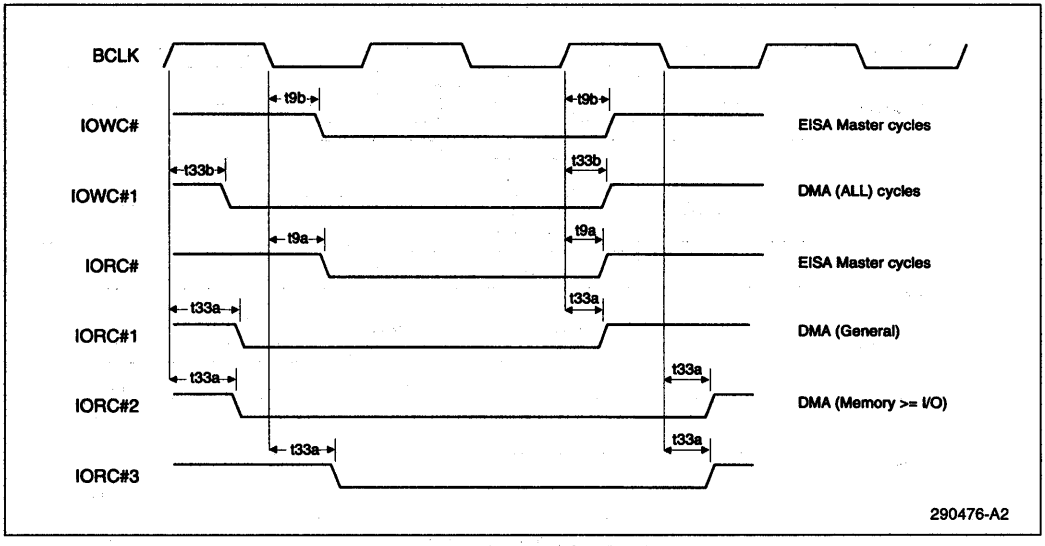


Figure 12-11a. IORC#, IOWC# Signals

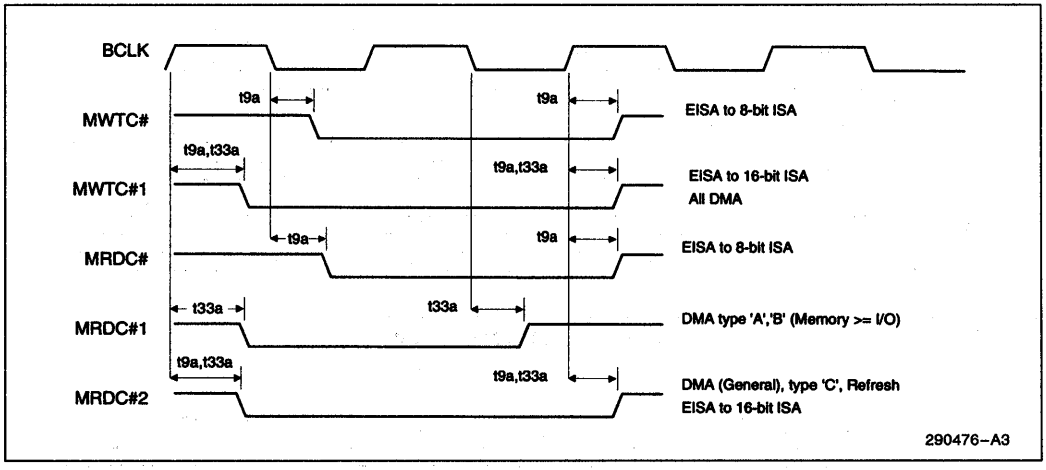


Figure 12-11b. MRDC#, MWTC# Signals

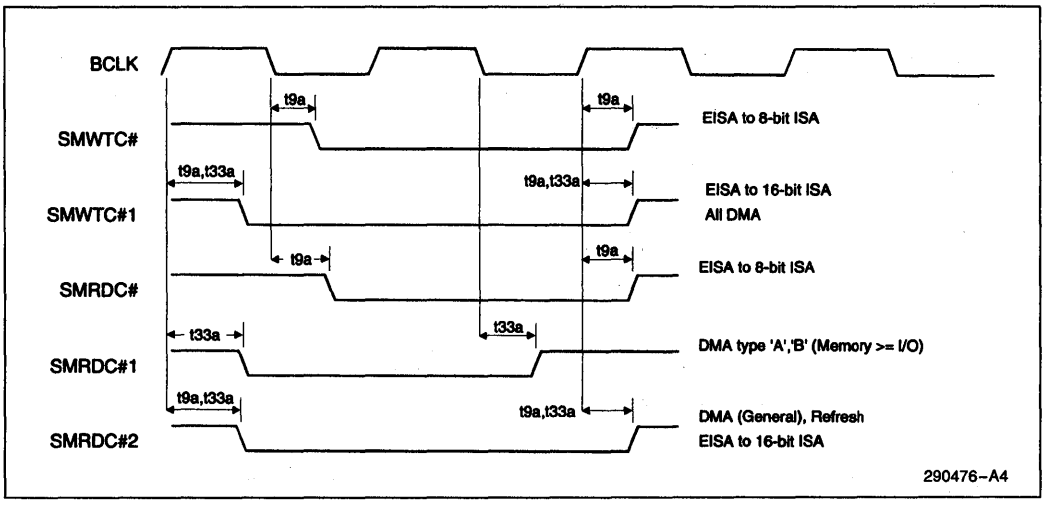


Figure 12-11c. SMRDC#, SMWTC# Signals (EISA Master Cycles)

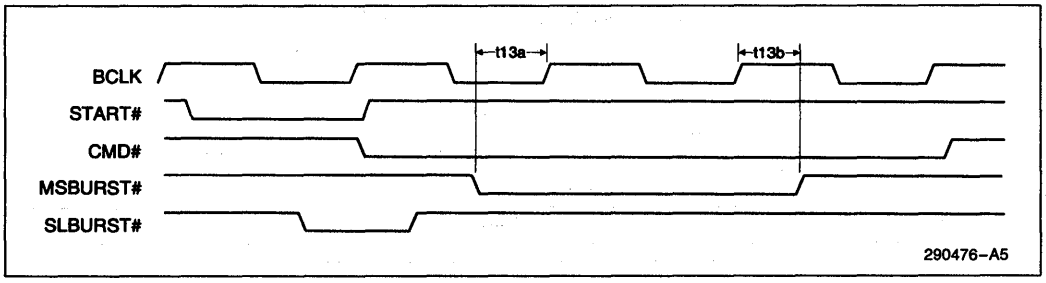


Figure 12-12. EISA Burst Cycle

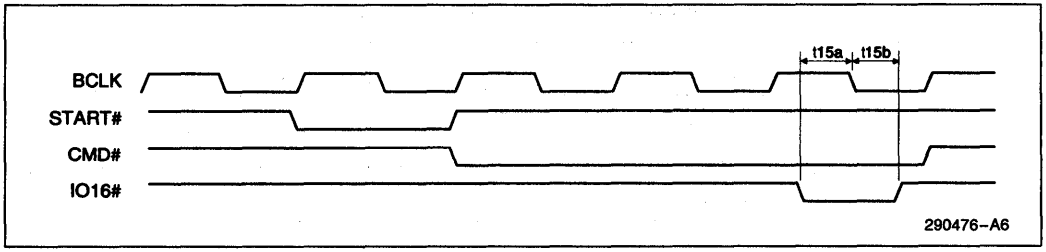


Figure 12-13. IO16# Signal

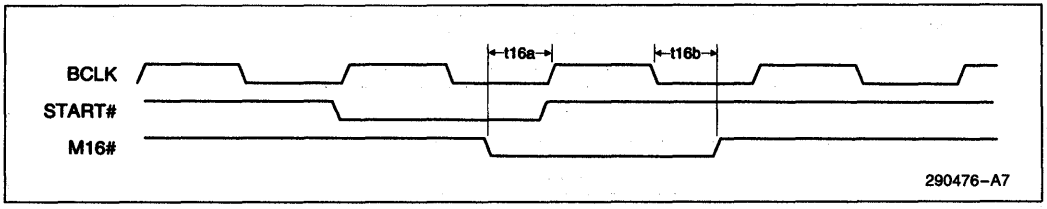


Figure 12-14. M16# Signal

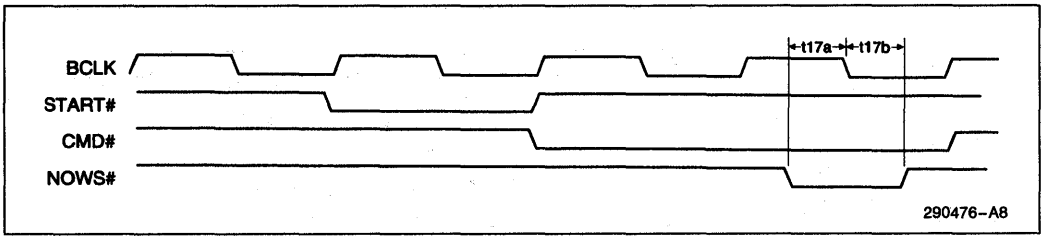


Figure 12-15. NOWS# Signal

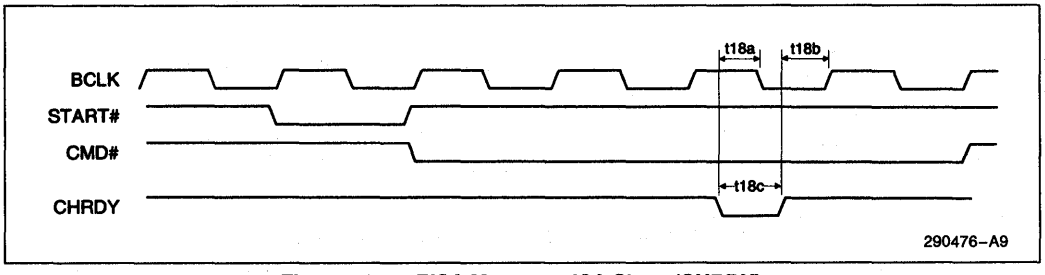


Figure 12-16. EISA Master to ISA Slave (CHRDY)

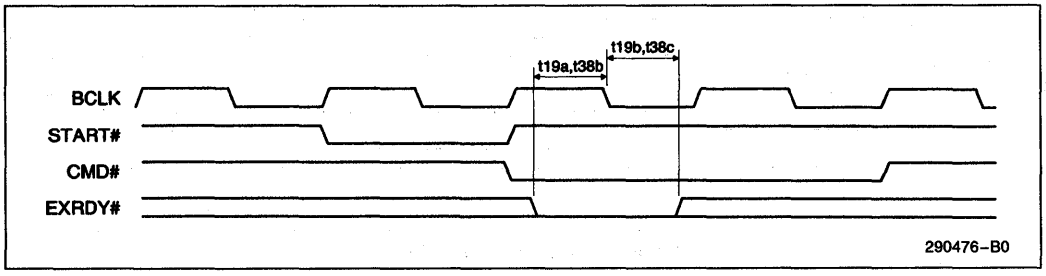


Figure 12-17. EISA Slave (EXRDY)

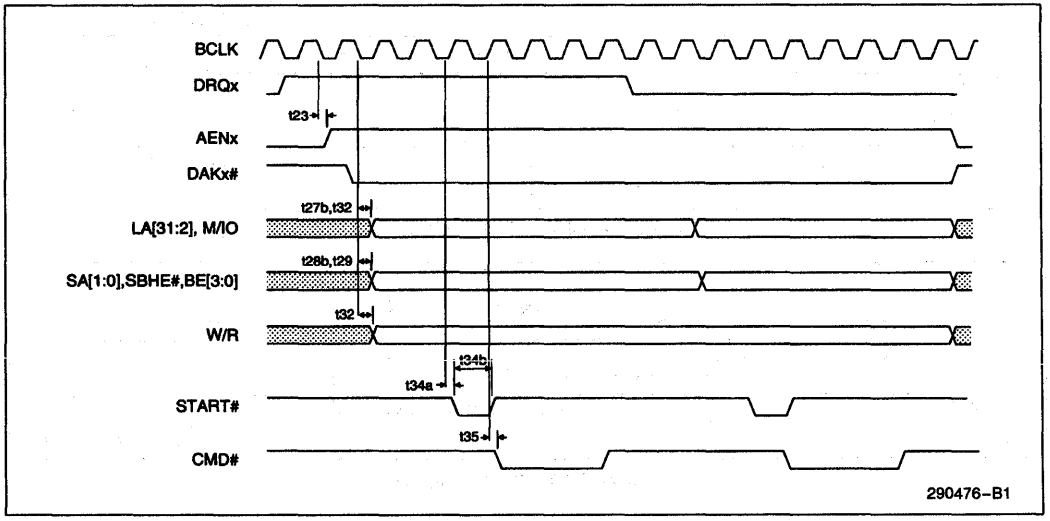


Figure 12-18. DMA Cycle (General)

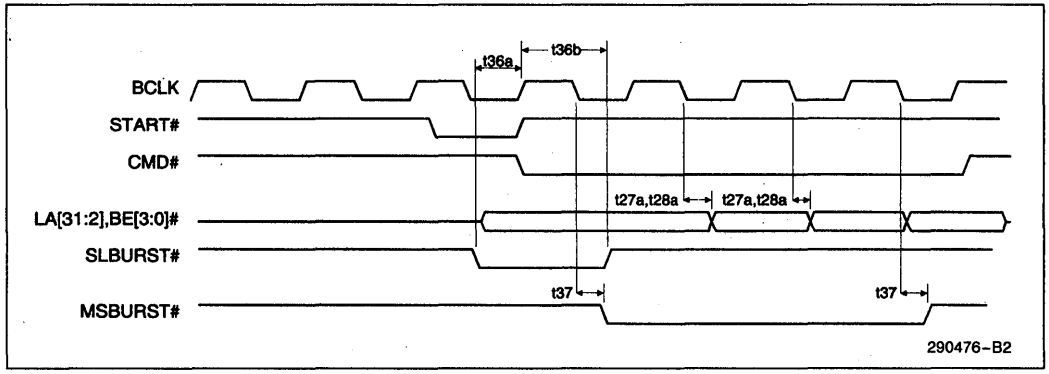


Figure 12-19. DMA Burst Cycle (Type "C")

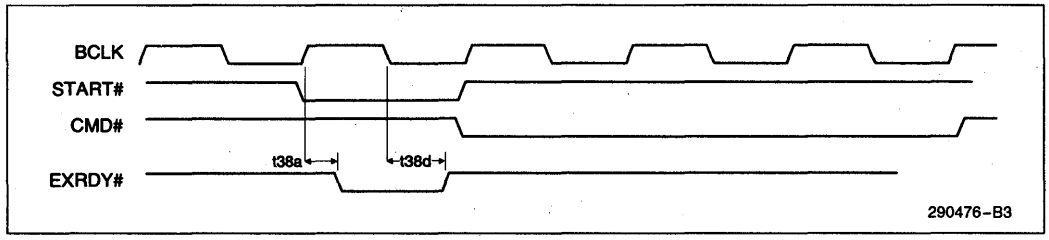


Figure 12-20. DMA Master Burst Write Cycle

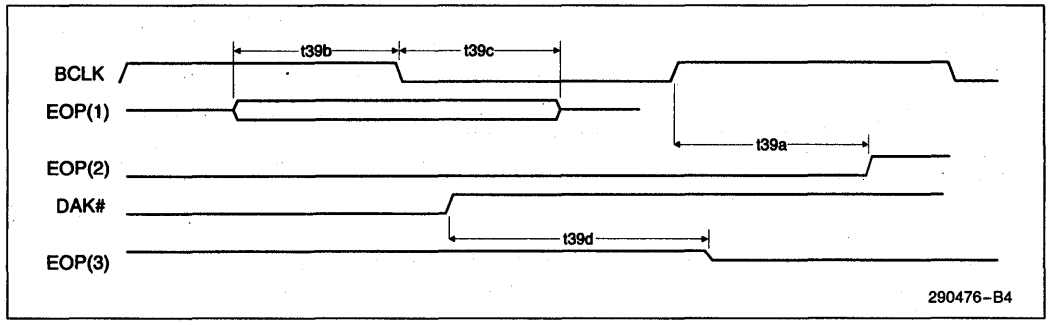


Figure 12-21. EOP Signal

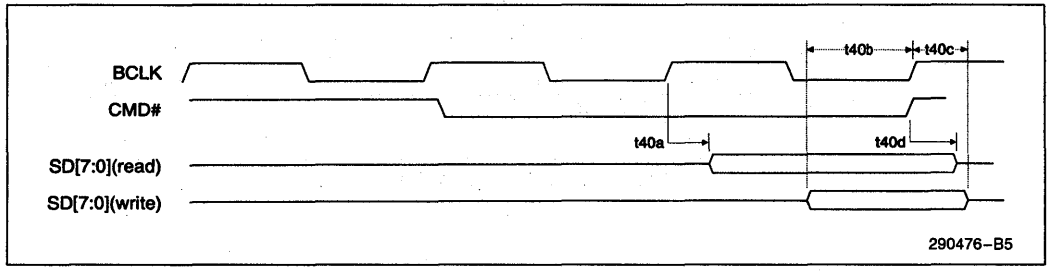


Figure 12-22. SD[7:0] Signals

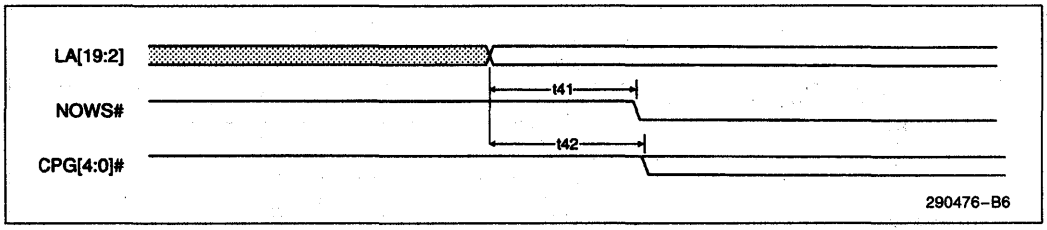
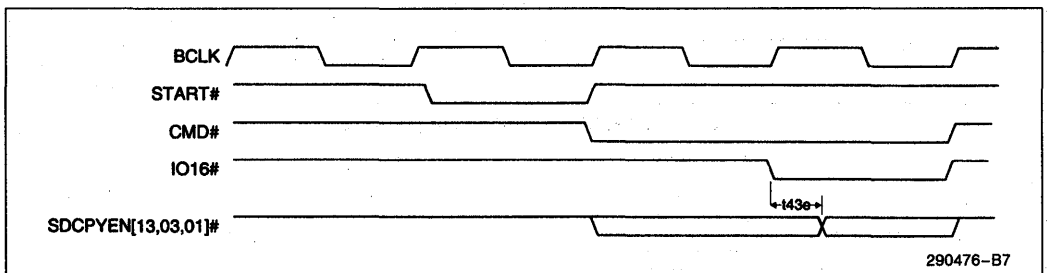


Figure 12-23. NOWS#, CPG[4:0]# Signals



NOTE:
SDCPYEN02# is handled the same for 8-bit and 16-bit slaves. IO16# does not change this signal.

Figure 12-24. EISA or DMA Master Read from 16-Bit ISA I/O

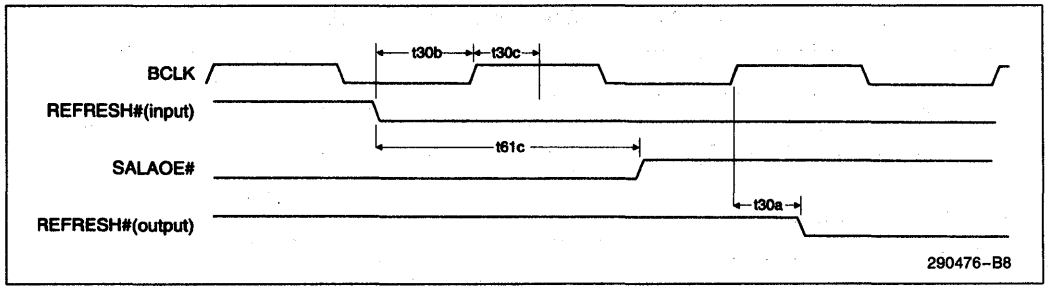


Figure 12-25. REFRESH# Signal

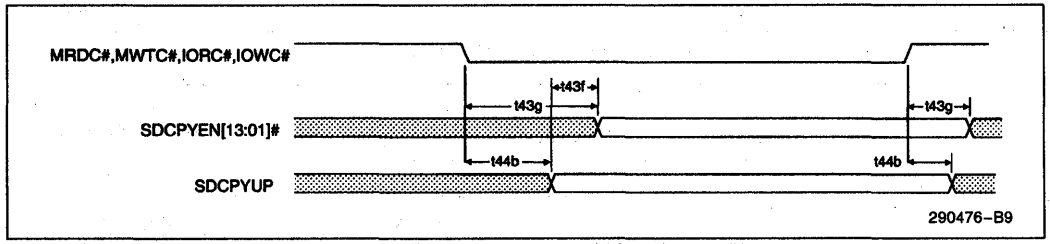


Figure 12-26. ISA Mismatch

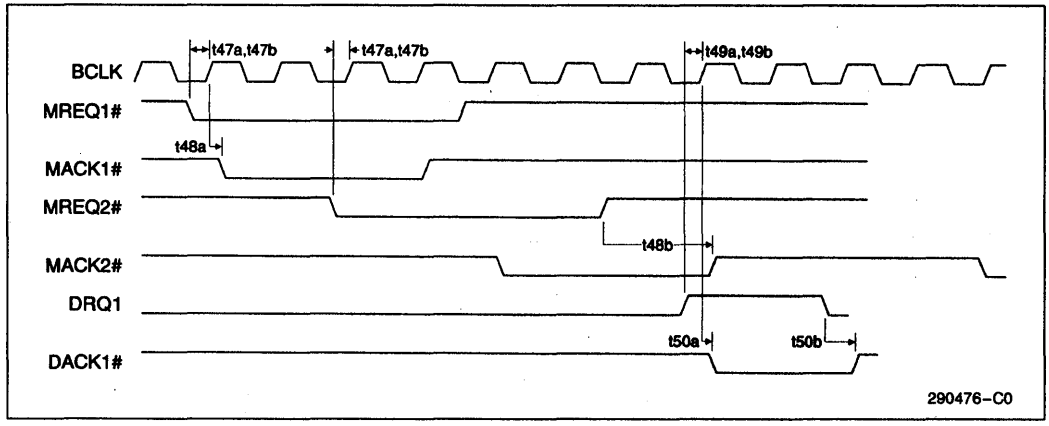


Figure 12-27. DMA and EISA Arbitration

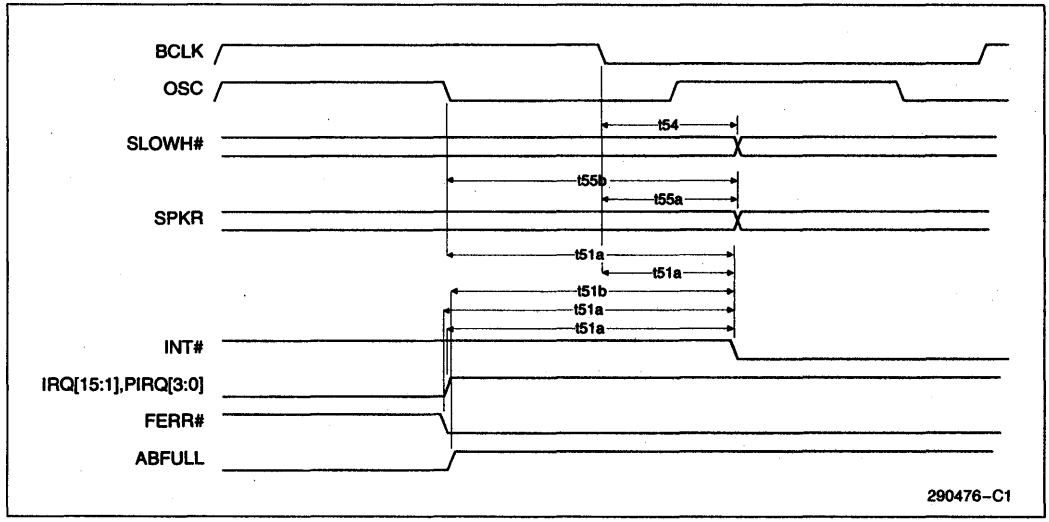


Figure 12-28. Timers and Interrupts

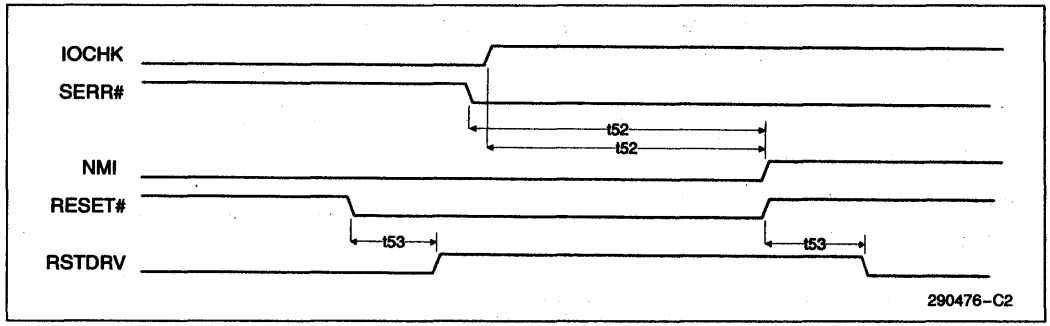


Figure 12-29. NMI and RSTDRV

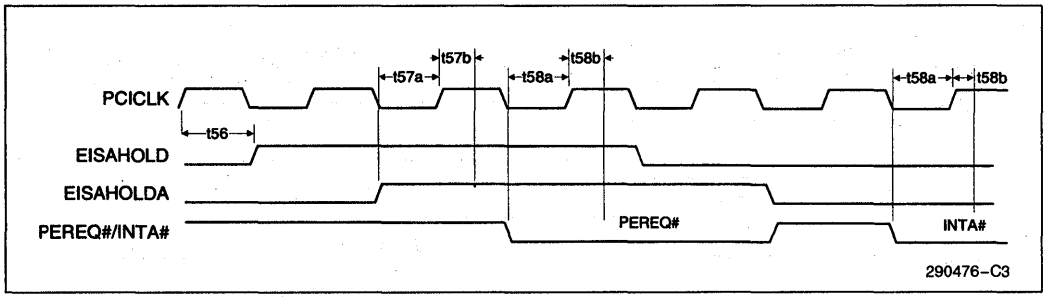


Figure 12-30. EISAHOLD Signals

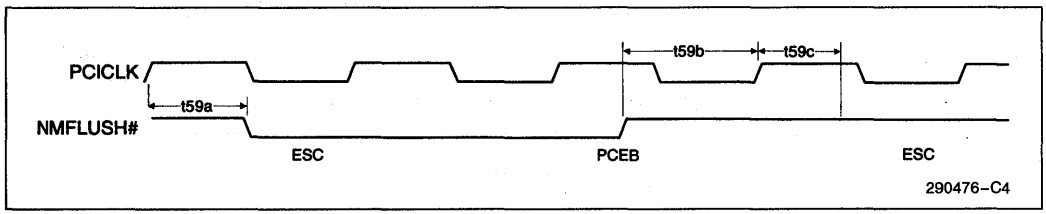


Figure 12-31. Flush Request

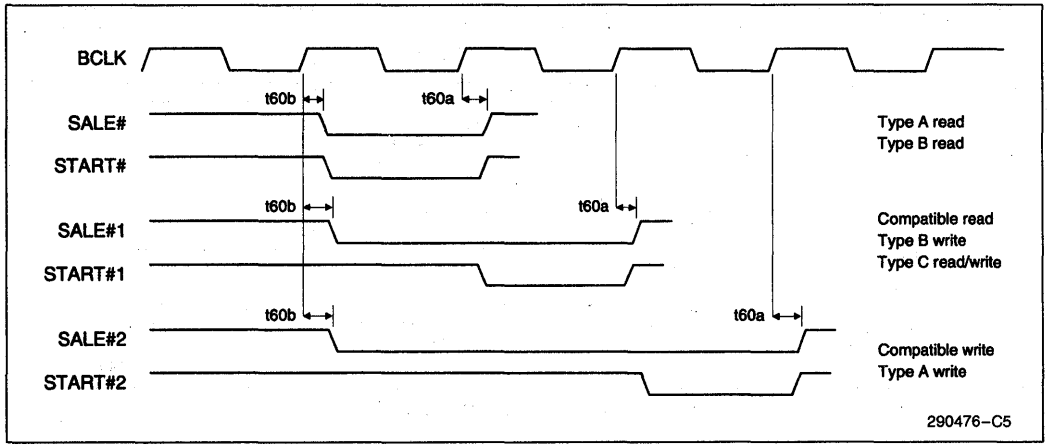


Figure 12-32. SALE for DMA Cycles

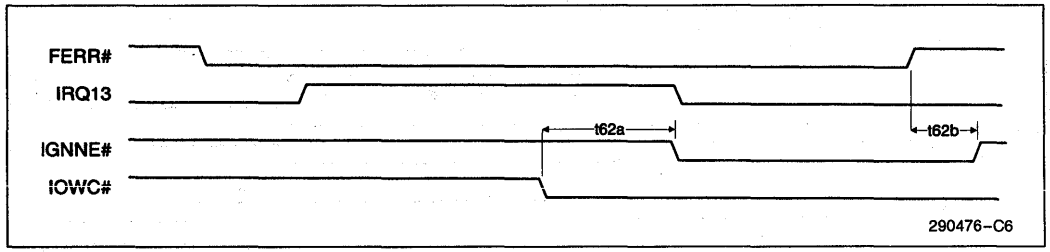
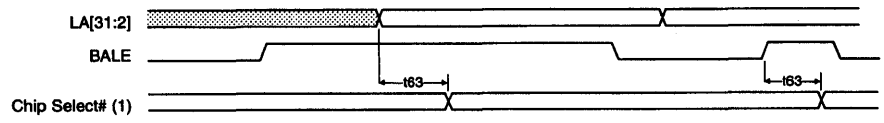


Figure 12-33. IGNNE# Signal



NOTE:
1. LBIOSCS#, KEYBDCS#, FDCCS#, GPCS[2:0]#/ECS[2:0]#.

Figure 12-34. Chip Select

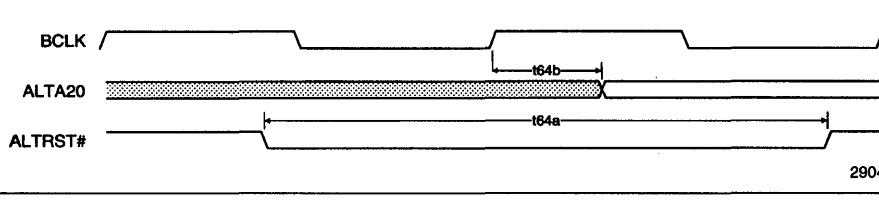


Figure 12-35. Keyboard Controller Signals

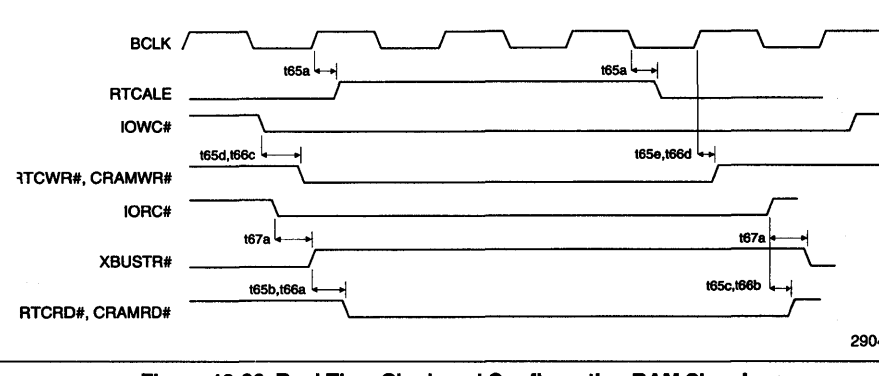


Figure 12-36. Real Time Clock and Configuration RAM Signals

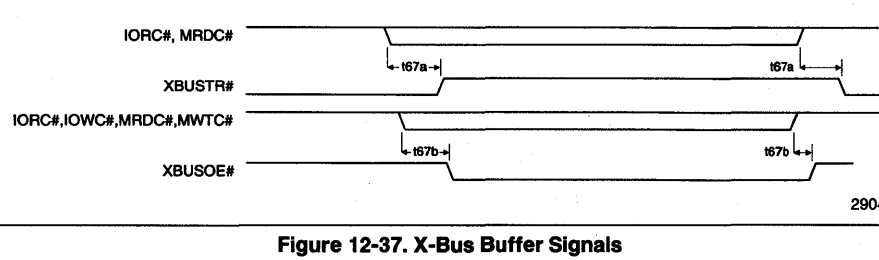


Figure 12-37. X-Bus Buffer Signals

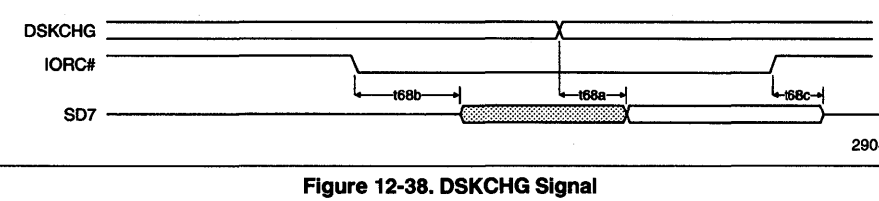


Figure 12-38. DSKCHG Signal

13.0 PINOUT AND PACKAGE INFORMATION

The ESC package is a 208-pin plastic Quad flat pack (PQFP). The package signals are shown in Figure 13-1 and listed in Tables 13-1 and 13-2.

13.1 Pinout and Pin Assignment

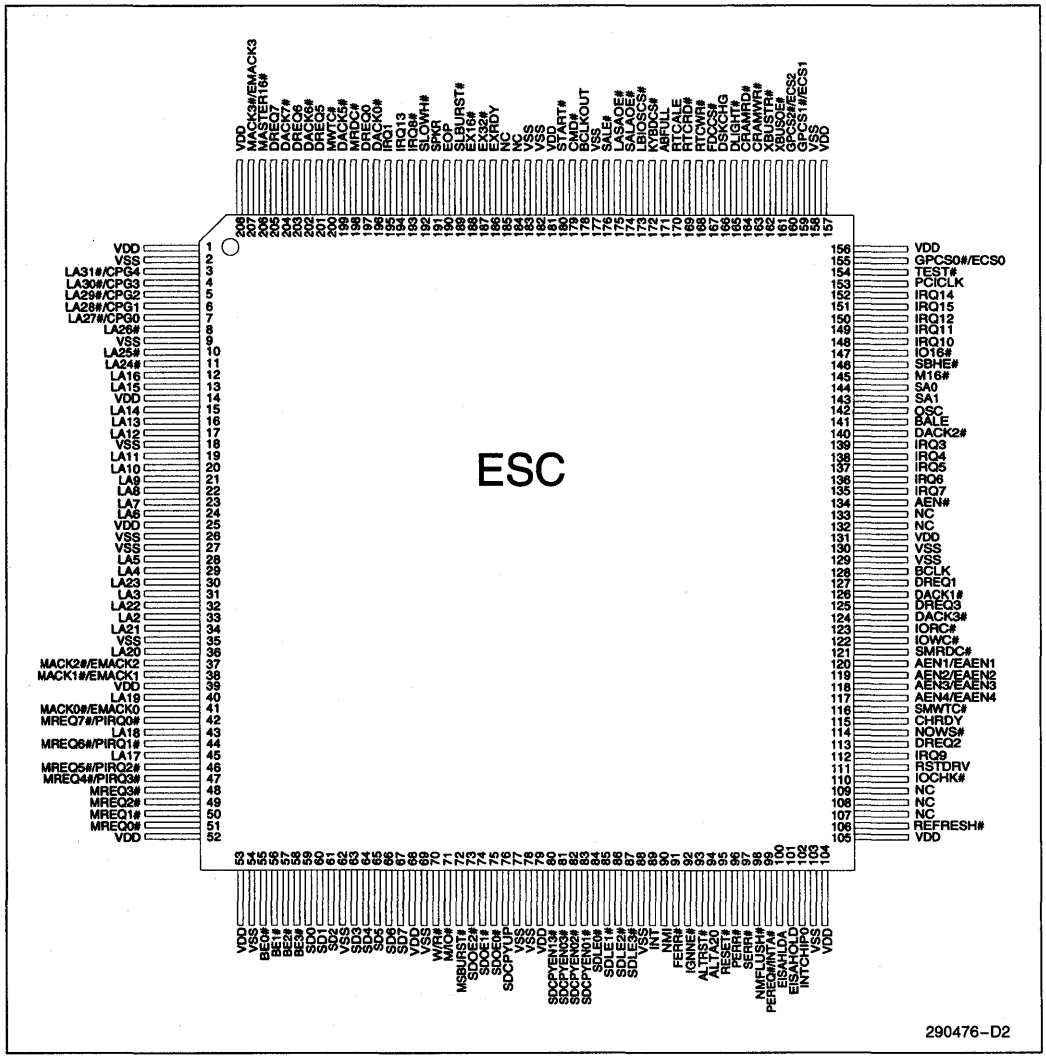


Figure 13-1. ESC Package Pinout

Table 13-1. ESC Alphabetical Pin Assignment

Name	Pin #	Type
ABFULL	171	in
AEN#	134	out
AEN1/EAEN1	120	out
AEN2/EAEN2	119	out
AEN3/EAEN3	118	out
AEN4/EAEN4	117	out
ALTA20	94	out
ALTRST#	93	out
BALE	141	out
BCLK	128	in
BCLKOUT	178	out
BE0#	55	t/s
BE1#	56	t/s
BE2#	57	t/s
BE3#	58	t/s
CHRDY	115	o/d
CMD#	179	out
CRAMRD#	164	out
CRAMWR#	163	out
DACK0#	196	out
DACK1#	126	out
DACK2#	140	out
DACK3#	124	out
DACK5#	199	out
DACK6#	202	out
DACK7#	204	out
DLIGHT#	165	out
DREQ0	197	in
DREQ1	127	in
DREQ2	113	in
DREQ3	125	in
DREQ5	201	in
DREQ6	203	in
DREQ7	205	in
DSKCHG	166	in
EISAHLDA	100	in
EISAHOLD	101	out

Name	Pin #	Type
EOP	190	t/s
EX16#	188	o/d
EX32#	187	o/d
EXRDY	186	o/d
FDCCS#	167	out
FERR#	91	in
GPCS0#/ECS0	155	out
GPCS1#/ECS1	159	out
GPCS2#/ECS2	160	out
IGNNE#	92	out
INT	89	out
INTCHIP0	102	t/s
IO16#	147	o/d
IOCHK#	110	in
IORC#	123	t/s
IOWC#	122	t/s
IRQ1	195	in
IRQ3	139	in
IRQ4	138	in
IRQ5	137	in
IRQ6	136	in
IRQ7	135	in
IRQ8#	193	in
IRQ9	112	in
IRQ10	148	in
IRQ11	149	in
IRQ12	150	in
IRQ13	194	in
IRQ14	152	in
IRQ15	151	in
KYBDCS#	172	out
LA2	33	t/s
LA3	31	t/s
LA4	29	t/s
LA5	28	t/s
LA6	24	t/s
LA7	23	t/s

Name	Pin #	Type
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#/CPG0	7	t/s
LA28#/CPG1	6	t/s
LA29#/CPG2	5	t/s
LA30#/CPG3	4	t/s
LA31#/CPG4	3	t/s
LASAOE#	175	out
LBIOSCS#	173	out
M/IO#	71	t/s
M16#	145	o/d
MACK0#/ EMACK0	41	out
MACK1#/ EMACK1	38	out
MACK2#/ EMACK2	37	out
MACK3#/ EMACK3	207	out
MASTER16#	206	in

Table 13-1. ESC Alphabetical Pin Assignment (Continued)

Name	Pin #	Type
MRDC#	198	t/s
MREQ0#	51	in
MREQ1#	50	in
MREQ2#	49	in
MREQ3#	48	in
MREQ4#/ PIRQ3#	47	in
MREQ5#/ PIRQ2#	46	in
MREQ6#/ PIRQ1#	44	in
MREQ7#/ PIRQ0#	42	in
MSBURST#	72	t/s
MWTC#	200	t/s
NC	107	NC
NC	108	NC
NC	109	NC
NC	132	NC
NC	133	NC
NC	184	NC
NC	185	NC
NMFLUSH#	98	t/s
NMI	90	out
NOWS#	114	o/d
OSC	142	in
PCICLK	153	in
PEREQ#/INTA#	99	in
PERR#	96	in
REFRESH#	106	t/s
RESET#	95	in
RSTDRV	111	out
RTCALE	170	out
RTCRD#	169	out
RTCWR#	168	out
SA0	144	t/s

Name	Pin #	Type
SA1	143	t/s
SALAOE#	174	out
SALE#	176	out
SBHE#	146	t/s
SD0	59	t/s
SD1	60	t/s
SD2	61	t/s
SD3	63	t/s
SD4	64	t/s
SD5	65	t/s
SD6	66	t/s
SD7	67	t/s
SDCPYEN01#	83	out
SDCPYEN02#	82	out
SDCPYEN03#	81	out
SDCPYEN13#	80	out
SDCPYUP	76	out
SDLE0#	84	out
SDLE1#	85	out
SDLE2#	86	out
SDLE3#	87	out
SDOE0#	75	out
SDOE1#	74	out
SDOE2#	73	out
SERR#	97	in
SLBURST#	189	in
SLOWH#	192	out
SMRDC#	121	out
SMWTC#	116	out
SPKR	191	out
START#	180	t/s
TEST#	154	in
V_{DD}	1	V
V_{DD}	14	V
V_{DD}	25	V

Name	Pin #	Type
V_{DD}	39	V
V_{DD}	52	V
V_{DD}	53	V
V_{DD}	68	V
V_{DD}	79	V
V_{DD}	104	V
V_{DD}	105	V
V_{DD}	131	V
V_{DD}	156	V
V_{DD}	157	V
V_{DD}	181	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}	54	V
V_{SS}	62	V
V_{SS}	69	V
V_{SS}	77	V
V_{SS}	78	V
V_{SS}	88	V
V_{SS}	103	V
V_{SS}	129	V
V_{SS}	130	V
V_{SS}	158	V
V_{SS}	177	V
V_{SS}	182	V
V_{SS}	183	V
W/R#	70	t/s
XBUSOE#	161	out
XBUSTR#	162	out

Note:
NC pins require individual pullup resistors of 8K–10K.

Table 13-2. ESC Numerical Pin Assignment

Pin #	Name	Type
1	V_{DD}	V
2	V_{SS}	V
3	LA31#/CPG4	t/s
4	LA30#/CPG3	t/s
5	LA29#/CPG2	t/s
6	LA28#/CPG1	t/s
7	LA27#/CPG0	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
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

Pin #	Name	Type
37	MACK2#/ EMACK2	out
38	MACK1#/ EMACK1	out
39	V_{DD}	V
40	LA19	t/s
41	MACK0#/ EMACK0	out
42	MREQ7#/ PIRQ0#	in
43	LA18	t/s
44	MREQ6#/ PIRQ1#	in
45	LA17	t/s
46	MREQ5#/ PIRQ2#	in
47	MREQ4#/ PIRQ3#	in
48	MREQ3#	in
49	MREQ2#	in
50	MREQ1#	in
51	MREQ0#	in
52	V_{DD}	V
53	V_{DD}	V
54	V_{SS}	V
55	BE0#	t/s
56	BE1#	t/s
57	BE2#	t/s
58	BE3#	t/s
59	SD0	t/s
60	SD1	t/s
61	SD2	t/s
62	V_{SS}	V
63	SD3	t/s
64	SD4	t/s
65	SD5	t/s
66	SD6	t/s
67	SD7	t/s

Pin #	Name	Type
68	V_{DD}	V
69	V_{SS}	V
70	W/R#	t/s
71	M/IO#	t/s
72	MSBURST#	t/s
73	SDOE2#	out
74	SDOE1#	out
75	SDOE0#	out
76	SDCPYUP	out
77	V_{SS}	V
78	V_{SS}	V
79	V_{DD}	V
80	SDCPYEN13#	out
81	SDCPYEN03#	out
82	SDCPYEN02#	out
83	SDCPYEN01#	out
84	SDLE0#	out
85	SDLE1#	out
86	SDLE2#	out
87	SDLE3#	out
88	V_{SS}	V
89	INT	out
90	NMI	out
91	FERR#	in
92	IGNNE#	out
93	ALTRST#	out
94	ALTA20	out
95	RESET#	in
96	PERR#	in
97	SERR#	in
98	NMFLUSH#	t/s
99	PEREQ#/ INTA#	in
100	EISAHLDA	in
101	EISAHOLD	out
102	INTCHIP0	t/s

Note:
NC pins require individual pullup resistors of 8K–10K.

Table 13-2. ESC Numerical Pin Assignment (Continued)

Pin #	Name	Type
103	V_{SS}	V
104	V_{DD}	V
105	V_{DD}	V
106	REERESH#	t/s
107	NC	NC
108	NC	NC
109	NC	NC
110	IOCHK#	in
111	RSTDRV	out
112	IRQ9	in
113	DREQ2	in
114	NOWS#	o/d
115	CHRDY	o/d
116	SMWTC#	out
117	AEN4/EAEN4	out
118	AEN3/EAEN3	out
119	AEN2/EAEN2	out
120	AEN1/EAEN1	out
121	SMRDC#	out
122	IOWC#	t/s
123	IORC#	t/s
124	DACK3#	out
125	DREQ3	in
126	DACK1#	out
127	DREQ1	in
128	BCLK	in
129	V_{SS}	V
130	V_{SS}	V
131	V_{DD}	V
132	NC	NC
133	NC	NC
134	AEN#	out
135	IRQ7	in
136	IRQ6	in
137	IRQ5	in
138	IRQ4	in

Pin #	Name	Type
139	IRQ3	in
140	DACK2#	out
141	BALE	out
142	OSC	in
143	SA1	t/s
144	SA0	t/s
145	M16#	o/d
146	SBHE#	t/s
147	IO16#	o/d
148	IRQ10	in
149	IRQ11	in
150	IRQ12	in
151	IRQ15	in
152	IRQ14	in
153	PCICLK	in
154	TEST#	in
155	GPCS0#/ECS0	out
156	V_{DD}	V
157	V_{DD}	V
158	V_{SS}	V
159	GPCS1#/ECS1	out
160	GPCS2#/ECS2	out
161	XBUSOE#	out
162	XBUSTR#	out
163	CRAMWR#	out
164	CRAMRD#	out
165	DLIGHT#	out
166	DSKCHG	in
167	FDCCS#	out
168	RTCWR#	out
169	RTCRD#	out
170	RTCALE	out
171	ABFULL	in
172	KYBDCS#	out
173	LBIOSCS#	out
174	SALAOE#	out

Pin #	Name	Type
175	LASAOE#	out
176	SALE#	out
177	V_{SS}	V
178	BCLKOUT	out
179	CMD#	out
180	START#	t/s
181	V_{DD}	V
182	V_{SS}	V
183	V_{SS}	V
184	NC	NC
185	NC	NC
186	EXRDY	o/d
187	EX32#	o/d
188	EX16#	o/d
189	SLBURST#	in
190	EOP	t/s
191	SPKR	out
192	SLOWH#	out
193	IRQ8#	in
194	IRQ13	in
195	IRQ1	in
196	DACK0#	out
197	DREQ0	in
198	MRDC#	t/s
199	DACK5#	out
200	MWTC#	t/s
201	DREQ5	in
202	DACK6#	out
203	DREQ6	in
204	DACK7#	out
205	DREQ7	in
206	MASTER16#	in
207	MACK3#/ EMACK3	out
208	V_{DD}	V

13.2 Package Characteristics

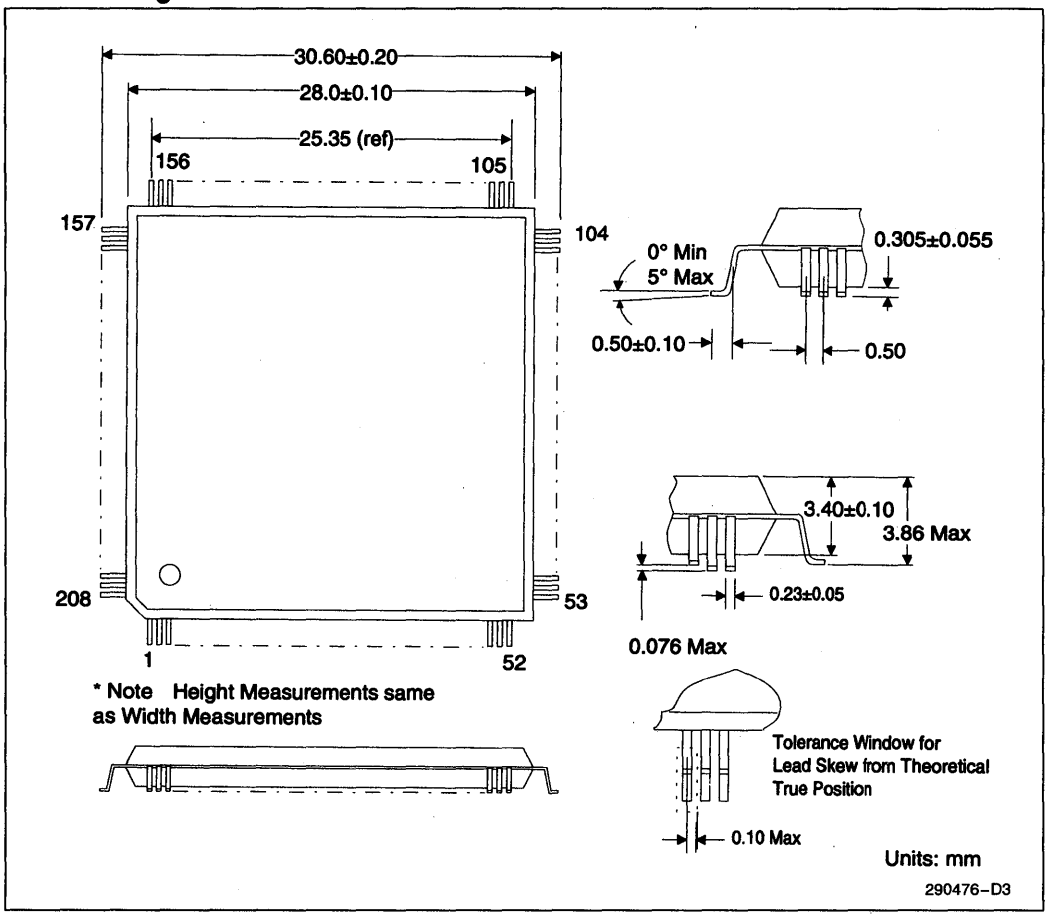


Figure 13-2. Packaging Dimension Information

82375EB
PCI-EISA Bridge (PCEB)

- **Provides the Bridge Between the PCI 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 at 25 MHz 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 Busses**
- **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**
- **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**

The 82375EB PCI-EISA Bridge (PCEB) provides the master/slave functions on both the Peripheral Component Interconnect (PCI) 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.

Manufactured and tested for Intel Corporation by LSI Logic in accordance with LSI's internal standards.

*Windows and Win-NT are trademarks of Microsoft Corporation.

April 1993
Order Number: 290477-001

82375EB PCI-EISA BRIDGE (PCEB)

CONTENTS

CONTENTS

CONTENTS

CONTENTS

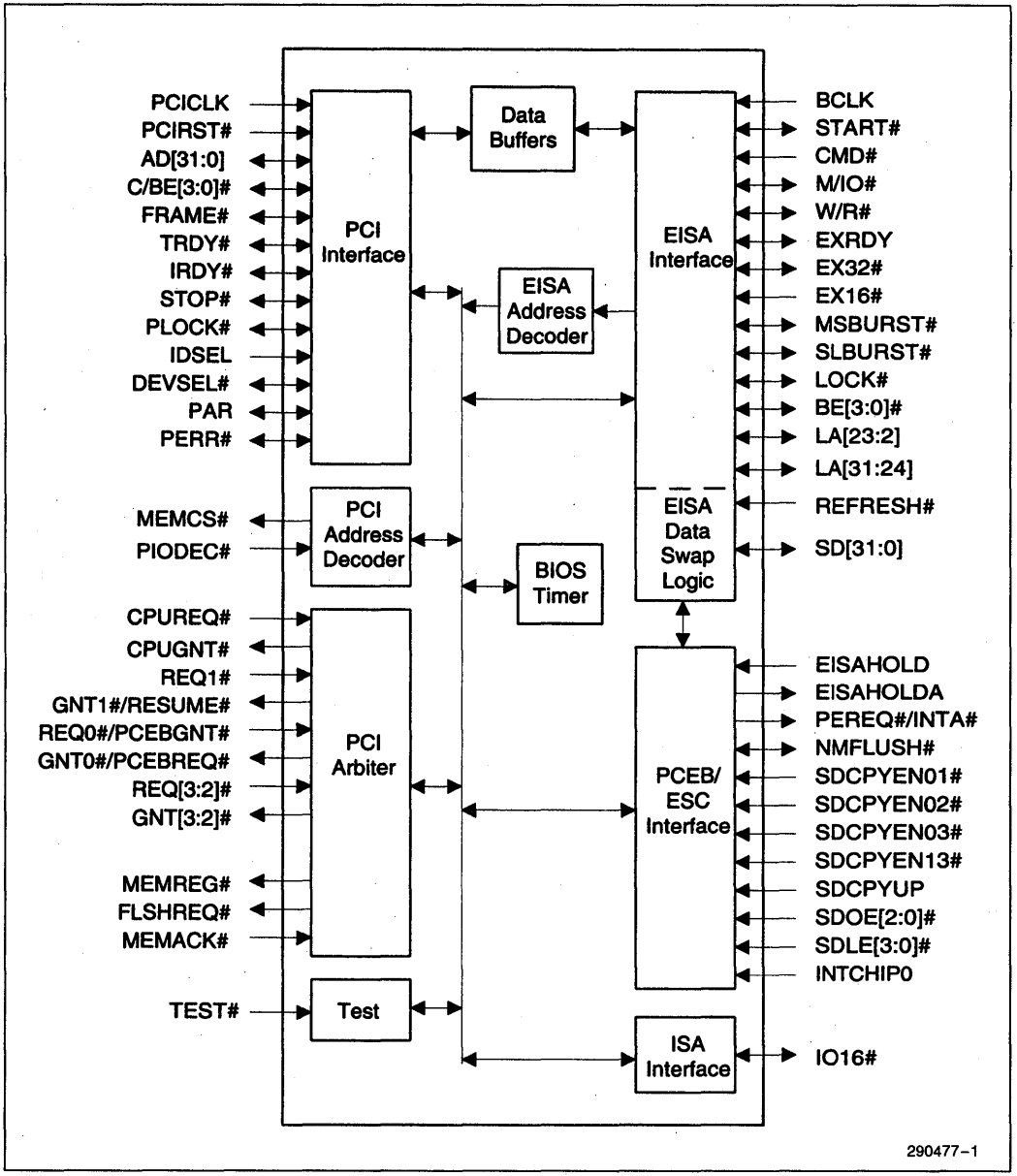


Figure 1-0. PCEB Block Diagram

1.0 ARCHITECTURAL OVERVIEW

The PCI-EISA bridge chipset provides an I/O subsystem core for the next generation of high-performance personal computers (e.g., those based on the Intel486™ or Pentium™ processor). System designers can take advantage of the power of the PCI (Peripheral Component Interconnect) for the local I/O 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 chipset 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 1-0 shows a block diagram of a typical system using the PCI-EISA Bridge chipset. 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

This bus hierarchy allows concurrency for simultaneous operations on all three bus environments. Data buffering permits concurrency for operations that crossover into another bus environment. For example, a PCI device could post data destined to EISA into the PCEB. This permits the PCI Bus transaction to complete in a minimum time, freeing up the PCI Bus for further transactions. The PCI device does not have to wait for the transfer to complete to its final destination. Meanwhile, any ongoing EISA Bus transactions are permitted to complete. The posted data is then transferred to its EISA Bus destination when the EISA Bus is available. The PCI-EISA Bridge chipset implements extensive buffering for PCI-to-EISA and EISA-to-PCI bus transactions. In addition to concurrency for the operations that cross bus environments, data buffering allows the fastest operations within a particular bus environment (via PCI burst transfers and EISA burst transfers).

The PCI with 132 MByte/sec and EISA with 33 MByte/sec peak data transfer rates represent bus environments with significantly different bandwidths. Without buffering, transfers that cross the single bus environment are performed at the speed of the slower bus. Data buffers provide a mechanism for data rate adoption so that the operation of the fast bus environment (PCI), i.e., usable bandwidth, is not significantly impacted by the slower bus environment (EISA).

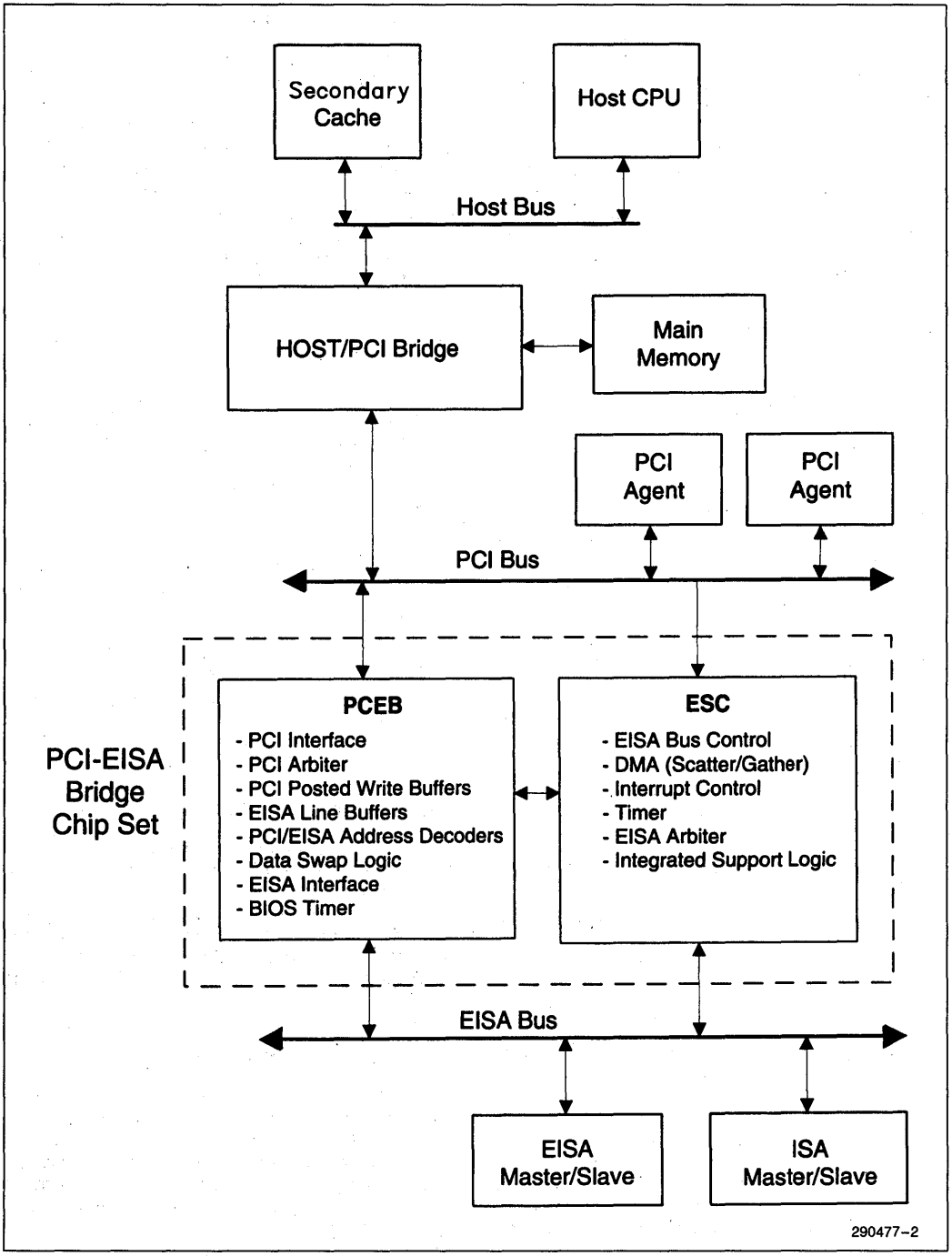


Figure 1-1. PCI-EISA Chip Set System Block Diagram

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 slave registers 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

System partitioning shown in Figure 1-1 illustrates how the PCI can be used as a common interface between different portions of a system platform that are typically supplied by the chipset 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 chipset (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 1-1 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 chipset integrates central bus functions on both the PCI and EISA Buses. For the PCI, the functions include PCI bus arbitration and default bus driver. For the EISA Bus, central functions include the EISA Bus controller and EISA arbiter that are integrated in the ESC component and EISA data swap buffers that are integrated in the PCEB.

Integrated System Functions

The PCI-EISA Bridge chipset integrates system functions including PCI parity and system error reporting, buffer coherency management protocol, 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 MHz to 33 MHz. For PCI-initiated transfers, the PCEB can only be a slave. The PCEB becomes a slave when it positively decodes the cycle. The PCEB also becomes a slave for unclaimed cycles on the PCI Bus. These unclaimed cycles are either negatively or 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 and burst cycles for memory operations. Note that burst transfers cannot be performed to the PCEB's internal registers. Burst memory write cycles to the EISA Bus can transfer up to four Dwords, depending on available space in the PCEB's Posted Write Buffers. When space is no longer available in the buffers, the PCEB terminates the transaction. This supports the Incremental Latency Mechanism as defined in the PCI Specification. Note that, if the Posted Write Buffers are disabled, PCI burst operations are not performed and all transfers are single cycle.

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 signal (C/BE#) 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. The arbiter can be disabled, if an external arbiter is used.

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 is 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. For memory write operations through the Posted Write Buffers, the PCEB uses EISA burst transfers, if supported by the slave, to flush the buffers. Otherwise, single cycle transfers are used. Single cycle transfers are used for all I/O cycles and memory reads.

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

To isolate the slower EISA Bus from the PCI Bus, the PCEB provides two types of data buffers. Buffer management control guarantees data coherency.

Four Dword wide Posted Write Buffers permit posting of PCI-initiated memory write cycles to the EISA Bus. For EISA-initiated cycles to the PCI Bus, there are four 16-byte wide Line Buffers. 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 ± 1 μs.

1.2 ESC Overview

The PCEB and ESC form an 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-bit 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 is also responsible for generating 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.

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.

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.

s/o/d Sustained 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 MHz 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 and CPUREQ# is high, the PCEB drives the AD[31:0] signals to keep them from floating. If CPUREQ# is low and REQ1# is low, the PCEB does not drive AD[31:0] while PCIRST# is asserted. If CPUREQ# is low and REQ1# is high, the PCEB drives AD[31:0] while 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 ms, 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. As long as PCIRST# is asserted and CPUREQ# is high, the PCEB drives the AD[31:0] signals to keep them from floating. If CPUREQ# is low and REQ1# is low, the PCEB does not drive AD[31:0] while PCIRST# is asserted. If CPUREQ# is low and REQ1# is high, the PCEB drives AD[31:0] while PCIRST# is asserted. When the internal arbiter is enabled (CPUREQ# sampled high at the time when PCIRST# is negated), the PCEB acts as the central resource responsible for driving the AD[31:0] signals when no one is granted the PCI Bus and the bus is idle. When the internal arbiter is disabled, the PCEB does not drive AD[31:0] as the central resource. The PCEB is always responsible for driving AD[31:0] when it is granted the bus (PCEBGNT# and idle bus) and as appropriate when it participates in bus transactions. During reset, these signals are tri-stated.

2.1 PCI Bus Interface Signals (Continued)

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. As long as PCIRST# is asserted and CPUREQ# is high, the PCEB drives C/BE[3:0]# to keep them from floating. If CPUREQ# is low and REQ1# is low, the PCEB does not drive C/BE[3:0]# while PCIRST# is asserted. If CPUREQ# is low and REQ1# is high, the PCEB drives C/BE[3:0]# while PCIRST# is asserted. When the internal arbiter is enabled (CPUREQ# sampled high at the time when PCIRST# is negated), the PCEB acts as the central resource responsible for driving the C/BE[3:0]# signals when no device is granted the PCI Bus and the bus is idle. When the internal arbiter is disabled, the PCEB does not drive C/BE[3:0]# as the central resource. The PCEB is always responsible for driving C/BE[3:0]# when it is granted the bus (PCEBGNT# and idle bus) and as appropriate when it is the master of a transaction. During reset, these signals are tri-stated.
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, these signals are tri-stated.
TRDY#	s/t/s	**TARGET READY:** TRDY#, as an output, indicates the PCEB 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, these signals are tri-stated.
IRDY#	s/t/s	**INITIATOR READY:** IRDY#, as an output, indicates the PCEB 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, these signals are 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, these signals are tri-stated.

2.1 PCI Bus Interface Signals (Continued)

Pin Name	Type	Description
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.
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, negative, 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 or not to negatively or 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. As long as PCIRST# is asserted and CPUREQ# is high, the PCEB drives the PAR signal to keep it from floating. If CPUREQ# is low and REQ1# is low, the PCEB does not drive PAR while PCIRST# is asserted. If CPUREQ# is low and REQ1# is high, the PCEB drives PAR while PCIRST# is asserted. When the internal arbiter is enabled (CPUREQ# sampled high at the time when PCIRST# is negated), 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. When the internal arbiter is disabled, the PCEB does not drive PAR as the central resource. The PCEB is always responsible for driving PAR when it is granted the bus (PCEBGNT# and idle bus) and as appropriate when it is the master of a transaction. 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. During reset, this signal is tri-stated.
PERR#	s/o/d	**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# has the following functions: 1. CPUREQ# is used to determine if the PCEB needs to drive the AD, C/BE#, and PAR signals while PCIRST# is asserted. As long as PCIRST# is asserted and CPUREQ# is high, the PCEB drives the AD, C/BE#, and PAR signals to keep them from floating. If CPUREQ# is low and REQ1# is low, the PCEB does not drive these signals while PCIRST# is asserted. If CPUREQ# is low and REQ1# is high, the PCEB drives these signals while PCIRST# is asserted. 2. If CPUREQ# is sampled high when PCIRST# makes a low-to-high transition, the internal arbiter is enabled. If it is sampled low, the internal arbiter is disabled. This requires system logic to drive the CPUREQ# during PCIRST# to determine the PCI arbiter configuration. 3. When the PCEB's internal PCI arbiter is enabled, this signal, if asserted, indicates that the Host CPU requests use of the PCI Bus. If the internal arbiter is disabled, this signal is meaningless after reset.
REQ1#	in	**REQUEST-1:** REQ1# provides the following two functions: 1. REQ1# is used, along with CPUREQ#, to determine if the PCEB will drive AD, C/BE#, and PAR during reset. If CPUREQ# is low and REQ1# is low, the PCEB does not drive AD, C/BE#, and PAR while PCIRST# is asserted. If CPUREQ# is low and REQ1# is high while PCIRST# is asserted, the PCEB drives the AD, C/BE#, and PAR signals during reset. 2. If the PCEB's internal arbiter is enabled, this signal is configured as REQ1#. An active low assertion indicates that master-1 requests the PCI Bus. If the internal arbiter is disabled, this pin is meaningless after reset.
REQ0#/ PCEBGNT#	in	**REQUEST-0 OR PCEB GRANT:** REQ0#/PCEBGNT# is a dual function pin. If the PCEB's internal arbiter is enabled, this pin is configured as REQ0#. An active low assertion indicates that master-0 requests the PCI Bus. If the internal arbiter is disabled, this pin is configured as PCEBGNT# which, when asserted, indicates the external PCI arbiter has granted use of the bus to the PCEB.
REQ[2:3]#	in	**REQUEST:** If the PCEB's internal arbiter is enabled, these signals are configured as REQ2# and REQ3#. A bus master asserts the corresponding request signal to request the PCI Bus. If the internal arbiter is disabled, these signals are meaningless after reset.
CPUGNT#	out	**CPU GRANT:** If the PCEB's internal arbiter is enabled, this signal is configured as CPUGNT#. The PCEB's internal arbiter asserts CPUGNT# to indicate that the CPU master (Host Bridge) has been granted the PCI Bus. If the internal arbiter is disabled, this pin is meaningless. During PCI reset, CPUGNT# is tri-stated.
GNT0#/ PCEBREQ#	out	**GRANT-0 OR PCEB REQUEST:** GNT0#/PCEBREQ# is a dual function pin. If the PCEB's internal arbiter is enabled, this pin is configured as GNT0#. The PCEB's internal arbiter asserts GNT0# to indicate that master-0 (REQ0#) has been granted the PCI Bus. If the internal arbiter is disabled, this pin is configured as PCEBREQ#. The PCEB asserts PCEBREQ# to request the PCI Bus. During PCI reset, this signal is tri-stated.

2.2 PCI Arbiter Signals (Continued)

Pin Name	Type	Description
GNT1#/ RESUME#	out	**GRANT-1 OR RESUME:** GNT1#/ RESUME# is dual function pin. If the PCEB's internal arbiter is enabled, this pin is configured as GNT1#. The PCEB's internal arbiter asserts GNT1# to indicate that master-1 has been granted the PCI Bus. If the internal arbiter is disabled, this pin is configured as RESUME#. The PCEB asserts RESUME# to indicate that the conditions causing the PCEB to retry the cycle have passed. During PCI reset, this signal is tri-stated.
GNT[2:3]#	out	**GRANT:** If the PCEB's internal arbiter is enabled, these pins are configured as GNT2# and GNT3#. The PCEB's internal arbiter drives these pins low to indicate that the corresponding initiator (REQ2# or REQ3#) 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#. MEMREQ# is high upon reset. **FLSHREQ# / MEMREQ# / Meaning** 1 / 1 / Idle 0 / 1 / Flush buffers pointing towards PCI to avoid ISA deadlock 1 / 0 / Reserved 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). This signal is synchronous to the PCI clock. During reset, this signal is high.
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. This signal is synchronous to the PCI clock. During reset, this signal is high.
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 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 high upon 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.

2.4 EISA Interface Signals (Continued)

Pin Name	Type	Description
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.
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 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#.

2.4 EISA Interface Signals (Continued)

Pin Name	Type	Description
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.
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.

2.4 EISA Interface Signals (Continued)

Pin Name	Type	Description
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.
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, 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. During PCIRST#, this signal is low.
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.
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.
INTCHIP0	t/s	**INTER CHIP 0:** INTCHIP0 is a reserved signal. The INTCHIP0 signal on the PCEB must be connected to the INTCHIP0 signal pin on the ESC for proper device operation.

2.6 PCEB/ESC Interface Signals (Continued)

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 signals 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 0 SD[15:8] SDLE2#: Latch Byte Lane 0 SD[23:16] SDLE3#: Latch Byte Lane 0 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 3-1) are located in PCI configuration space and are only accessible from the PCI Bus. The addresses shown in Table 3-1 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 3-1).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.

3.1 Configuration Registers

Table 3-1 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 3-1. 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 3-1. Configuration Registers

Configuration Address Offset	Abbreviation	Register Name	Access
00h-01h	VID	Vendor Identification	RO
02h-03h	DID	Device Identification	RO
04h-05h	PCICMD	Command Register	R/W
06h-07h	PCISTS	Status Register	RO, R/WC
08h	RID	Revision Identification	RO
09h-0Ch	—	Reserved	—
0Dh	MLTIM	Master Latency Timer	R/W
0Eh-3Fh	—	Reserved	—
40h	PCICON	PCI Control	R/W
41h	ARBCON	PCI Arbiter Control	R/W
42h	ARBPRI	PCI Arbiter Priority Control	R/W
43h	—	Reserved	—
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
48h-49h	EADC1	EISA Address Decode Control 1	R/W
4Ah-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	—
5Ch	EPMRA	EISA-to-PCI Memory Region Attributes	R/W
5Dh-5Fh	—	Reserved	—
60h-6Fh	MEMREGN[4:1]	EISA-to-PCI Memory Region Address (4 registers)	R/W
70h-77Fh	IOREGN[4:1]	EISA-to-PCI I/O Region Address (4 registers)	R/W
80h-81h	BTMR	BIOS Timer Base Address	R/W
84h	ELTCR	EISA Latency Timer Control Register	R/W
85h-87h	—	Reserved	—
88h-8Bh	PTCR	PCEB Test Control Register—***DO NOT WRITE!***	—
8Ch-FFh	—	Reserved	—

3.1.1 VID—VENDOR IDENTIFICATION REGISTER

Register Name: Vendor Identification
Address Offset: 00h-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.

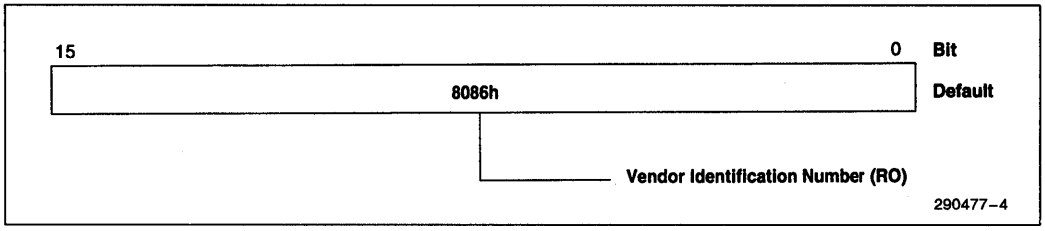


Figure 3-1. Vendor Identification Register

Table 3-2. Vendor Identification Register

Bit	Description
15:0	**VENDOR IDENTIFICATION NUMBER:** This is a 16-bit value assigned to Intel.

3.1.2 DID—DEVICE IDENTIFICATION REGISTER

Register Name: Device Identification
Address Offset: 02h-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.

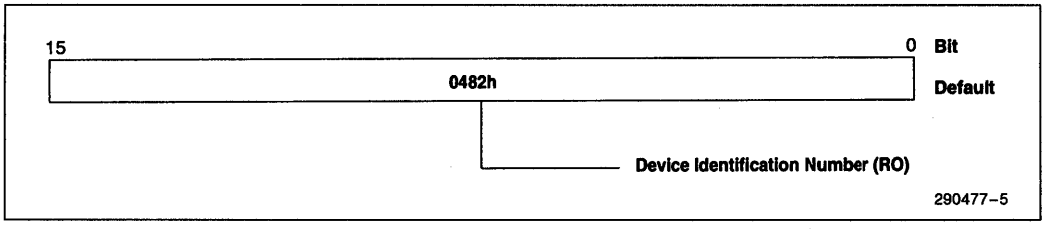


Figure 3-2. Device Identification Register

Table 3-3. Device Identification Register

Bit	Description
15:0	**DEVICE IDENTIFICATION NUMBER:** This is a 16-bit value assigned to the PCEB.

3.1.3 PCICMD—PCI COMMAND REGISTER

Register Name: PCI Command
Address Offset: 04h-05h
Default Value: 0007h
Attribute: Read/Write
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").

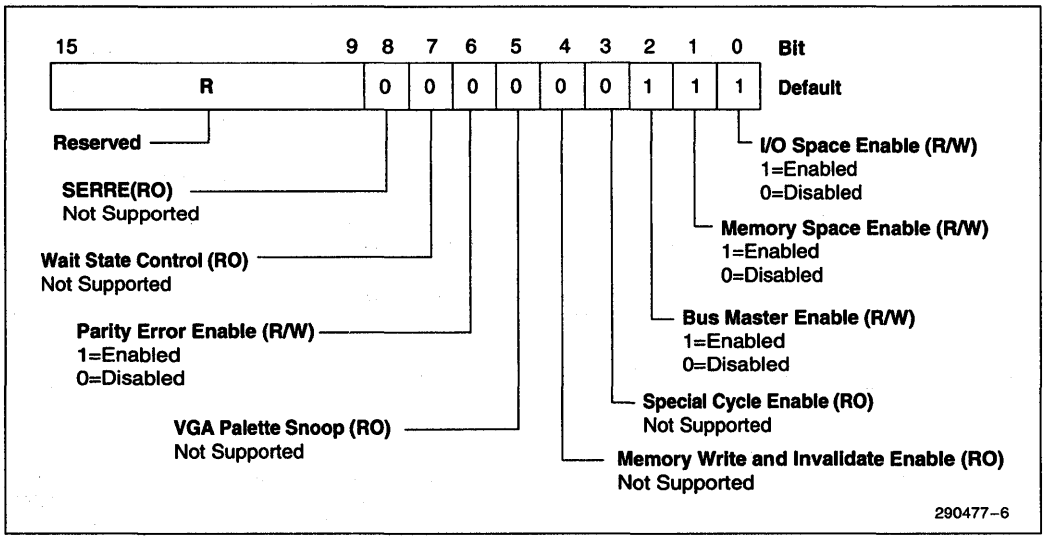


Figure 3-3. PCI Command Register

Table 3-4. PCI Command Register

Bit	Description
15:9	**RESERVED.**
8	**SERR# ENABLE (SERRE)—NOT SUPPORTED:** 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:** 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):** 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:** 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:** 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:** 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):** BME 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):** 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):** 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

Register Name: PCI Status
Address Offset: 06h-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").

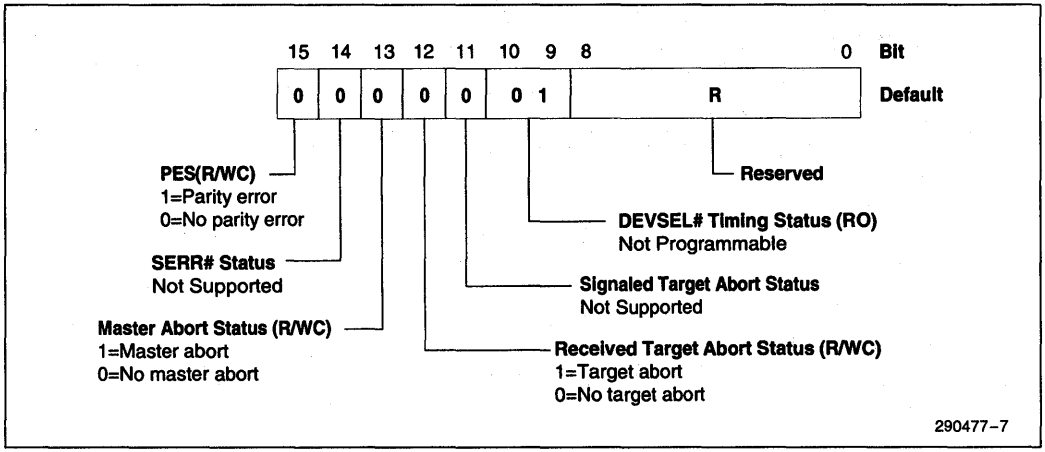


Figure 3-4. PCI Status Register

Table 3-5. PCI Status Register

Bit	Description
15	**PARITY ERROR STATUS (PERRS):** 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 IMPLEMENTED:** 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):** 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):** 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 IMPLEMENTED:** 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):** 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

Register Name: Revision Identification
Address Offset: 08h
Default Value: Revision Identification number
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.

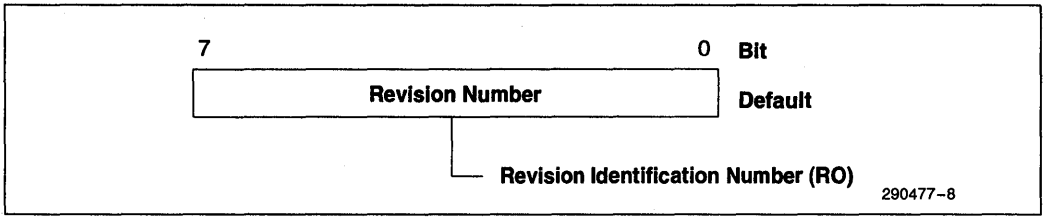


Figure 3-5. Revision Identification Register

Table 3-6. Revision Identification Register

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

Register Name: Master Latency Timer
Address Offset: 0Dh
Default Value: 08h
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.

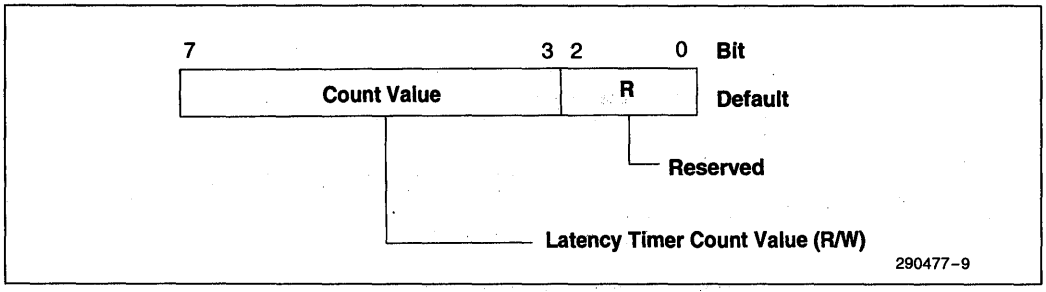


Figure 3-6. Master Latency Timer Register

Table 3-7. Master Latency Timer Register

Bit #	Bit Function
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. After PCIRST#, the default value of these bits is 00001b.
2:0	**RESERVED.**

3.1.7 PCICON—PCI CONTROL REGISTER

Register Name: PCI Control
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 and Posted Write 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.

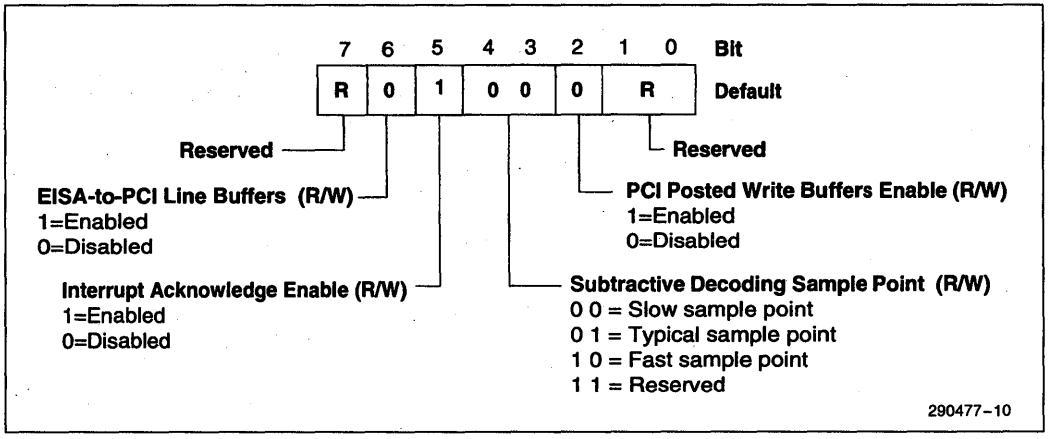


Figure 3-7. PCI Control Register

Table 3-8. PCI Control Register

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).
5	**INTERRUPT ACKNOWLEDGE ENABLE (IAE):** When IAE = 0, interrupt acknowledge is disabled and the PCEB ignores interrupt acknowledge cycles generated on the PCI Bus. (However, when this bit is 0, the 8259, residing in the ESC, can still be accessed.) When IAE = 1, the PCEB responds to interrupt acknowledge cycles in the normal fashion (i.e., uses the PEREQ#/INTA# signal to fetch the vector from the ESC). After PCIRST#, IAE = 1 (interrupt acknowledge cycles enabled).
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#. **Bit 4 / Bit 3 / Operation** 0 / 0 / Slow sample point - default value 0 / 1 / Typical sample point 1 / 0 / Fast sample point 1 / 1 / Reserved combination
2	**PCI POSTED WRITE BUFFER ENABLE (PPBE):** When PPBE = 1, the PCI Posted Write Buffers are enabled. When PPBE = 0, the PCI Posted Write Buffers are disabled. After PCIRST#, the PCI Posted Write Buffers are disabled (PPBE = 0).
1:0	**RESERVED.**

3.1.8 ARBCON—PCI ARBITER CONTROL REGISTER

Register Name: PCI Arbiter Control
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.

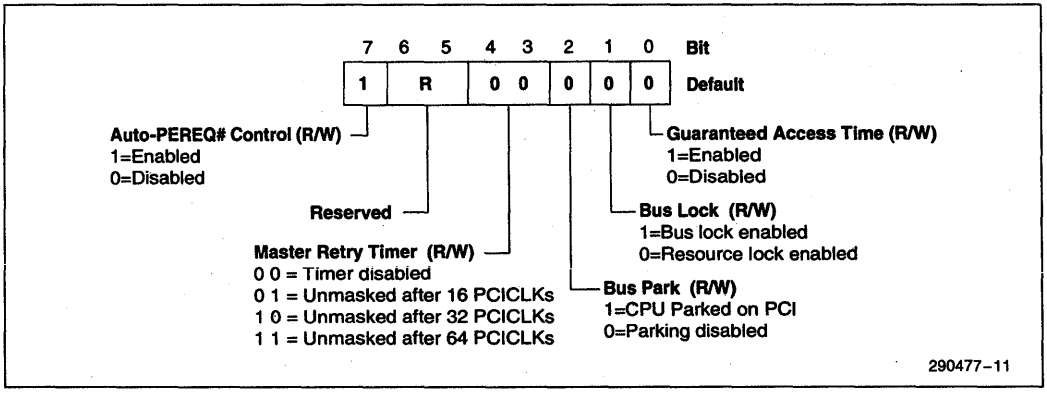


Figure 3-8. PCI Arbiter Control Register

Table 3-9. PCI Arbiter Control Register

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 below.
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. **Bit 4** **Bit 3** **Operation** 0 0 Timer disabled, Retries never masked—Default 0 1 Retries unmasked after 16 PCICLK's 1 0 Retries unmasked after 32 PCICLK's 1 1 Retries unmasked after 64 PCICLK's
2	**BUS PARK (BP):** When BP = 1, the PCEB parks 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, if the internal arbiter is enabled. 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).

NOTE:
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.

3.1.9 ARBPRI—PCI ARBITER PRIORITY CONTROL REGISTER

Register Name: PCI Arbiter Priority Control
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 modes—fixed priority, rotating priority and mixed priority modes. See Section 5.4, PCI Bus Arbitration for details on programming and using different arbitration modes.

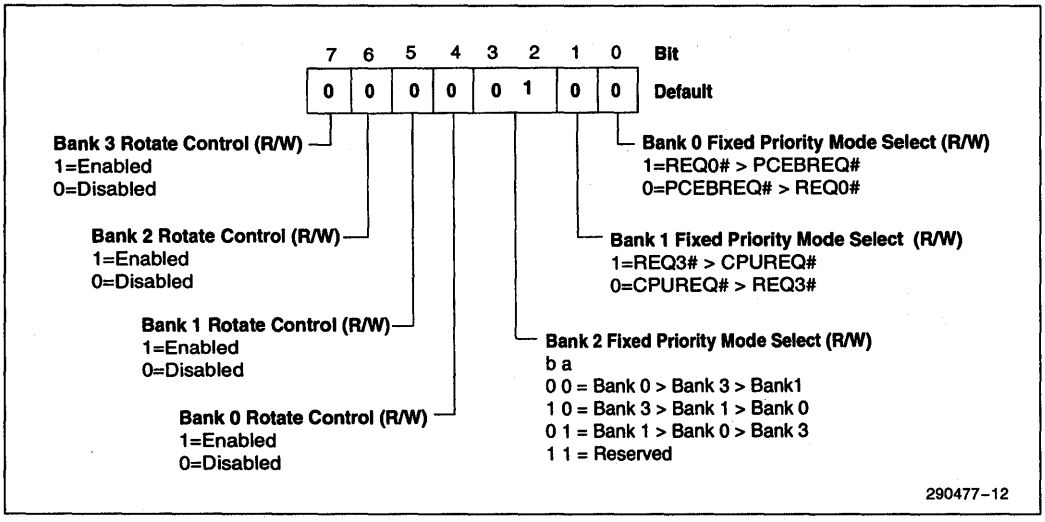


Figure 3-9. PCI Arbiter Priority Control Register

Table 3-10. PCI Arbiter Priority Control Register

Bit	Description
7	Bank 3 Rotate Control
6	Bank 2 Rotate Control
5	Bank 1 Rotate Control
4	Bank 0 Rotate Control
3:2	Bank 2 Fixed Priority Mode Select–b,a
1	Bank 1 Fixed Priority Mode Select
0	Bank 0 Fixed Priority Mode Select

3.1.10 MCSCON—MEMCS# CONTROL REGISTER

Register Name: MEMCS# Control
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).

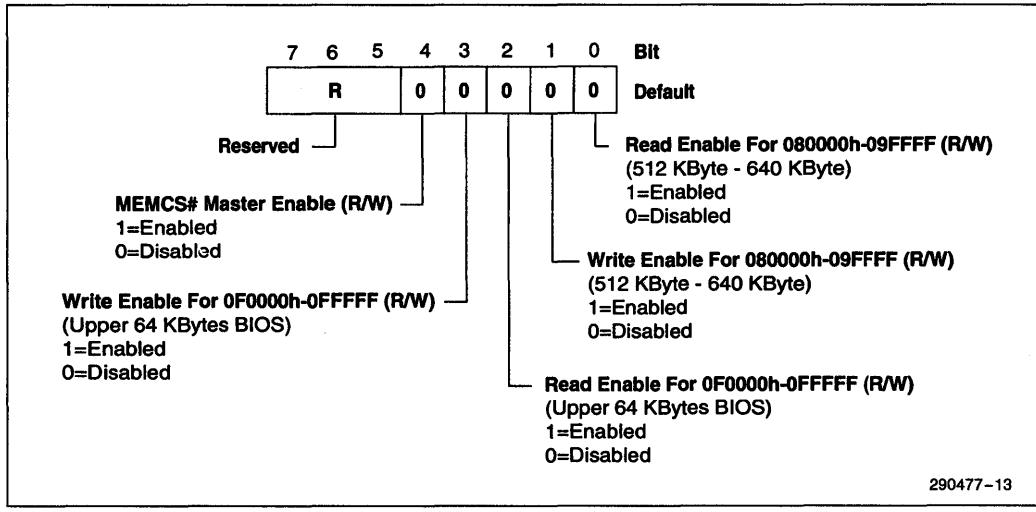


Figure 3-10. MEMCS# Control Register

Table 3-11. MEMCS# Control Register

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 0F0000h–0FFFFFh (UPPER 64 KBYTE BIOS):** When bit 3 = 1, the PCEB generates MEMCS# for PCI master memory write accesses to the address range 0F0000h–0FFFFFh. When bit 3 = 0, the PCEB does not generate MEMCS# for PCI master memory write accesses to the address range 0F0000h–0FFFFFh.
2	**READ ENABLE FOR 0F0000h–0FFFFFh (UPPER 64 KBYTE BIOS):** When bit 2 = 1, the PCEB generates MEMCS# for PCI master memory read accesses to the address range 0F0000h–0FFFFFh. When bit 2 = 0, the PCEB does not generate MEMCS# for PCI master memory read accesses to the address range 0F0000h–0FFFFFh.
1	**WRITE ENABLE FOR 080000h–09FFFFh (512 KBYTE–640 KBYTE):** When bit 1 = 1, the PCEB generates MEMCS# for PCI master memory write accesses to the address range 080000h–009FFFFh. When bit 1 = 0, the PCEB does not generate MEMCS# for PCI master memory write accesses to the address range 080000h–09FFFFh.
0	**READ ENABLE FOR 080000h–09FFFFh (512 KBYTE–640 KBYTE):** When bit 0 = 1, the PCEB generates MEMCS# for PCI master memory read accesses to the address range 080000h–09FFFFh. When bit 0 = 0, the PCEB does not generate MEMCS# for PCI master memory read accesses to the address range 080000h–09FFFFh.

3.1.11 MCSBOH—MEMCS# BOTTOM OF HOLE REGISTER

Register Name: MEMCS# Bottom of Hole
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.

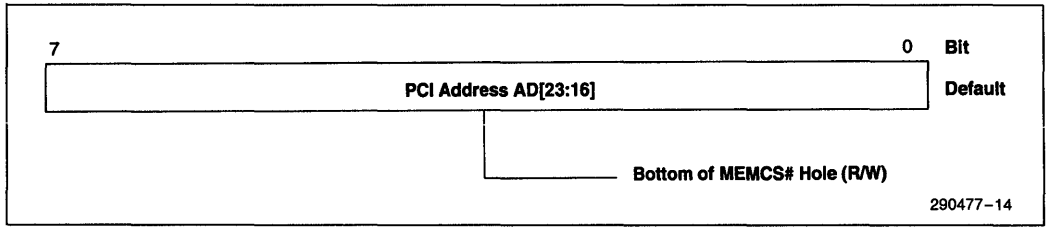


Figure 3-11. MEMCS# Bottom of Hole Register

Table 3-12. MEMCS# Bottom of Hole Register

Bit	Description
7:0	**BOTTOM OF MEMCS# HOLE:** Bits [7:0] correspond to address lines AD[23:16], respectively.

3.1.12 MCSTOH—MEMCS# TOP OF HOLE REGISTER

Register Name: MEMCS# Top of Hole
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.

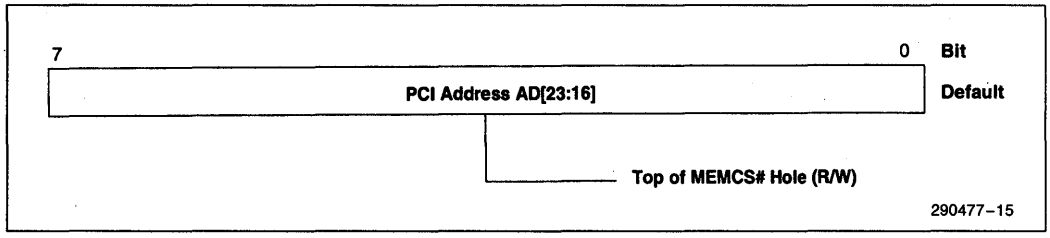


Figure 3-12. MEMCS# Top of Hole Register

Table 3-13. MEMCS# Top of Hole Register

Bit	Description
7:0	**TOP OF MEMCS# HOLE:** Bits [7:0] correspond to address lines AD[23:16], respectively.

3.1.13 MCSTOM—MEMCS# TOP OF MEMORY REGISTER

Register Name: MEMCS# Top of Memory
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).

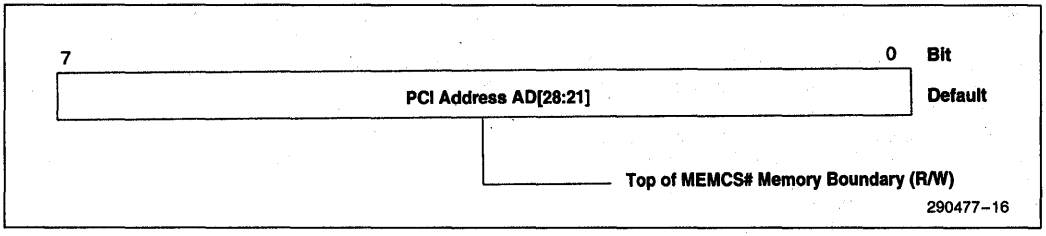


Figure 3-13. MEMCS# Top of Memory Register

Table 3-14. MEMCS# Top of Memory Register

Bit	Description
7:0	**TOP OF MEMCS# MEMORY BOUNDARY:** Bits [7:0] correspond to address lines AD[28:21], respectively.

3.1.14 EADC1—EISA ADDRESS DECODE CONTROL 1 REGISTER

Register Name: EISA Address Decode Control 1
Address Offset: 48h-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.

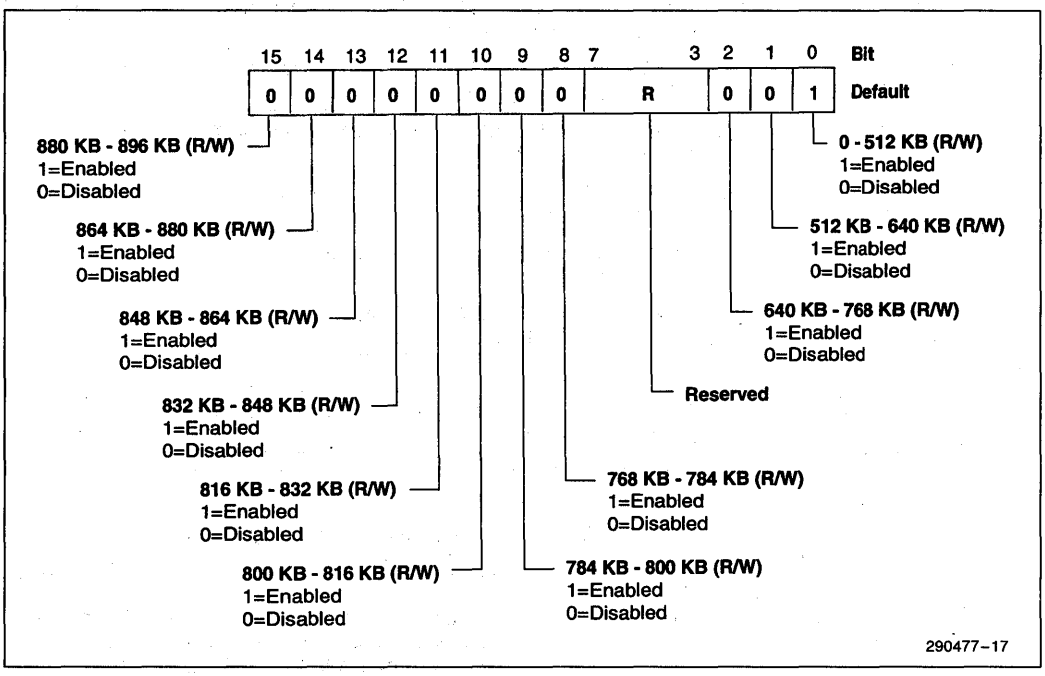


Figure 3-14. EISA Address Decode Control 1 Register

Table 3-15. EISA Address Decode Control 1 Register

Bit	Description
15	**880 KBYTES–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 KBYTES–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 KBTES–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 KBTES–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 KBTES–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 KBTES–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 KBTES–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 KBTES–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 KBTES–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 KBTES–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.

3.1.15 IORT—ISA I/O RECOVERY TIMER REGISTER

Register Name: ISA I/O Recovery Time
Address Offset: 4Ch
Default Value: 56h
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-bit and 16-bit recovery enabled with two clocks of I/O recovery.

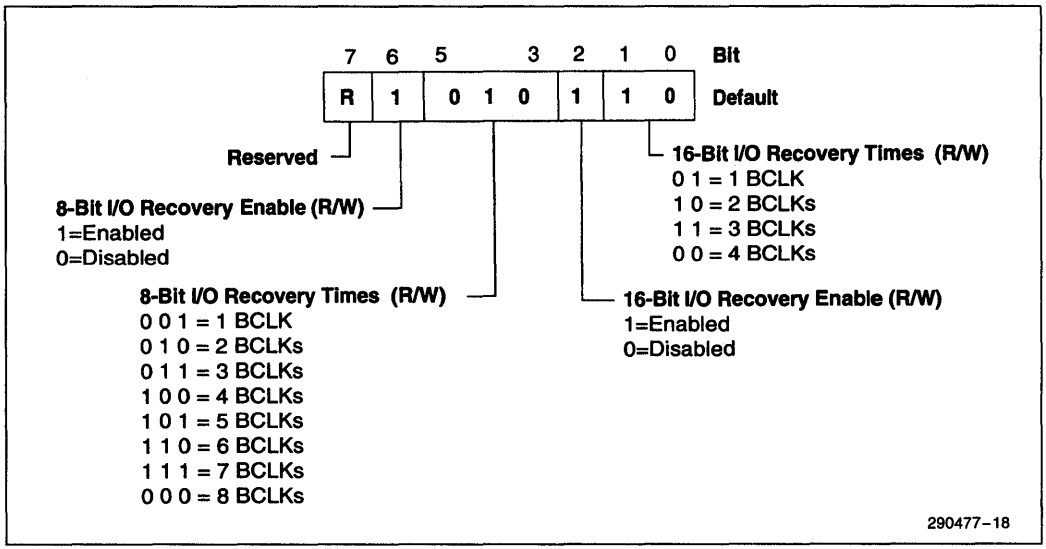


Figure 3-15. ISA Controller Recovery Timer Register

Table 3-16. ISA Controller Recovery Timer Register

Bit	Description
7	**RESERVED.**
6	**8-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. **Bit 5 / Bit 4 / Bit 3 / BCLK** 0 0 1 1 0 1 0 2 0 1 1 3 1 0 0 4 1 0 1 5 1 1 0 6 1 1 1 7 0 0 0 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. **Bit 1 / Bit 0 / BCLK** 0 1 1 1 0 2 1 1 3 0 0 4

3.1.16 MAR1—MEMCS# ATTRIBUTE REGISTER #1

Register Name: 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.

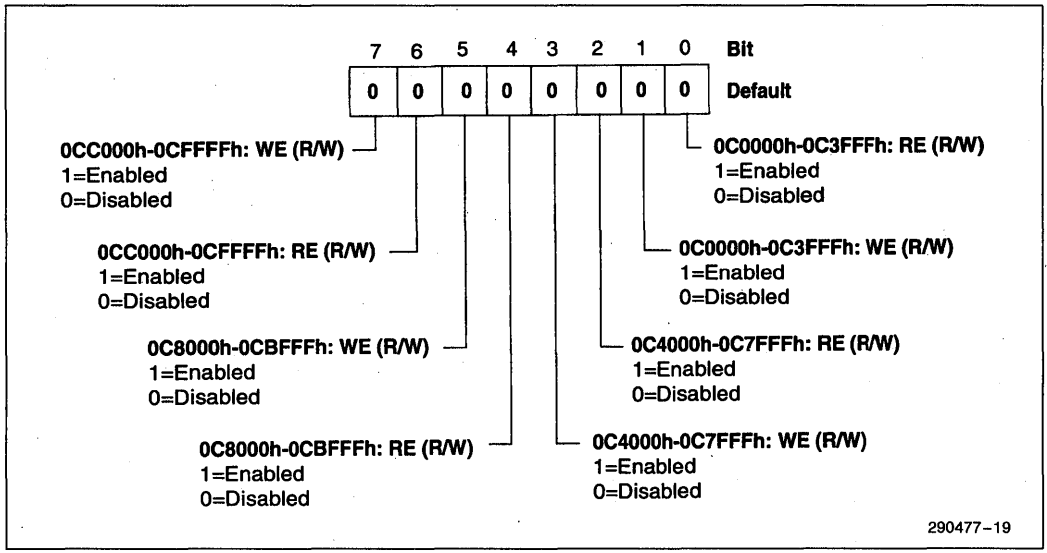


Figure 3-16. MEMCS# Attribute Register #1

Table 3-17. MEMCS# Attribute Register #1

Bit	Description
7	0CC000h–0CFFFFh Add-on BIOS: WE
6	0CC000h–0CFFFFh Add-on BIOS: RE
5	0C8000h–0CBFFFh Add-on BIOS: WE
4	0C8000h–0CBFFFh Add-on BIOS: RE
3	0C4000h–0C7FFFh Add-on BIOS: WE
2	0C4000h–0C7FFFh Add-on BIOS: RE
1	0C0000h–0C3FFFh Add-on BIOS: WE
0	0C0000h–0C3FFFh Add-on BIOS: RE

3.1.17 MAR2—MEMCS# ATTRIBUTE REGISTER #2

Register Name: 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.

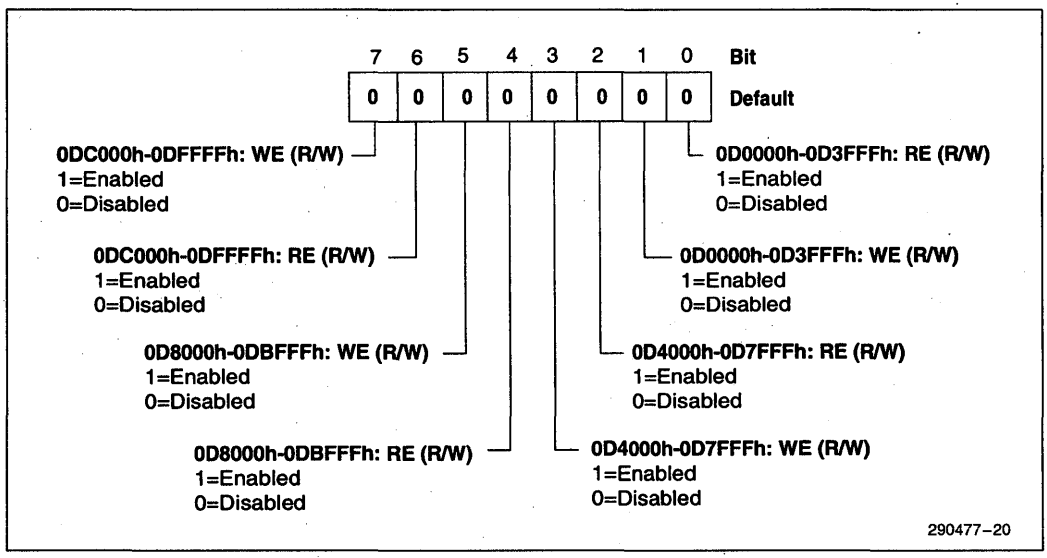


Figure 3-17. MEMCS# Attribute Register #2

Table 3-18. MEMCS# Attribute Register #2

Bit	Description
7	0DC000h–0DFFFFh Add-on BIOS: WE
6	0DC000h–0DFFFFh Add-on BIOS: RE
5	0D8000h–0DBFFFh Add-on BIOS: WE
4	0D8000h–0DBFFFh Add-on BIOS: RE
3	0D4000h–0D7FFFh Add-on BIOS: WE
2	0D4000h–0D7FFFh Add-on BIOS: RE
1	0D0000h–0D3FFFh Add-on BIOS: WE
0	0D0000h–0D3FFFh Add-on BIOS: RE

3.1.18 MAR3—MEMCS# ATTRIBUTE REGISTER #3

Register Name: 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.

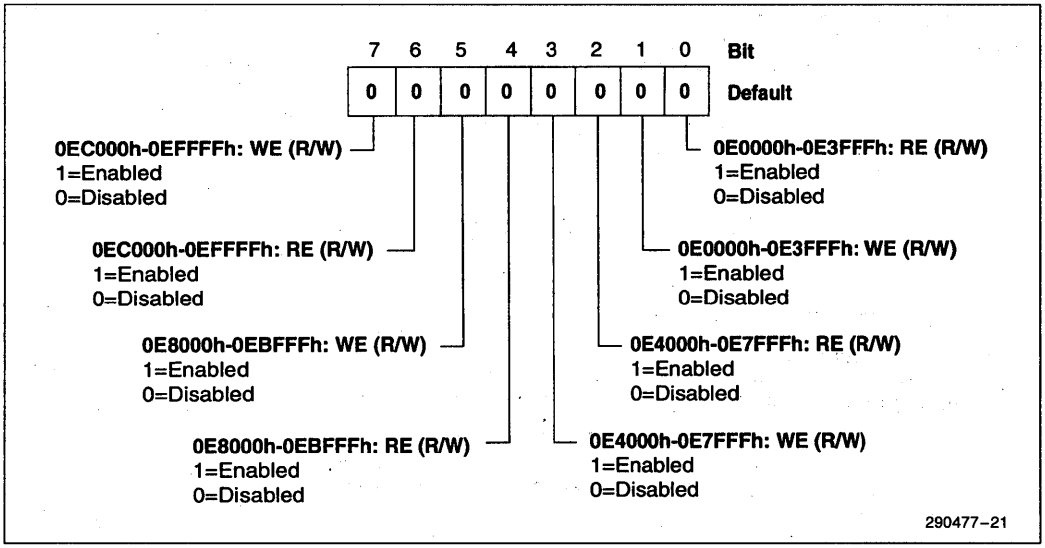


Figure 3-18. MEMCS# Attribute Register #3

Table 3-19. MEMCS# Attribute Register #3

Bit	Description
7	0EC000h–0EFFFFh BIOS Extension: WE
6	0EC000h–0EFFFFh BIOS Extension: RE
5	0E8000h–0EBFFFh BIOS Extension: WE
4	0E8000h–0EBFFFh BIOS Extension: RE
3	0E4000h–0E7FFFh BIOS Extension: WE
2	0E4000h–0E7FFFh BIOS Extension: RE
1	0E0000h–0E3FFFh BIOS Extension: WE
0	0E0000h–0E3FFFh BIOS Extension: RE

3.1.19 PDCON—PCI DECODE CONTROL REGISTER

Register Name: PCI Decode Control
Address Offset: 58h
Default Value: 00h
Attribute: Read/Write
Size: 8 bits

This register controls the mode of address decode (subtractive or negative) for memory cycles on the PCI Bus. This register is also used to enable/disable positive decode of PCI accesses to the IDE and 8259 locations residing in the expansion bus subsystem.

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
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, 0A0000h–0BFFFFh (as indicated by bit 2 of the EADC1 Register).

C. The four programmable PCI memory regions (define 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.

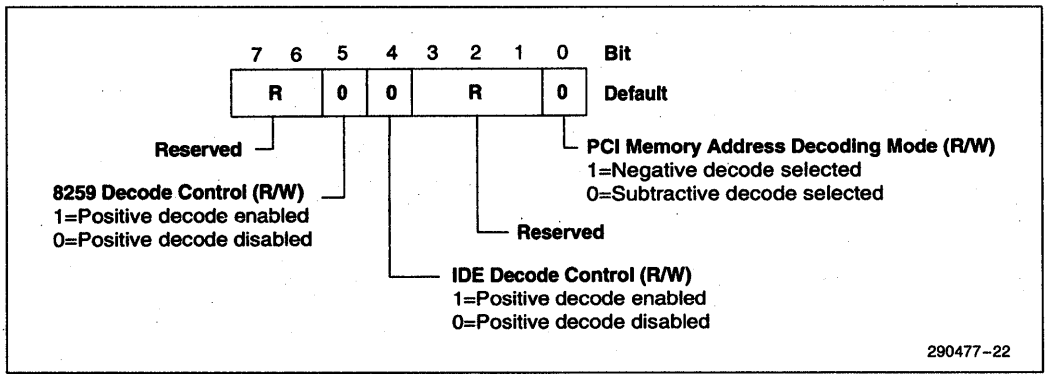


Figure 3-19. PCI Decode Control Register

Table 3-20. PCI Decode Control Register

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 1F0h–1F7h (primary) or 170h–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	**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.

3.1.20 EADC2—EISA ADDRESS DECODER CONTROL EXTENSION REGISTER

Register Name: EISA Address Decoder Control Extension
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 minus 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.

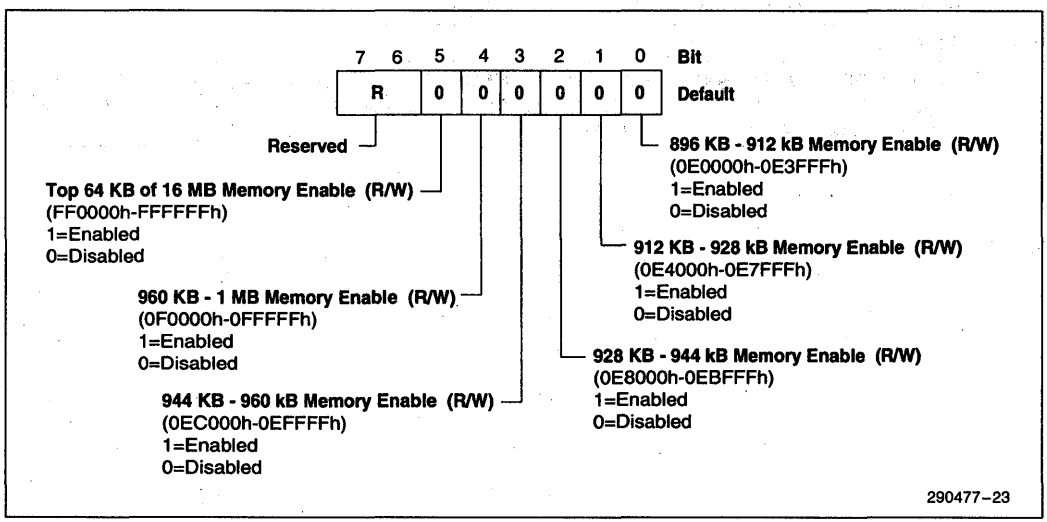


Figure 3-20. EISA Address Decoder Control Extension Register

Table 3-21. EISA Address Decoder Control Extension Register

Bit	Description
7:6	**RESERVED.**
5	**TOP 64 KBYTE OF 16 MBYTE MEMORY SPACE ENABLE (FF0000h–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 (0F0000h–0FFFFFh):** This memory block is enabled when this bit is 1 and disabled when this bit is 0.
3	**944 KBYTES–960 KBYTE MEMORY SPACE ENABLE (0EC000h–0EFFFFh):** This memory block is enabled when this bit is 1 and disabled when this bit is 0.
2	**928 KBYTES–944 KBYTE MEMORY SPACE ENABLE (0E8000h–0EBFFFh):** This memory block is enabled when this bit is 1 and disabled when this bit is 0.
1	**912 KBYTES–928 KBYTE MEMORY SPACE ENABLE (0E4000h–0E7FFFh):** This memory block is enabled when this bit is 1 and disabled when this bit is 0.
0	**896 KBYTES–912 KBYTE MEMORY SPACE ENABLE (0E0000h–0E3FFFh):** This memory block is enabled when this bit is 1 and disabled when this bit is 0.

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

Register Name: EISA-to-PCI Memory Region Attributes
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 run-time. Otherwise, data coherency can be affected, if a buffer containing valid write data is disabled and then, later, re-enabled.

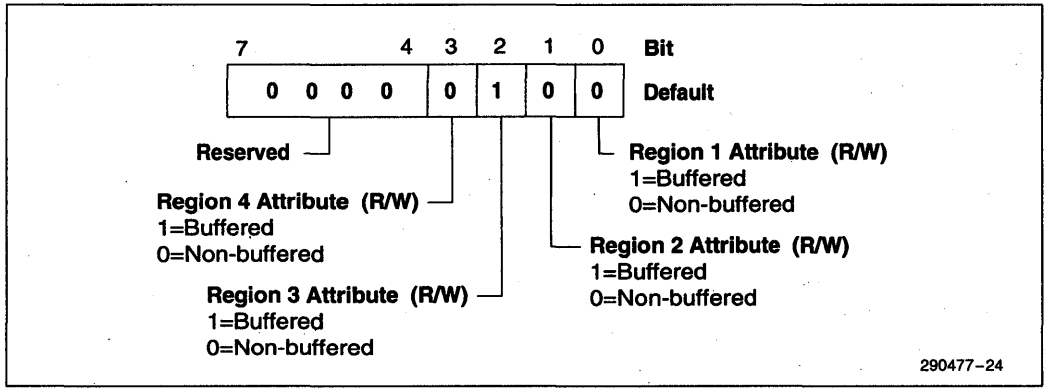


Figure 3-21. EISA-to-PCI Memory Region Attributes Register

Table 3-22. EISA-to-PCI Memory Region Attributes Register

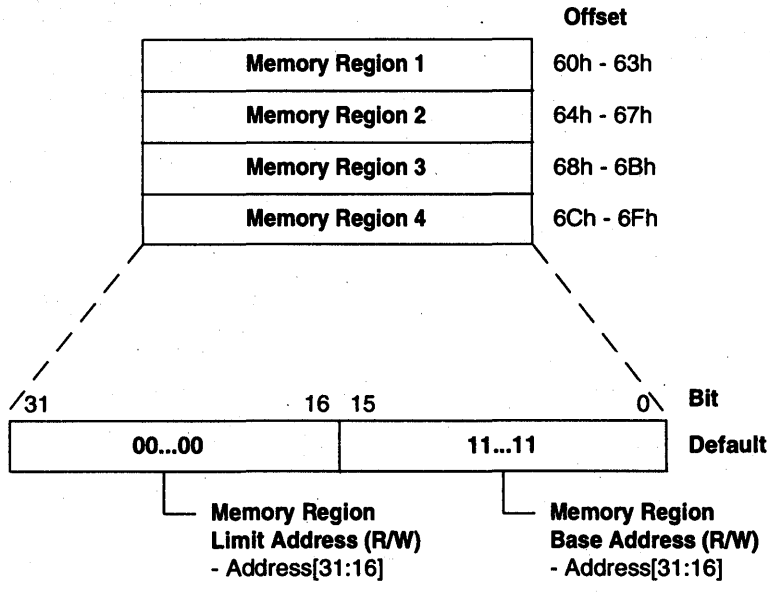
Bit	Description
7:4	**RESERVED.**
3	**REGION 4 ATTRIBUTE (REG-4):** EISA accesses to this PCI memory region are buffered when this bit is 1 and non-buffered when this bit is 0. If the Line Buffers are disabled via the PCICON Register (bit 6), buffering is disabled, regardless of the value of this bit.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

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

Register Name: EISA-to-PCI Memory Region Address
Address Offset: 60h-6Fh
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 GBytes. The default values of this register ensure that the regions are initially disabled.

A region is selected based on the following formula: Base Address ≤ address ≤ Limit Address.



NOTE:
The upper and lower words can be accessed individually.

Figure 3-22. EISA-to-PCI Memory Region Address Register

Table 3-23. EISA-to-PCI Memory Region Address Register

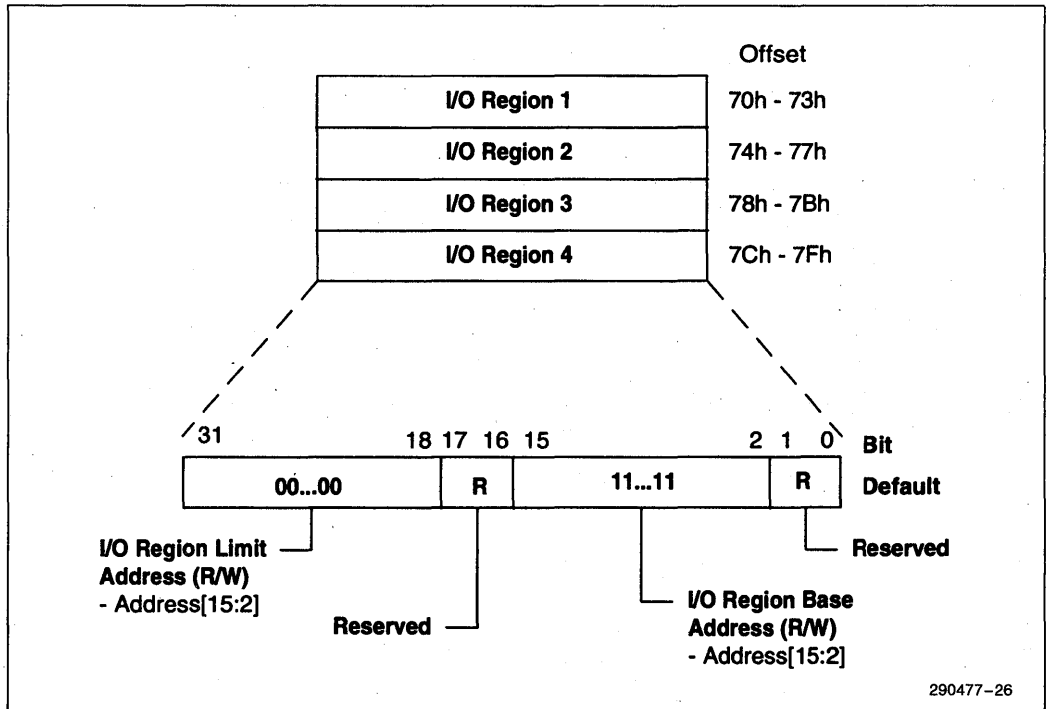
Bit	Description
31:16	**MEMORY REGION LIMIT ADDRESS:** For EISA-to-PCI accesses, bits [31:16] correspond to address lines LA[31:16] on the EISA Bus and AD[31:16] on the PCI Bus. This field determines the limit address of the memory region within the 4 GByte PCI memory space.
15:0	**MEMORY REGION BASE 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.

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

Register Name: EISA-to-PCI I/O Region Address
Address Offset: 70h-7Fh
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 IO regions are selected based on the following formula: Base Address ≤ address ≤ Limit Address.



NOTE:
The upper and lower words can be accessed individually.

Figure 3-23. EISA-to-PCI I/O Memory Region Address Register

Table 3-24. EISA-to-PCI I/O Region Address Register

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 64 KByte PCI I/O space.
1:0	**RESERVED.**

3.1.24 BTMR BIOS TIMER BASE ADDRESS REGISTER

Register Name: BIOS Timer Base Address
Address Offset: 80h-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.

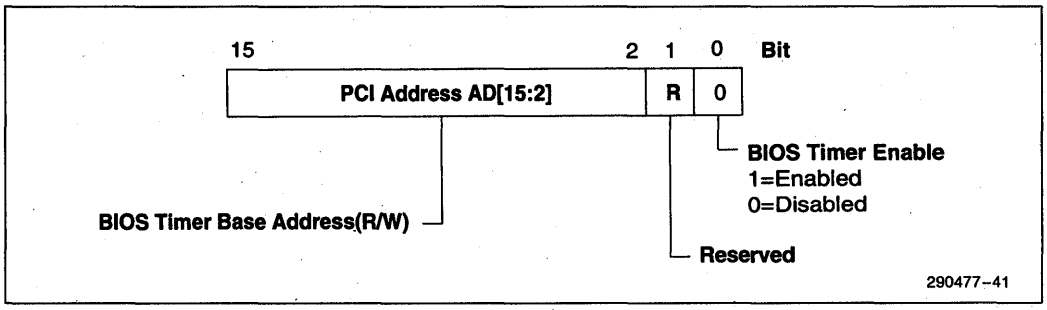


Figure 3-24. BIOS Timer Base Address Register

Table 3-25. BIOS Timer Base Address Register

Bit	Description
15:2	**BIOS TIMER BASE ADDRESS:** Bits [15:2] correspond to PCI address lines AD[15:2].
1	**RESERVED.**
0	**BTE (BIOS TIMER ENABLE):** When BTE = 1, the BIOS Timer is enabled. When BTE = 0, the BIOS Timer is disabled. The default is 0 (disabled).

3.1.25 ELTCR—EISA LATENCY TIMER CONTROL REGISTER

Register Name: EISA Latency Timer Control
Address Offset: 84h
Default Value: 00h
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:

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

where:

T_{pciclk} = 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 ms. 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 ms. 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.

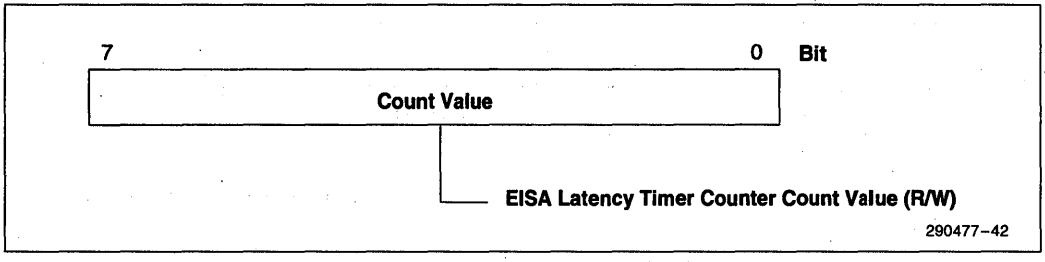


Figure 3-25. EISA Latency Timer Control Register

Table 3-26. EISA Latency Timer Control Register

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 BIOS TIMER REGISTER

Register Name: BIOS Timer
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 MHz or 8.33 MHz depending on the PCI clock. BCLK is divided by 8 to obtain the timer clock of 1.03 MHz 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.

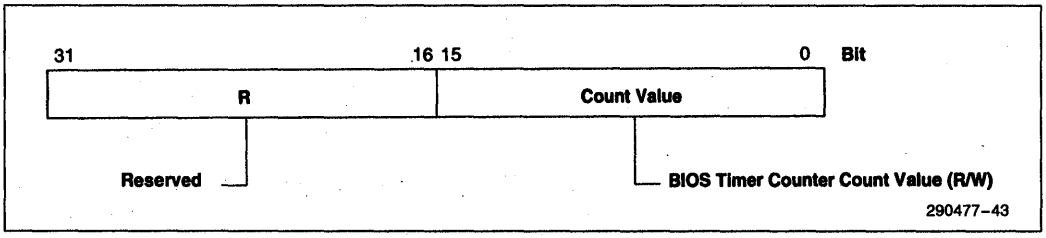


Figure 3-26. BIOS Timer Register

Table 3-27. BIOS Timer Register

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. 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.

The PCEB provides three methods for decoding the current PCI Bus cycle. The PCEB can use positive, subtractive, or negative 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) 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.

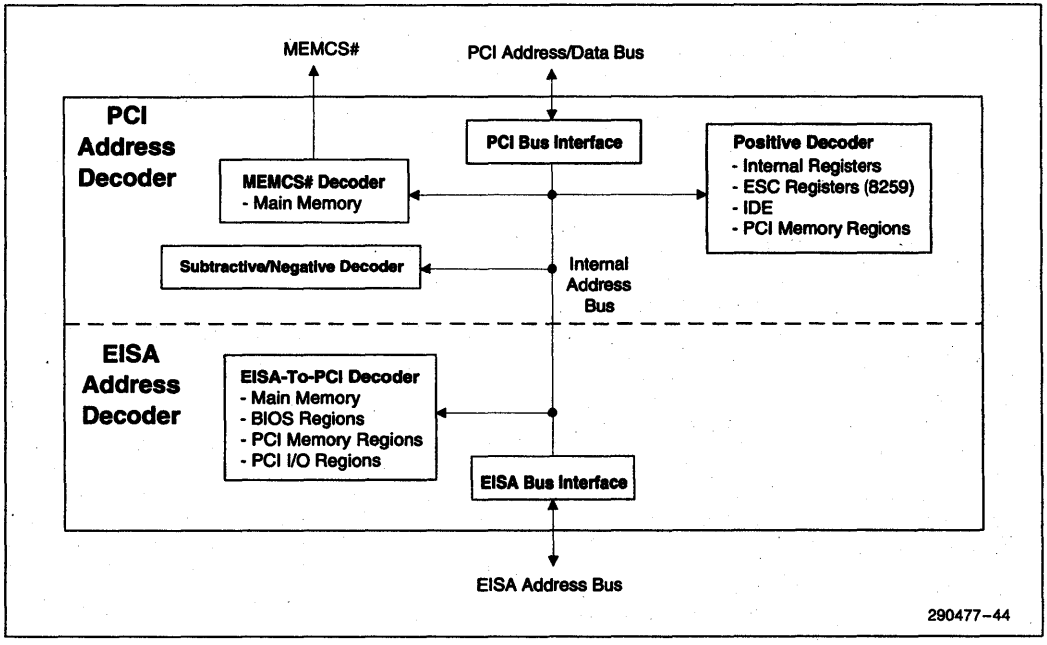


Figure 4-1. Block Diagram Of Address Decoder

3. **Negative decoding.** 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 1 PCICLK, negative decoding does not provide a latency improvement over subtractive decoding. However, for a 2 PCICLK time-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:
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:
   - Interrupt controller (8259) I/O registers contained within the ESC to optimize interrupt processing, if enabled through the PDCON Register.
   - 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 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

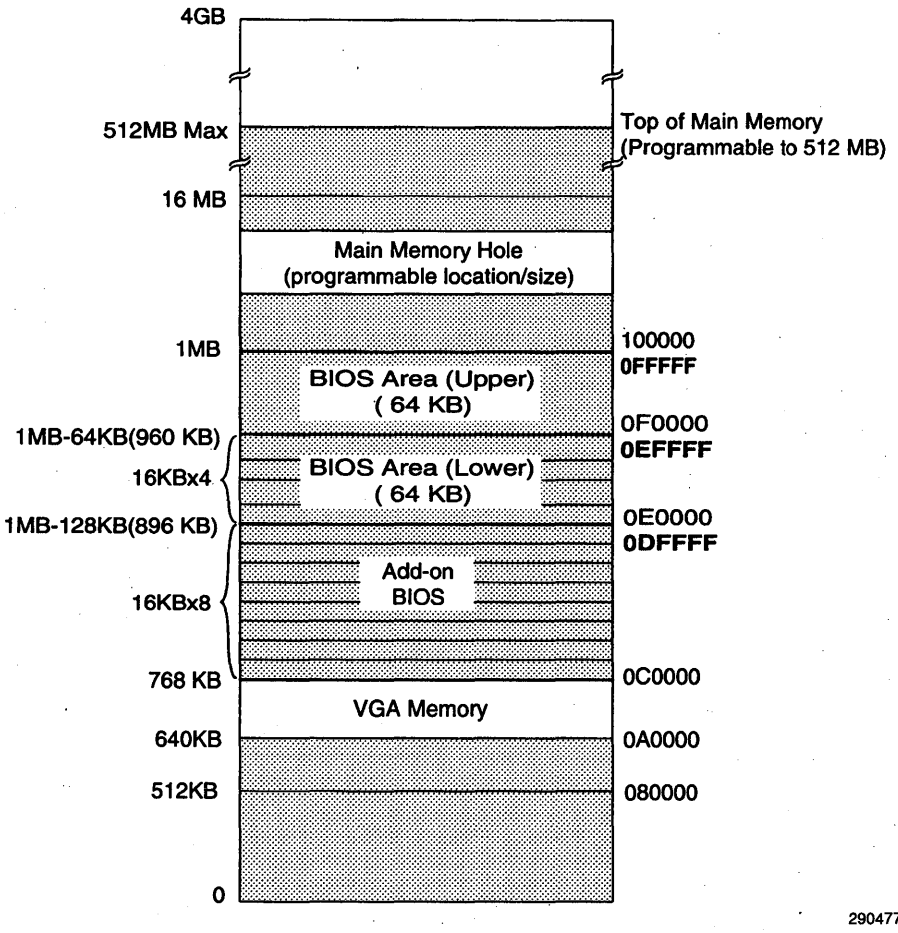
The MCSCON, MCSTOP, MCSBOH, MCSTOM, and PDCON Registers are used to program the decoding for PCI Bus memory cycles.

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 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 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 4-2). 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) decoded to the EISA Bus for a memory write.



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

Figure 4-2. MEMCS# Decode Areas

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

Table 4-1 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 KByte–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 4-1. Read Enable/Write Enable Attributes For MEMCS# Decoding

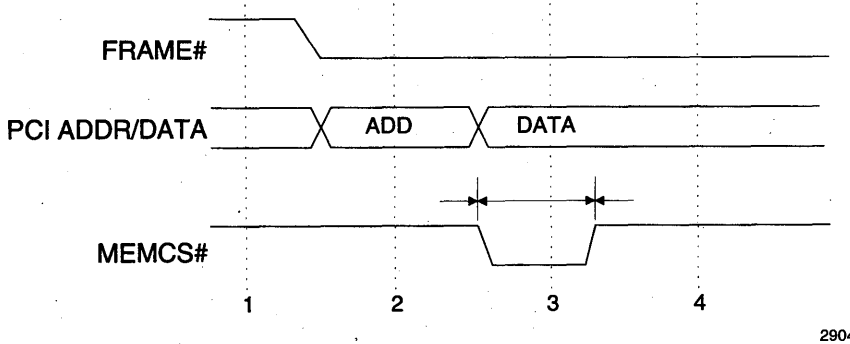
Memory Attribute Registers (Register bits are shown in brackets)	Attribute		Memory Segments	Comments
MCSCON[1:0]	WE	RE	080000h–09FFFFh	512K to 640K
MCSCON[3:2]	WE	RE	0F0000h–0FFFFFh	BIOS Area
MAR1[1:0]	WE	RE	0C0000h–0C3FFFh	Add-on BIOS
MAR1[3:2]	WE	RE	0C4000h–0C7FFFh	Add-on BIOS
MAR1[5:4]	WE	RE	0C8000h–0CBFFFh	Add-on BIOS
MAR1[7:6]	WE	RE	0CC000h–0CFFFFh	Add-on BIOS
MAR2[1:0]	WE	RE	0D0000h–0D3FFFh	Add-on BIOS
MAR2[3:2]	WE	RE	0D4000h–0D7FFFh	Add-on BIOS
MAR2[5:4]	WE	RE	0D8000h–0DBFFFh	Add-on BIOS
MAR2[7:6]	WE	RE	0DC000h–0DFFFFh	Add-on BIOS
MAR3[1:0]	WE	RE	0E0000h–0E3FFFh	BIOS Extension
MAR3[3:2]	WE	RE	0E4000h–0E7FFFh	BIOS Extension
MAR3[5:4]	WE	RE	0E8000h–0EBFFFh	BIOS Extension
MAR3[7:6]	WE	RE	0EC000h–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 4-3. 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.

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



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 4-3. MEMCS# Generation

(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 bits 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 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, the DEVSEL# sample point can be configured to three different settings by programming the PCICON Register. If the "fast" point is selected, the cycle is forwarded to EISA when DEVSEL# is inactive at the F sample point. If the "typical" point is selected, DEVSEL# is sampled on both F and 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 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.

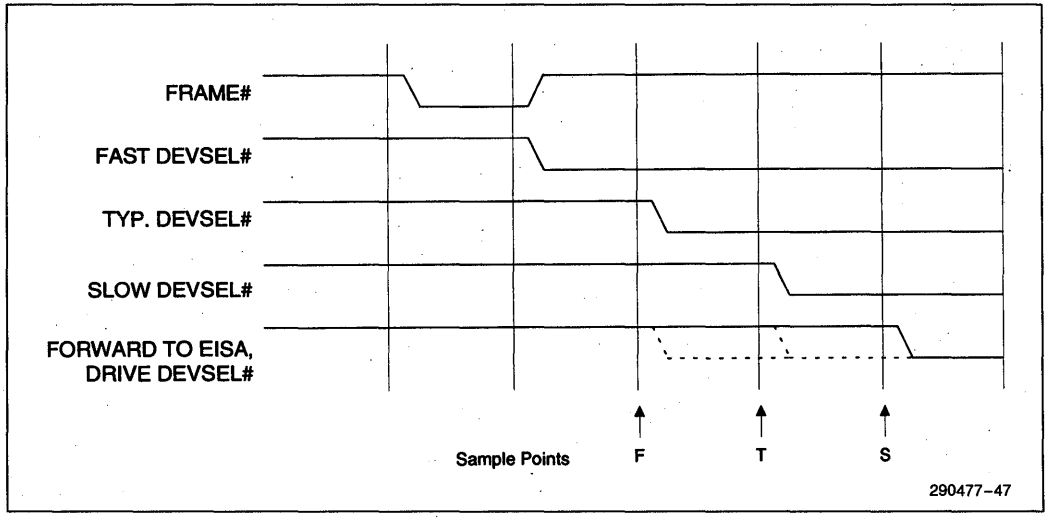


Figure 4-4. DEVSEL# Sample Points

Only unclaimed PCI cycles within the memory address range from 0 GByte to 4 GByte and I/O address range from 0 KByte 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.

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 EISA memory regions that are mapped to the PCI. Note that 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 either subtractive or negative decoding.

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, Region [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.

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 4-2 shows the 8259 I/O address map. After PCIRST#, positively decoded access to these address ranges is disabled.

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 4-3 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 4-2. ESC Resident Programmable Interrupt Controller (PIC) Registers

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

Table 4-3. EISA Resident IDE Registers

Address (hex)	Address 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
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
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 decoding are not used on the EISA Bus.

The PCEB permits several EISA memory address ranges (items a-i, below) 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 KByte to 512 KByte
b. 512 KByte to 640 KByte
c. 640 KByte to 768 KByte (VGA memory)
d. 768 KByte to 896 KByte in eight 16 KByte sections (Expansion ROM)
e. 896 KByte to 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 (FF0000h–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.

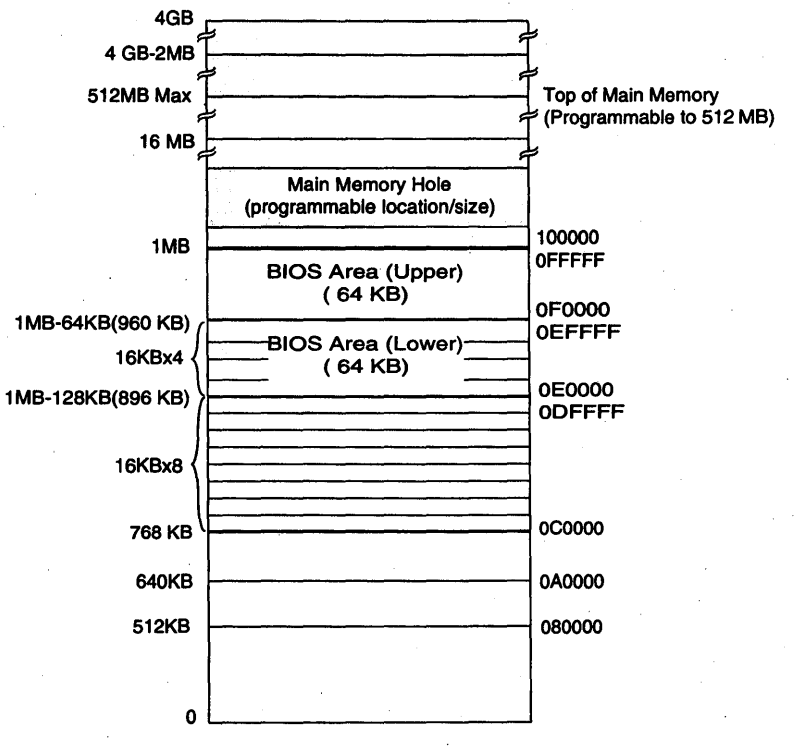
For detailed information on the PCEB registers used to control these address regions, refer to Section 3.1, PCEB Configuration Registers.

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 Line Buffer Operation:

- 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



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 region 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 4-5. EISA Address Decoder Map

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 (FF0000h–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:

Base_Address ≤ Address ≤ 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 address 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.

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.

NOTES:

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 5-1 summarizes the PCEB's support of the PCI Bus commands.

Table 5-1. 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 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

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.

Special Cycle

Target support: None.

Initiator support: None.

I/O Read

Target support:
Decode: Positive (PCEB and some ESC registers) and Subtractive
Data Path: Flow through
PCEB 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 (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 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.

I/O Write

Target support:
Decode: Positive (PCEB/ESC registers) and Subtractive
Data Path: Flow through
PCEB 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.

Memory Read

Target support:
Decode: Negative 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.

Memory Write

Target support:
Decode: Negative and Subtractive
Data Path: Posted Write Buffer or 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 subtrac-

tively decoding and forwarded to the EISA Bus. The memory write cycle is subject to retry. If the PCEB is locked, if the cycle triggers buffer management activity, if the PCI Posted Write Buffer (PWB) is full, or if posting is disabled because 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 cycle is posted, if the PCI Posted Write buffer is enabled, and the PCI cycle is terminated in zero wait states. If the PCI Posted Write Buffer is disabled, the PCEB holds the PCI Bus, in wait states, using TRDY# until the cycle is completed on the EISA Bus.

As a target, the PCEB only supports linear incrementing burst ordering that is signaled by the master with AD[1:0] = 00 during the address phase. Burst with any other type of ordering (AD[1:0] ≠ 00) is split into single data phase transfers using target disconnect.

Incremental burst memory writes destined for the EISA Bus are posted at 0 wait states until the PWB is full (4 Dwords). When the PWB is full, burst memory writes are target terminated using the disconnect semantics of the STOP# signal (Figure 5-12, Disconnect A/B).

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.

Configuration Read, Configuration Write
Target support:
Decode: via IDSEL pin
Data Path: Flow through
PCEB 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.

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.

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.

Memory Write and Invalidate

Target support:
PCEB 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 AD[1:0] during the address phase):

AD[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, both as a target and 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. In Figure 5-1, 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 5-2 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.

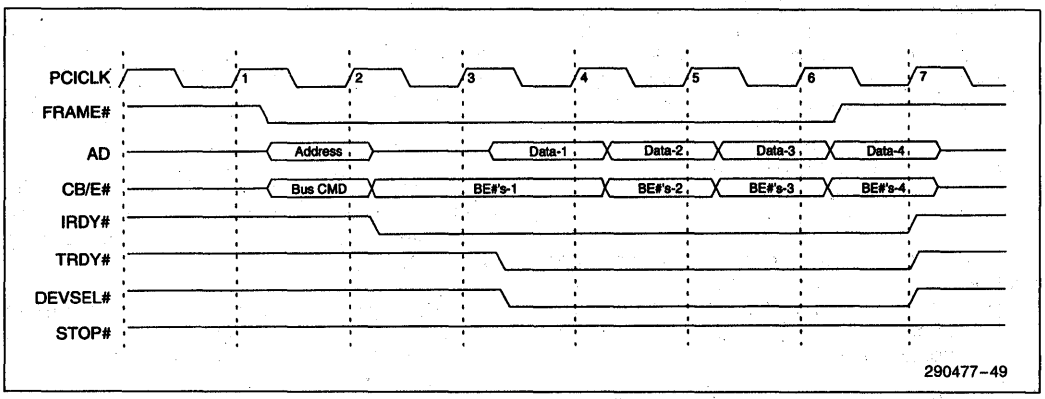


Figure 5-1. PCEB Burst Read From PCI Memory

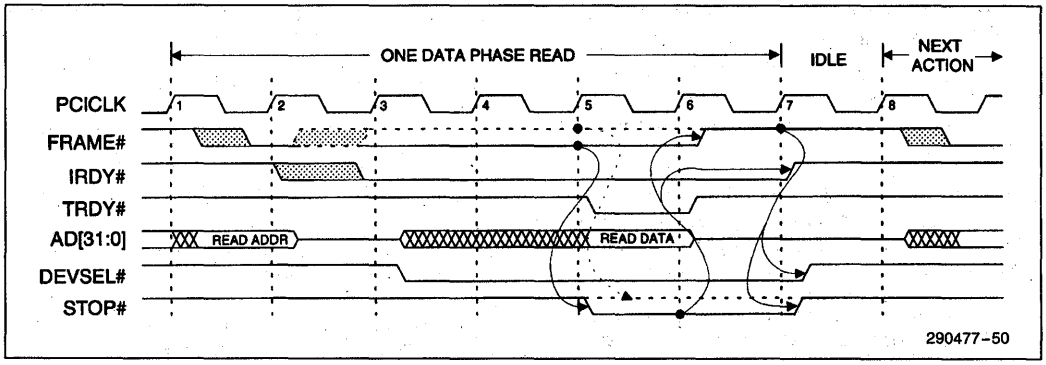


Figure 5-2. PCI Master Read from the PCEB (Burst with Target Termination)

5.1.4 BASIC READ

As a PCI master, the PCEB performs memory and I/O read transfers. Figure 5-1 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 5-1, 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 5-2 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 5-2 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 5-2). Note that, if the PCEB subtractively or negatively 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 (Figure 5-2). If a single transaction is indicated (FRAME# is sampled inactive during the first data phase), the PCEB asserts TRDY# without asserting STOP#.

5.1.5 BASIC WRITE

Figure 5-3 shows the PCEB, as a master, writing to PCI memory in zero wait states. Figure 5-4 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. If the internal Posted Write Buffer (PWB) is disabled during a burst memory write, the PCEB always target terminates after the first data phase. When the PWB is enabled, there is a space in the buffer, and the EISA Bus is owned by the PCEB, the write is posted until the PWB is filled. Additional data phases, when the PWB is full, causes the PCEB to terminate the transaction with a retry. For single write transactions that are not terminated with retry, the PCEB finishes the cycle in a normal fashion by asserting TRDY# without asserting STOP#.

Figure 5-4 shows the fastest PCEB response to a write cycle targeted to an internal PCI configuration register. During I/O write accesses to the EISA Bus or during memory writes to the EISA Bus when the PWB is disabled, the PCEB always adds wait states. The PCEB adds wait states by holding TRDY# high until the transfer on the EISA Bus is completed.

During a single memory write access to EISA memory when the PWB is enabled, the PCEB performs the access in a one wait state cycle (Note: This is due to timing constraints for address decoding that the PCI specification places on devices that can support 0 wait write operations for the first data phase.) During postable burst memory writes, only the first data phase has a wait state. The rest of the data phases (up to 4) are transferred in 0 wait states.

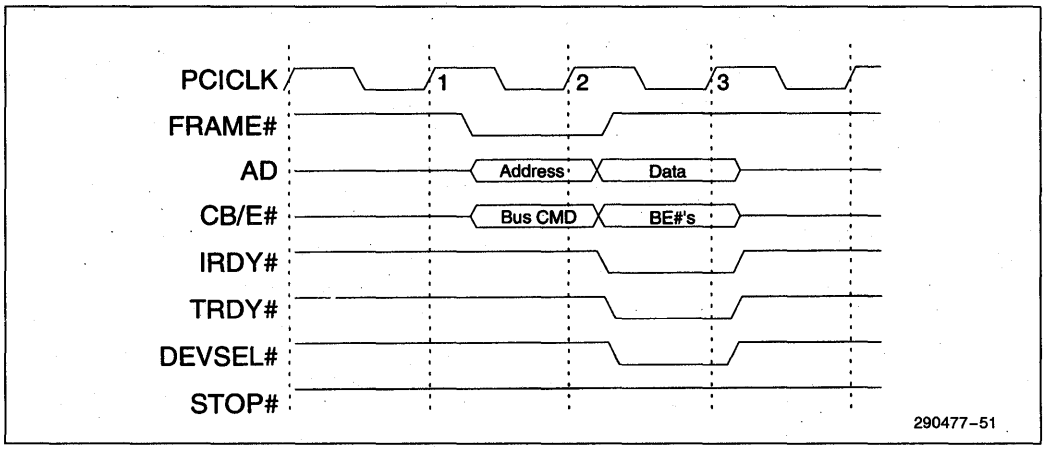


Figure 5-3. PCEB Write to PCI Memory

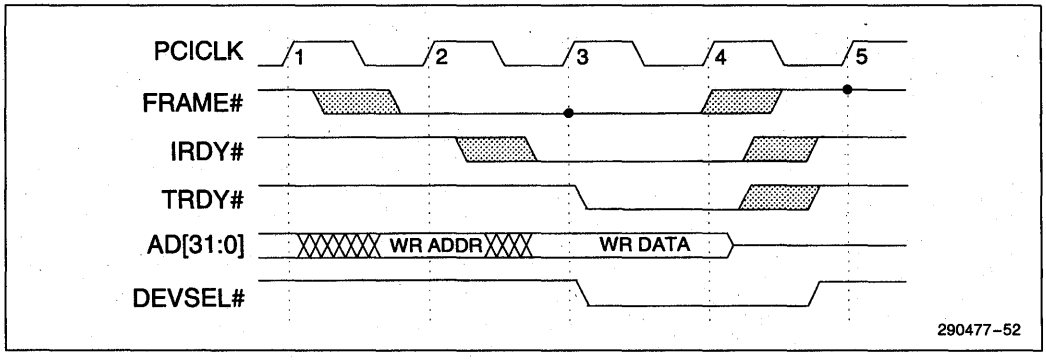


Figure 5-4. Fastest PCI Write to PCEB

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 5-5 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.

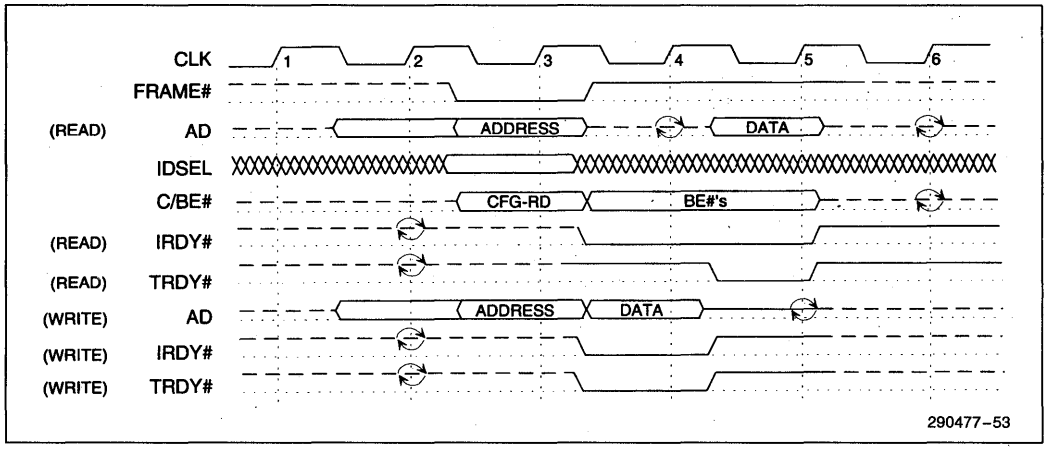


Figure 5-5. Configuration Cycle

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 5-6 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.

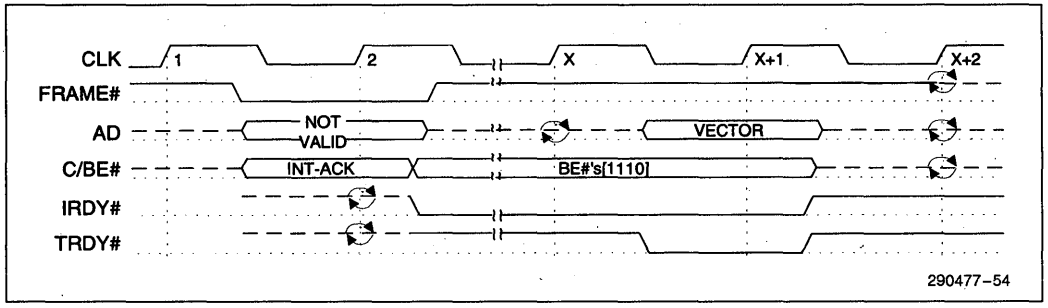


Figure 5-6. PCI Interrupt Acknowledge Cycle

5.1.8 EXCLUSIVE ACCESS

Refer to Figures 5-7, 5-8, and 5-9 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 the 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.

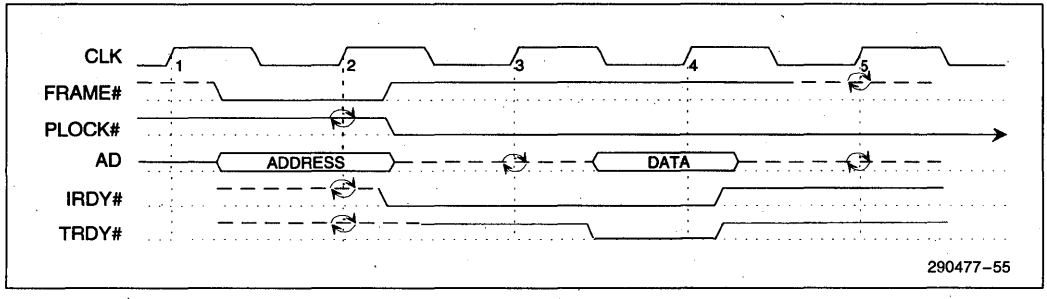


Figure 5-7. Beginning a Locked Cycle

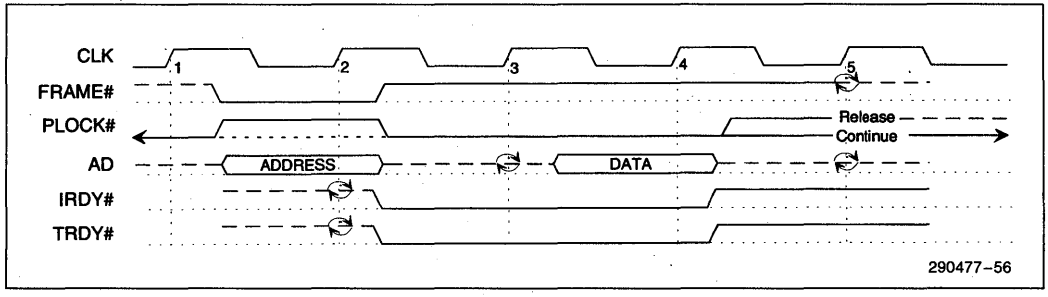


Figure 5-8. Continuing Locked Cycle

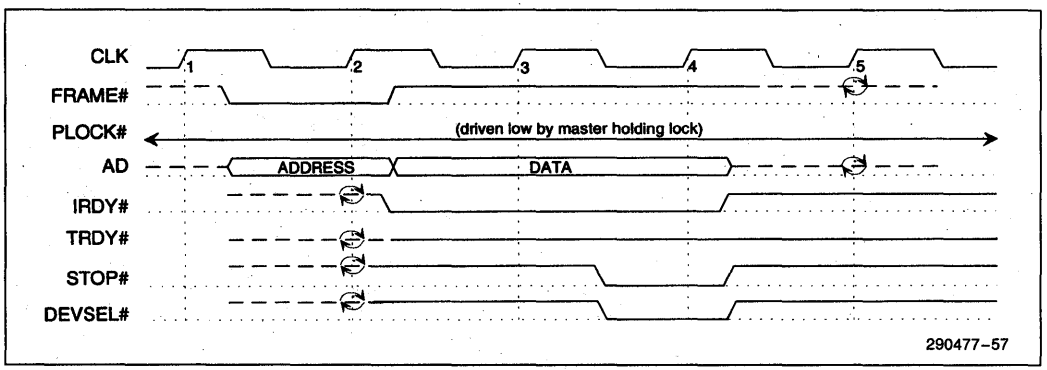


Figure 5-9. Access to Locked Target with PLOCK# Asserted During Address Phase

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 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 5-10). Accordingly, since the PCEB subtractively or negatively decodes all unclaimed PCI cycles (except configuration cycles), it provides a configuration option to reduce by 1 or 2 clocks 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.

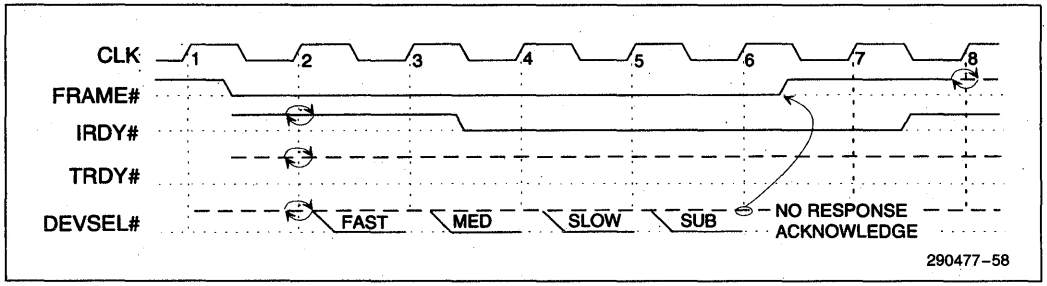


Figure 5-10. Device Selection (DEVSEL#)

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 5-11) 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.

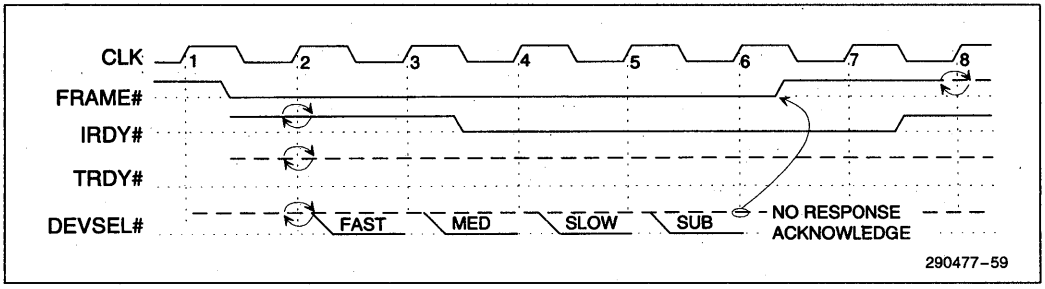


Figure 5-11. Master Initiated Termination (Master Abort)

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.

Figures 5-12 and 5-13 show four types of target-initiated terminations. In general, the PCEB initiates a disconnect for PCI cycles destined to EISA after the first data phase due to incremental latency requirements. The exception is the case of a PCI memory write cycle destined to the EISA subsystem when Posted Write Buffers (PWB) are enabled and there is a space available in the PWB. When the PWBs are completely full, the PCEB generates a disconnect during burst access. If the first data phase can not be posted, the PCEB generates a retry.

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.
- the cycle is a memory write and the Posted Write Buffer is full.

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.

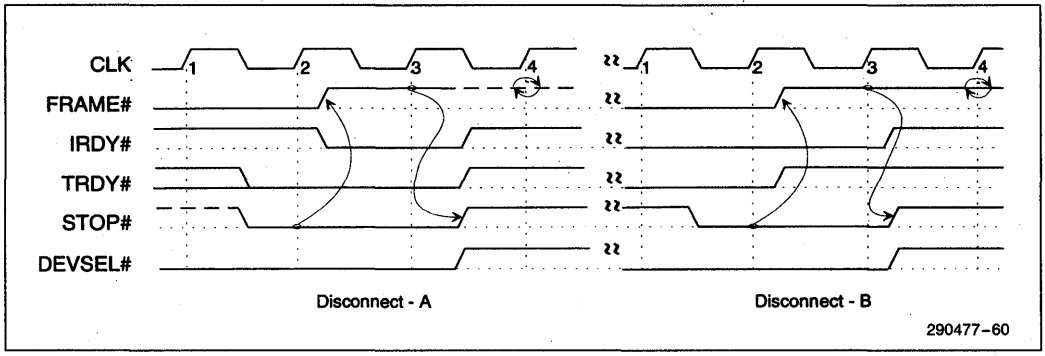


Figure 5-12. Target Initiated Termination

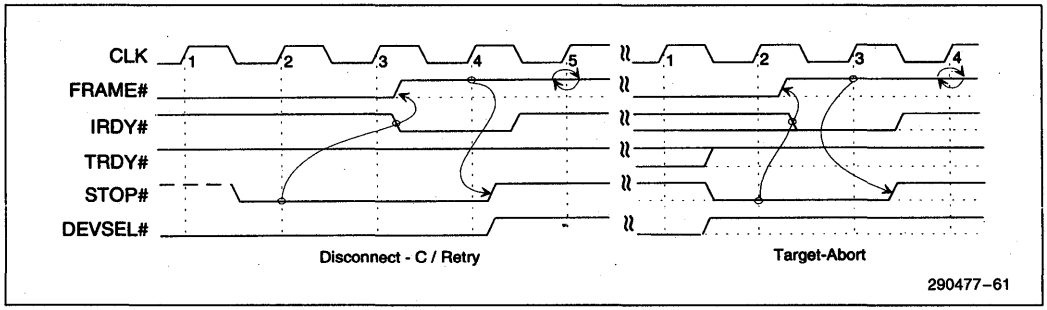


Figure 5-13. Target Initiated Termination

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), except in the case of memory write transactions, which can be posted. During posting, as soon as all PWBs are occupied the PCEB terminates the cycle using disconnect semantics. This is because the next data phases would exceed the 8 PCI clock incremental latency limit and the PCI would be kept in wait states for more than 2 BCLKs (= 8 PCICLKs), until one of the PWBs is emptied to its destination on the EISA Bus.

Retry

- For memory write cycles when all post write buffers are full. (See Section 6.0, Data Buffering.)
- 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 (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 its 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 and data is not posted in the Posted Write Buffers.

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 whenever 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.)

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). If the PCEB's internal PWBs are enabled, the EISA Bus is owned by the PCEB, and there is a space available in the PWBs, then the PCI memory cycles destined to EISA are posted until all PWBs are occupied. When the PWBs are occupied, the following cycle is disconnected (in the case of burst) or retried (in the case of single cycles). All other PCI-to-EISA cycles (memory read and I/O read or 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 chipset 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 5-14 summarize behavior of the PCEB during the address and data phase of a PCI Bus cycle.

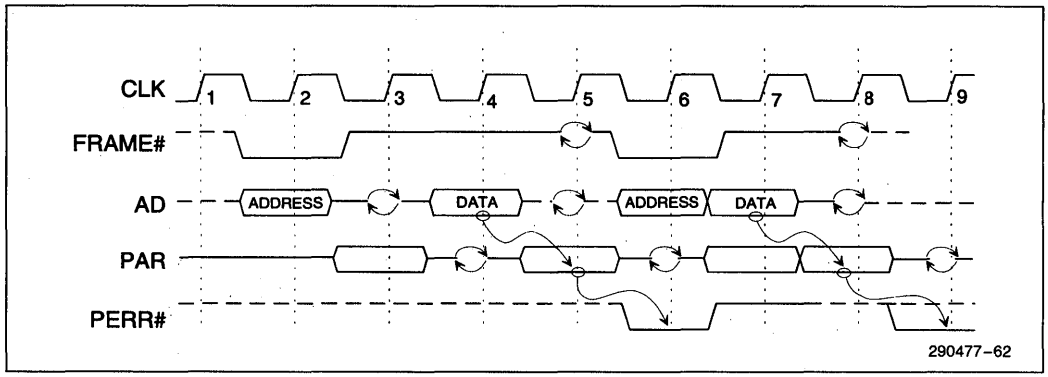


Figure 5-14. Parity Operation

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[31:0]#. 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 masters—The Host/PCI Bridge, PCEB, and four other masters. The PCEB's REQ#/GNT# signals are internal. If an external arbiter is used, the internal arbiter can be disabled. When disabled, the PCEB's internal REQ#, GNT#, and RESUME# signals become visible for an external arbiter (via the GNT0#/PCEBREQ#, REQ0#/PCEBGNT#, and GNT1#/RESUME# dual-function signal pins, respectively). During power-up, the internal arbiter is enabled if CPUREQ# is sampled high 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 PRIORITY SCHEME

The PCI arbitration priority scheme is programmable through the PCI Arbiter Priority Control Register. The arbiter consists of four banks that can be configured for the six masters to be arranged in a purely rotating priority scheme, one of twelve fixed priority schemes, or a hybrid combination (Figure 5-15).

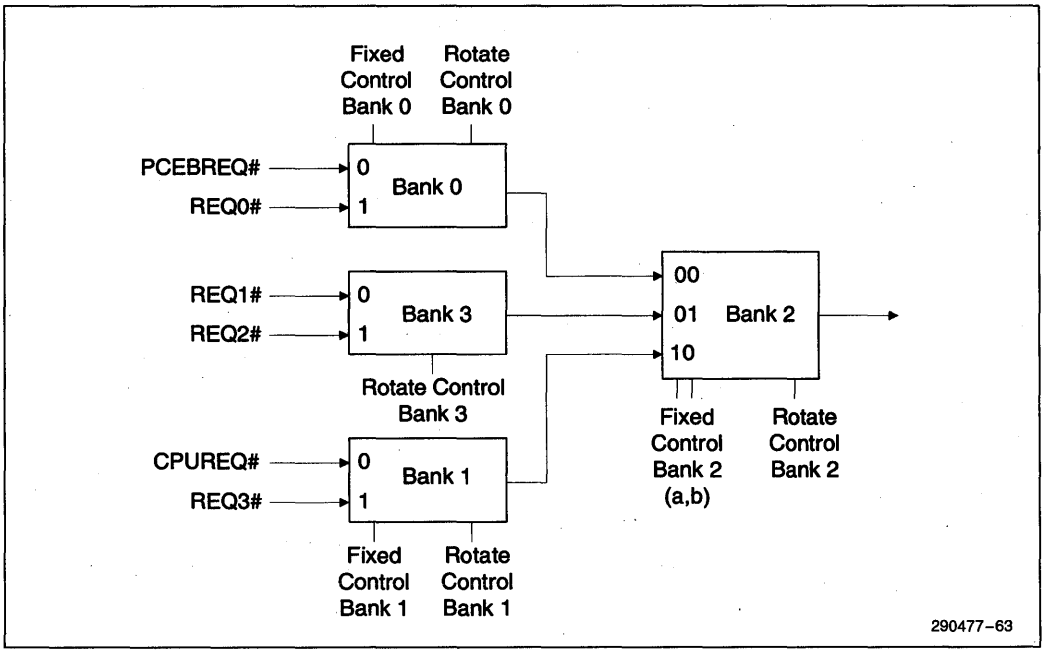


Figure 5-15. Arbiter Conceptual Block Diagram

Note that PCEBREQ#/PCEBGNT# are PCEB internal signals.

The PCI Arbiter Priority Control (ARBPRI) Register bits are shown below:

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 selecting fixed mode number 4 (Table 5-2) with the CPU the highest priority device guaranteeing access to BIOS.

Note that Bank 3 does not have programmability for fixed priority mode. If this bank is not used in rotating mode, the REQ1# is always a higher priority request than the REQ2#. This should be considered in system design, especially, if REQ1# and REQ2# are used to support PCI masters residing on the PCI daughter cards attached to PCI expansion connectors.

5.4.1.1 Fixed Priority Mode

The twelve selectable fixed priority schemes are listed in Table 5-2.

Table 5-2. Fixed Priority Mode Bank Control Bits

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

The fixed bank control bit(s) selects which requester is the highest priority device within that particular bank. For fixed priority mode, bits [7:4] of the ARBPRI Register must be 0's (rotate mode disabled).

The selectable fixed priority schemes provide 12 of the 64 possible fixed mode permutations possible for the six masters. Note that the priority between REQ1# and REQ2# is hardwired (no programmability of priority within Bank 3) where REQ1# is always higher priority in fixed mode.

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 POWER-UP CONFIGURATION

The PCEB's internal arbiter is enabled if CPUREQ# is sampled high on the low-to-high edge of PCIRST#. After PCIRST#, the internal arbiter, if enabled, is set to fixed priority mode number 4 with CPU parking turned off. Fixed mode number 4 guarantees that the CPU is capable of accessing BIOS to configure the system, regardless of the state of the other REQ#'s. Note that the Host/PCI Bridge should drive CPUREQ# high during the rising edge of PCIRST#. When the internal arbiter is enabled, the PCEB acts as the central resource and drives AD[31:0], C/BE[3:0]#, and PAR when no device is granted the PCI Bus and the bus is idle. The PCEB always drives these signals when it is granted the bus (PCEBGNT# and PCI Bus idle) and as appropriate when it is the master of a transaction. After reset, if the internal arbiter is enabled, CPUGNT#, GNT[3:0]#, and the internal PCEBGNT# are driven, based on the arbitration scheme and the asserted REQ#'s.

If an external arbiter is present in the system, the CPUREQ# signal should be tied low. When CPUREQ# is sampled low on the rising edge of the PCIRST#, the internal arbiter is disabled. When the internal arbiter is disabled, the PCEB does not drive AD[31:0], C/BE[3:0]#, and PAR as the central resource. The PCEB only drives these signals when it is granted the bus (PCEBGNT# and idle bus) and as appropriate when it is the master of a transaction. If the internal arbiter is disabled, GNT0# becomes PCEBREQ# (output signal), GNT1# becomes RESUME# (output signal), and REQ0# becomes PCEBGNT# (input signal). This exposes the normally embedded PCEB arbitration signals. Since these signals retain their input/output character there is no contention issue.

5.4.3 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 5-16 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 5-16 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.

5.4.3.1 REQ# and GNT# Rules

Figure 5-16 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 its request.

5.4.3.2 Back-to-Back Transactions

Figure 5-17 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.

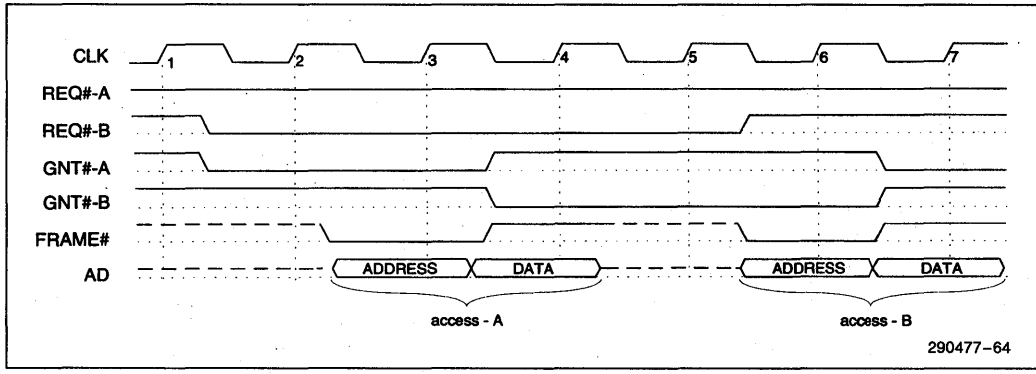


Figure 5-16. Basic Arbitration

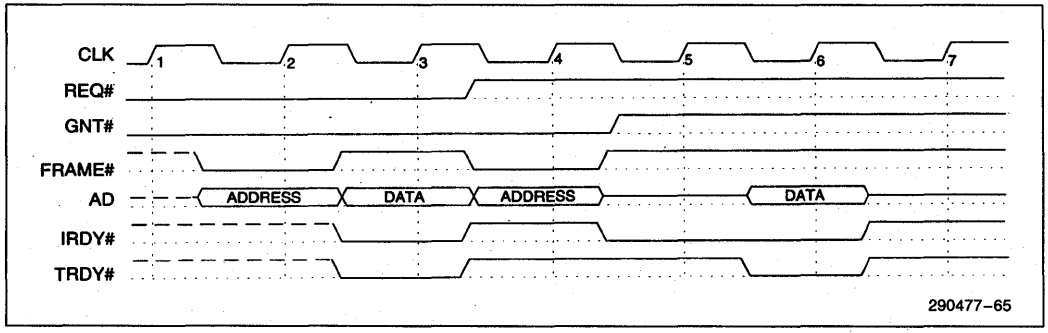


Figure 5-17. Arbitration for Back-to-Back Access

5.4.4 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:

1. PCEB target retries
2. 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.

5.4.4.1 Resume Function (RESUME#)

The PCEB forces a retry to a PCI master (resulting in the 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
3. The PCI-to-EISA Posted Write Buffer is full
4. The PCEB is locked as a resource and PLOCK# is asserted during the address phase.

The PCEB asserts RESUME# (internal or external) for a clock cycle when the PCEB has retried a PCI cycle for one of the above reasons and that condition passes. RESUME# is pulsed when:

1. The buffer management triggered by the retried cycle is complete.
2. The EISA Bus that caused retry because it was occupied becomes available.
3. The PWB that caused retry because it was full becomes available.
4. An exclusive access to the PCEB that caused retry of the non-exclusive access has completed (PCEB unlocked).

When RESUME# is asserted, it unmasks the REQ#s that are masked and flagged to be cleared by RESUME#. The RESUME# signal becomes visible as an output when the internal arbiter is disabled. This allows an external arbiter to optionally avoid retry thrashing associated with the PCEB (i.e., EISA Bus) as a target. When asserted, RESUME# is asserted for one PCICLK.

5.4.4.2 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). This is necessary to assist the Host/PCI bridge in determining when to re-enable its disabled posted write buffers.

5.4.5 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.

5.4.6 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 (including the PCEB Posted Write Buffer). 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 5-3 shows the relationship between MEMREQ# and FLSHREQ#.

Table 5-3. FLSHREQ# and MEMREQ#

FLSHREQ#	MEMREQ#	Meaning
1	1	Idle.
0	1	Flush buffers pointing towards the PCI Bus to avoid EISA deadlock.
1	0	Reserved.
0	0	GAT mode. Guarantees PCI Bus immediate access to main memory.

5.4.6.1. 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), flushes its PCI-to-EISA Posted Write Buffers 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. The PCEB flushes its internal EISA-to-PCI Posted Write Buffers, if they contain valid write data.
4. 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.

5. 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.
6. 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#.

5.4.6.2 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 2.1 μs 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), flushes its PCI-to-EISA posted write buffers 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).

3a. 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.

3b. 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 or external 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 (internal or external PCEBGNT#), 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 or external 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 or external 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 or external 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.

5.4.6.3 Interrupt Synchronization—Buffer Flushing

The ESC contains the system interrupt controller. Therefore, the PCEB/ESC chipset 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 has been 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.

5.4.6.4 System Buffer Flushing Protocol-State Machine

Figure 5-18 illustrates the functionality of the state machine contained within PCEB that implements the system buffer flushing protocol.

Definition of States

Normal: State where normal buffering is enabled.

WaitForGAT: Waiting for acknowledgment that the system has all buffers flushed and disabled and the EISA Bus master will have ownership of the entire system.

GAT: State where the system is ready for an EISA master to take over and have complete access to the system. The EISA Bus may be granted.

FlushEISA: Request for all devices in the system to flush all buffered writes that could end up going to the EISA Bus.

EISAbusy: State where an EISA Bus master owns the standard bus and no write data that might go to the EISA Bus may be posted.

Definition of Signals

GAT: GAT mode status bit.

REQ: EISA master request for PCI.

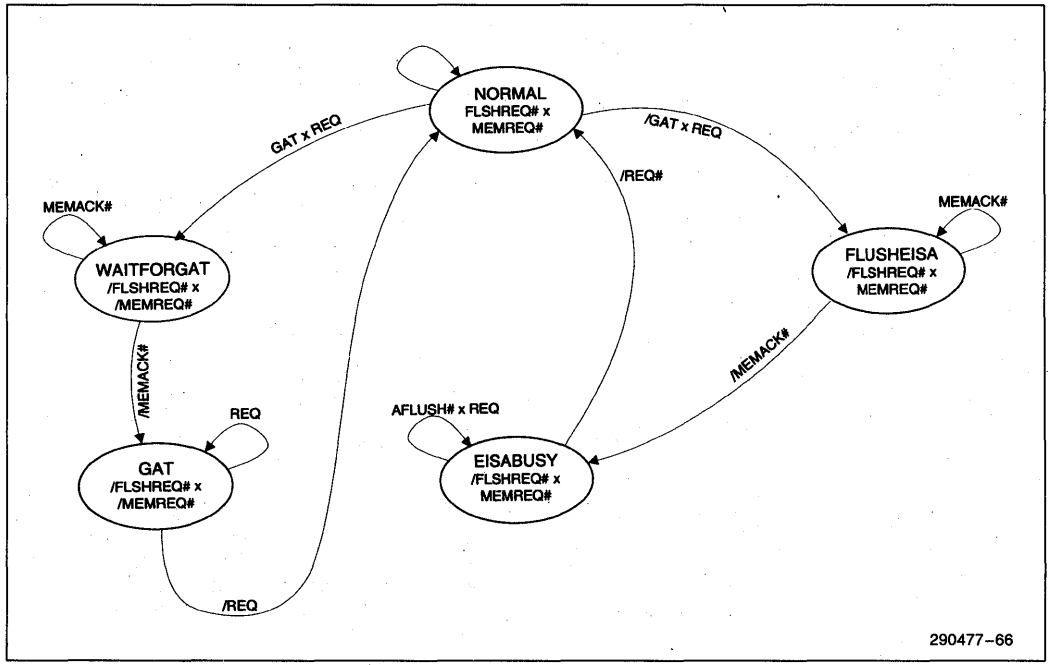


Figure 5-18. Buffer Flush Protocol-State Machine

5.4.7 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.

5.4.8 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.

5.4.8.1 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).

5.4.8.2 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 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 the EISA Bus. As soon as the ELT timer expires, the PCI cycle 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 EISA Latency Timer (ELT) is controlled by the ELTCR Register. The value written into ELTCR is system dependent. It is typically between 1 ms and 3 ms.

6.0 DATA BUFFERING

The PCEB contains data buffers (Figure 6-1) to isolate the PCI Bus from the EISA Bus and to provide concurrent EISA and PCI Bus operations. The Posted Write Buffers are used for PCI-to-EISA memory writes and the Line Buffers are used for EISA-to-PCI memory reads and writes. Control bits in the PCICON and EPMRA Registers permit the 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.

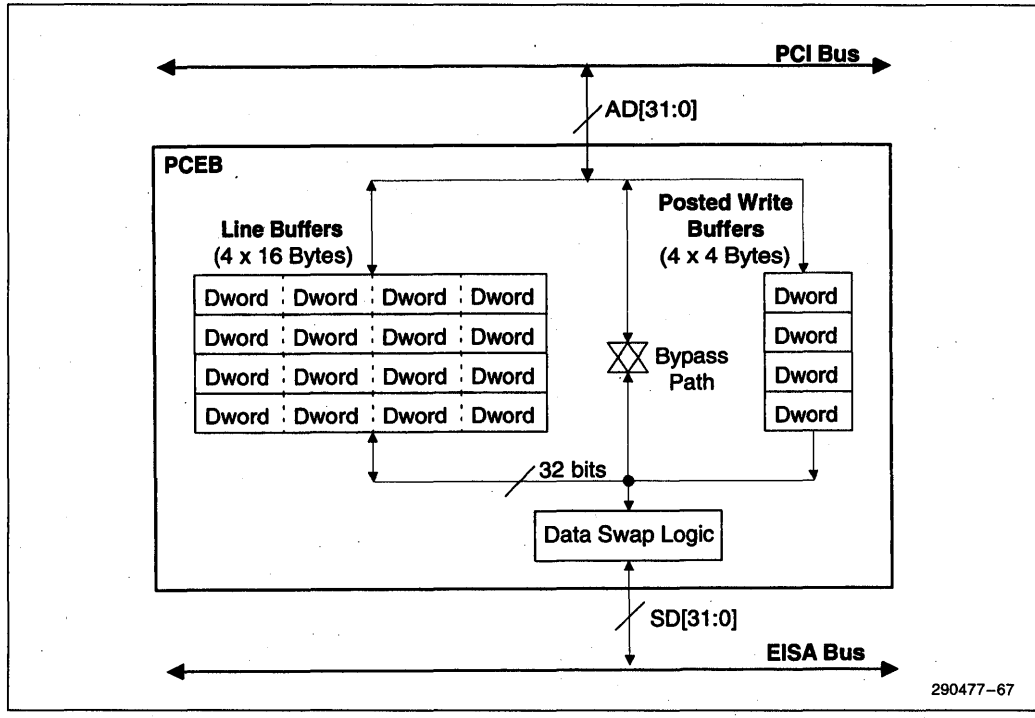


Figure 6-1. PCEB Data Buffers

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.

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.

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 scheduled 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.

6.2 Posted Write Buffers

The PCEB contains four Posted Write Buffers (PWB). The buffers are each one Dword wide (4 bytes). The PWBs are uni-directional and are used by PCI devices (including the CPU via the Host/PCI Bridge) to transfer data from a PCI master to EISA memory. If the PWBs are enabled, PCI-to-EISA memory writes are posted in the PWBs. If these buffers are disabled, PCI-to-EISA memory writes are not posted and use the bypass path. Posting means that the PCI Bus operation completes when the data is placed in the PWBs. The PCI operation does not have to wait for the transfer to complete to its destination on the EISA Bus, as is the case of a non-posted write. (Non-posted writes occur when the PWBs are disabled or a PCI I/O or configuration write occurs.) Posting permits the PCI Bus operation to complete in a minimum time.

To maintain strong ordering, data assembly within a Dword is not allowed. Thus, during each data phase of a PCI Bus write operation to the PWBs, each data transfer (byte, word, or Dword) is placed in separate PWBs. The corresponding address is stored for each PWB, until the data is written to the EISA/ISA device.

Posting is only permitted when the PCEB owns the EISA Bus. If the PCEB does not own the EISA Bus, the PCI master is retried and the PCEB requests the EISA Bus. The PCEB masks retried PCI masters until the EISA Bus is acquired, at which time the PCI masters are unmasked. If the PWBs are full and a new PCI-to-EISA memory write cycle occurs, the PCI master is retried and the PWBs flushed. After the first PWB is flushed, the PCI masters are unmasked and posting is permitted. If a PCI burst write fills the PWBs, the PCEB issues a disconnect to the PCI master and flushes the PWBs.

If an EISA/ISA master or DMA requests the EISA Bus during posting, the EISA latency timer is started. Posting and flushing of the PWBs can continue until the timer expires. If the timer expires, the PCEB allows the PCI master to complete the current bus cycle and retries further PCI requests. If the current PCI Bus cycle requires posting of more than four data transfers (or the PWBs become full), the PCEB issues a PCI target disconnect. When the PWBs are flushed, the PCEB grants ownership to the requesting EISA/ISA master or DMA.

NOTES:

1. If posting is disabled via the PCICON Register, a PCI master requesting a PCI-to-EISA transfer is retried until the PCEB owns the EISA Bus. Each PCI-to-EISA Bus transfer must complete all the way to the EISA destination before the next transfer can begin. The PCEB, on behalf of the EISA/ISA master or DMA, introduces wait states to the PCI master, if necessary.
2. If the ESC requests the EISA Bus to do a refresh cycle, the PCEB temporarily gives the bus to the ESC and does not flush the PWBs.

6.3 Buffer Management Summary

Table 6-1 shows Line Buffer and Posted Write Buffer actions for different cycles. Note that the first three columns together define the cycles that may trigger buffer activity.

Table 6-1. Buffer Management Summary

Master (Origin)	Cycle Type	Slave (Destination)	Line Buffer Data in Write State	Line Buffer Data in Read State	Posted Write Buffers
PCI	Memory Read	EISA	Flush	No Action	Flush
PCI	Memory Write	EISA	No Action	Invalidate	Flush if full post if not full
PCI	I/O Read	EISA	Flush	No Action	Flush
PCI	I/O Write	EISA	No Action	Invalidate	Flush
PCI	Interrupt Acknowledge	PCEB/ESC	Flush	No Action	Flush
PCI	Configuration Cycle	PCEB Registers	No Action	No Action	No Action
PCI	Memory Read/Write	PCI	No Action	No Action	No Action
PCI	I/O Read/Write	PCI	No Action	No Action	No Action
EISA	Bus Ownership Change	—	Flush	No Action	Flush
EISA	Memory Read/Write	EISA	No Action	No Action	No Action(2)
EISA	Memory Read/Write	PCI	(Note 1)	(Note 1)	No Action(2)
EISA	I/O Read/Write	EISA	No Action	No Action	No Action(2)
EISA	I/O Read/Write	PCI	Flush	Invalidate	No Action(2)

NOTES:
1. Change from write to read operation or from read to write causes the Line Buffers to be flush or invalidate, respectively.
2. Buffers are already flushed.
3. 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.

NOTES:

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 bits 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 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 read/write accesses, memory reads, and non-posted memory writes (Posted Write Buffers are disabled), the PCEB runs standard EISA Bus cycles. When the Posted Write Buffers are enabled, the PCEB posts PCI write data in the buffers. The data is transferred to the EISA Bus when the buffers are flushed. If just one buffer needs to be flushed, the PCEB runs a standard EISA cycle. For more than one buffer, the PCEB runs a burst cycle, if bursts are supported by the slave. If the slave does not support bursts, consecutive standard cycles are run.

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 7-1 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#.

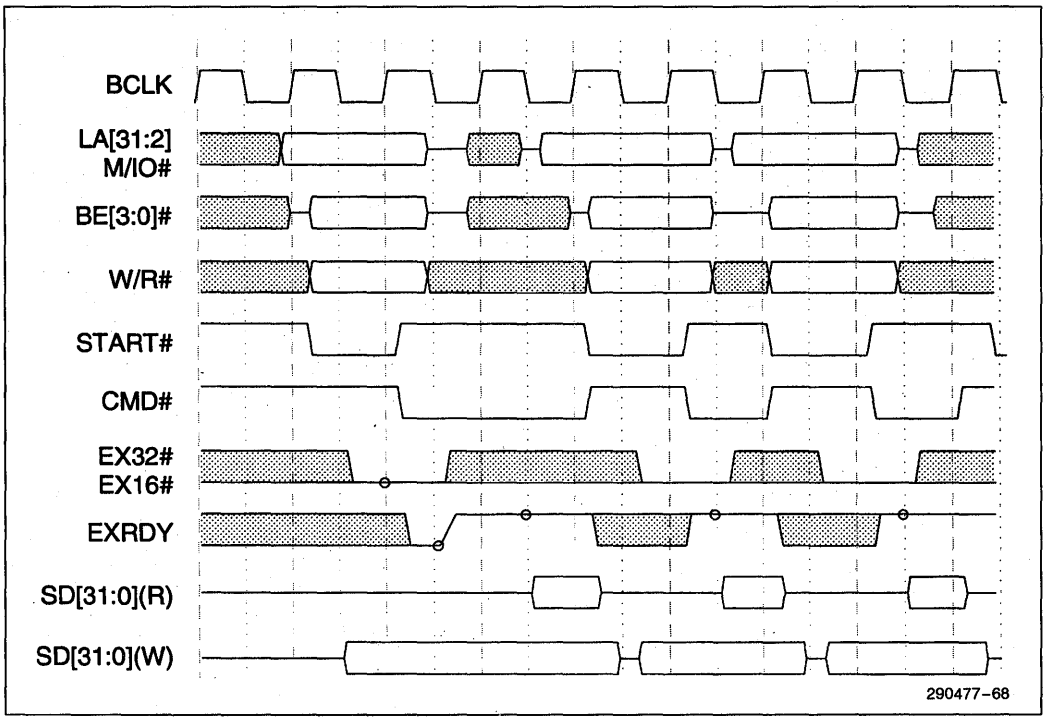


Figure 7-1. EISA Memory and I/O Read/Write Cycle (one extended and two standard cycles)

7.1.2 EISA MEMORY BURST CYCLES

The EISA burst cycle permits a continuous sequence of read or write cycles in zero wait-states (1 BCLK per transfer). As an EISA master, the PCEB can generate burst memory writes during PCI-to-EISA Posted Write Buffer flushing. However, the PCEB does not generate burst memory reads. Figure 7-2 shows a burst write on the EISA Bus. During the burst, five data transfers occur with a wait state added on the third data transfer.

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

The PCEB presents the write data on the rising edge of BCLK, a half cycle after presenting the address. An EISA memory slave that asserts SLBURST# must sample memory write data on a rising BCLK edge when CMD# is asserted (regardless of the state of MSBURST#). The PCEB terminates the burst cycles by negating MSBURST# and completing the last transfer.

Although a burst transfer normally performs zero wait state cycles, a slave can add wait states during a burst sequence by negating EXRDY before the falling edge of BCLK (with CMD# asserted). The PCEB, as a master, samples EXRDY on the falling edge of BCLK and extends the cycle until EXRDY is asserted. The PCEB can still change the next address even though EXRDY is negated.

Addresses asserted during a burst sequence to a memory must be within a 1024 byte memory page (address lines LA[31:10] can not change during the burst). To cross a page boundary, the burst sequence is terminated by the PCEB by negating the MSBURST# on the last cycle in the page. If the burst cycle crosses a page boundary, the PCEB terminates the cycle at the page boundary and the burst cycle is restarted on the new page.

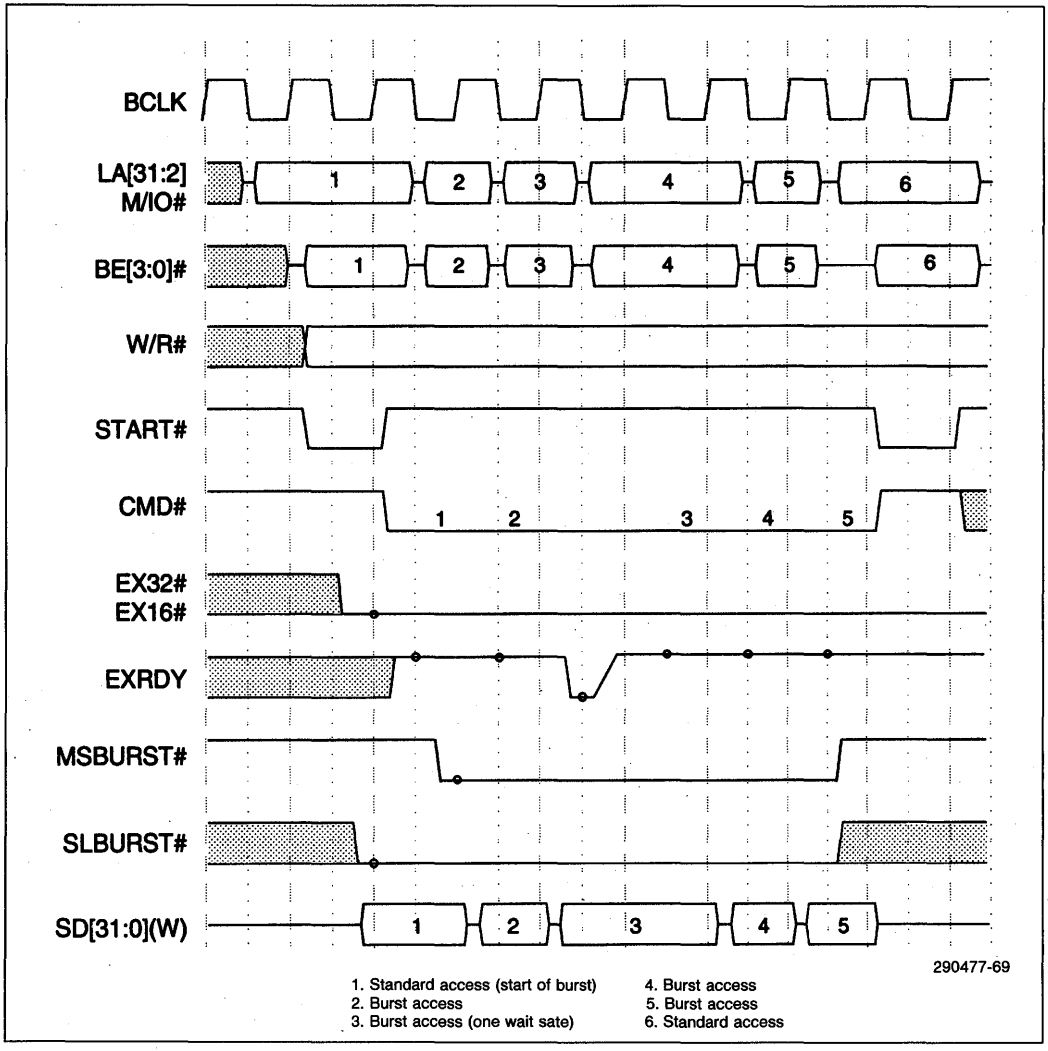


Figure 7-2. EISA Memory Write Burst Cycle

7.1.3 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 7-3 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[31:2].

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 buffers 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 buffers 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 logic. 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.

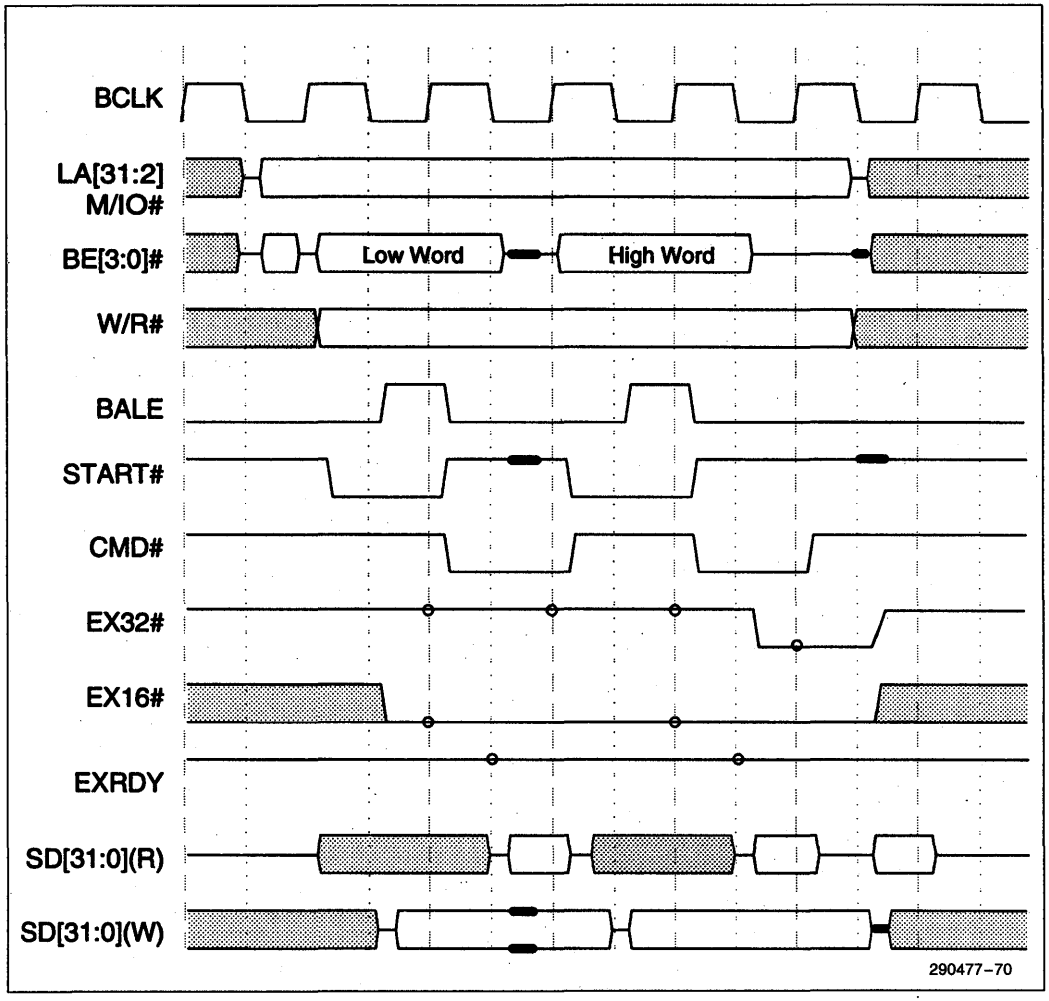


Figure 7-3. EISA Back-Off Cycle

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 7-4 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#.

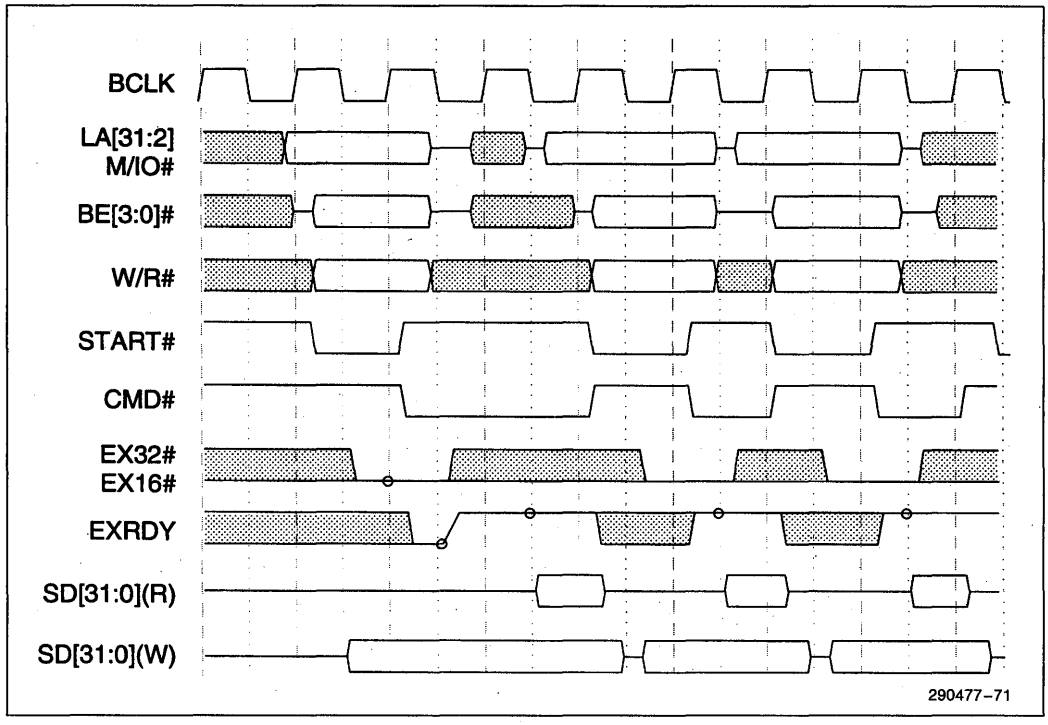


Figure 7-4. EISA Memory and I/O Read/Write Cycles (one extended and two standard cycles)

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 7-5 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.

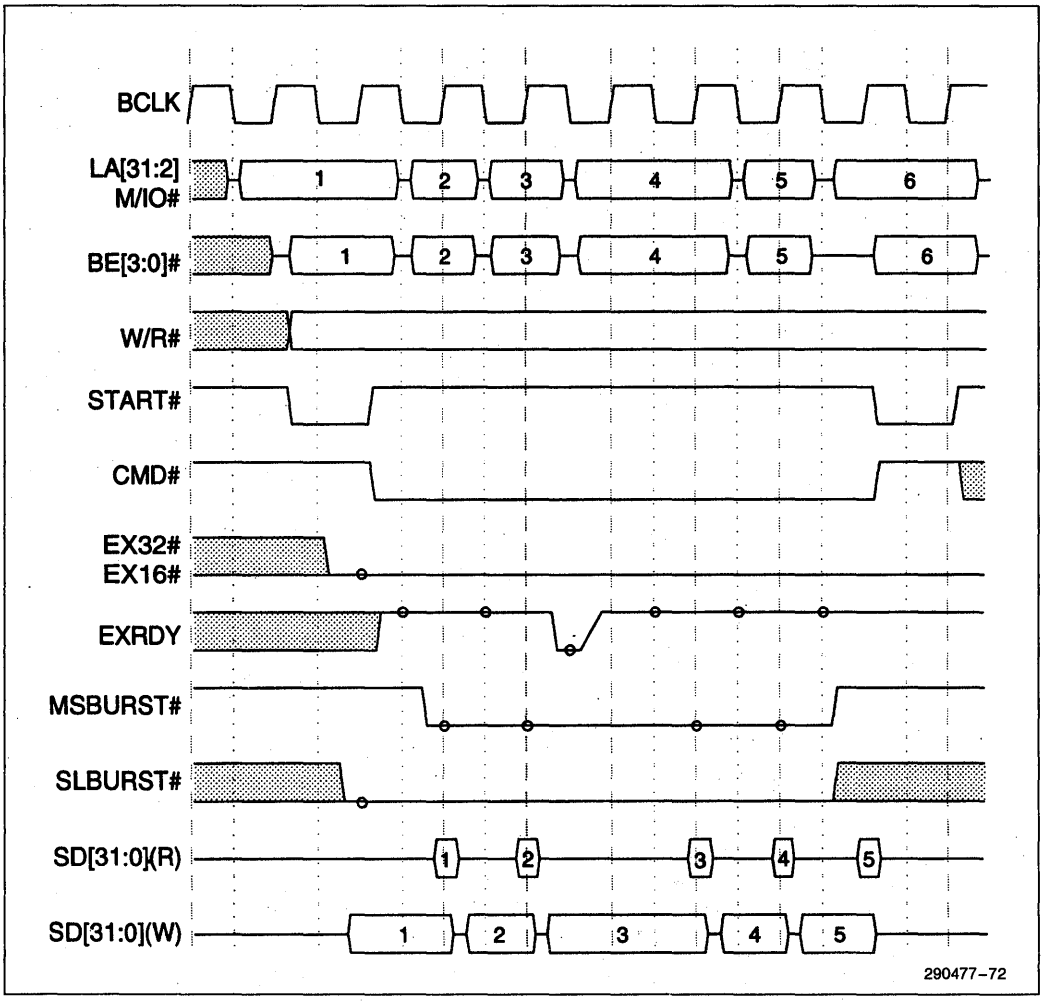


Figure 7-5. EISA Burst Cycle

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 8-1). 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.

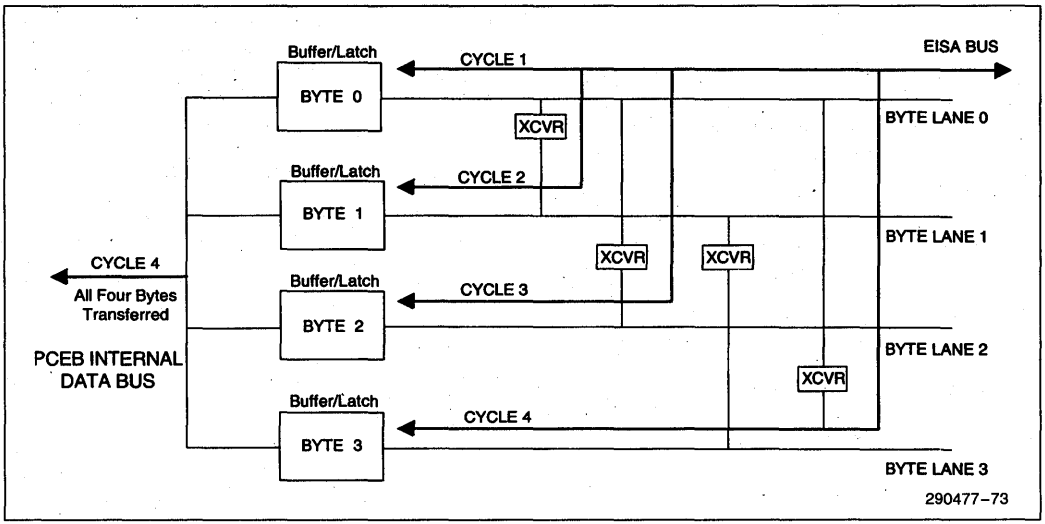


Figure 8-1. Assembly Function: PCI 32-bit Read from an 8-bit EISA or ISA Slave-BE[3:0]# = 0000

8.2 The Copy Operation (Up or Down)

The copy operation 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.

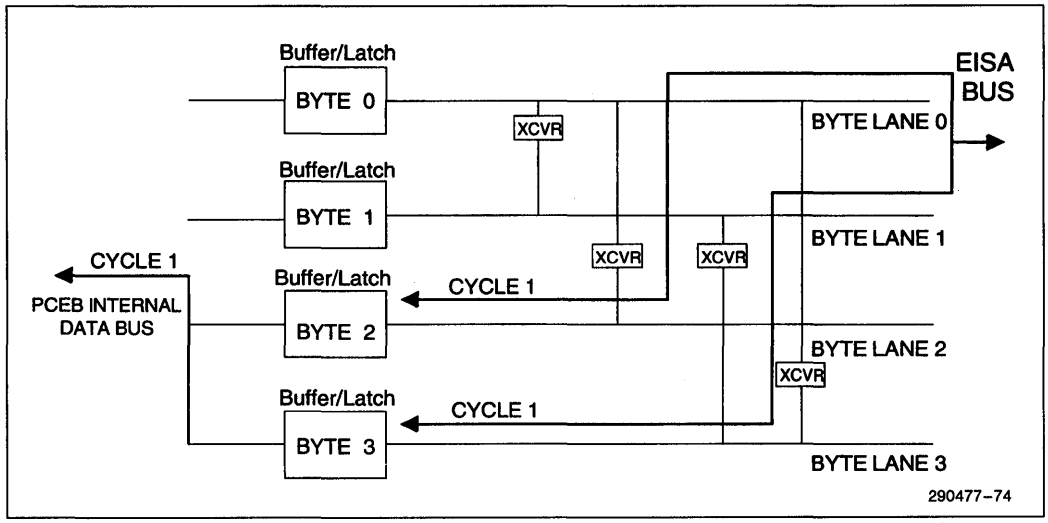


Figure 8-2. Copy Function: PCI 16-bit Read from a 16-bit EISA or ISA Slave-BE[3:0]# = 0011

8.3 The Re-drive Operation

The re-drive operation is used when both the master and the slave, other than PCEB, are on the EISA Bus and the master/slave size combination is mismatched. 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 mismatch, 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 swap logic 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. The data swap logic 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.

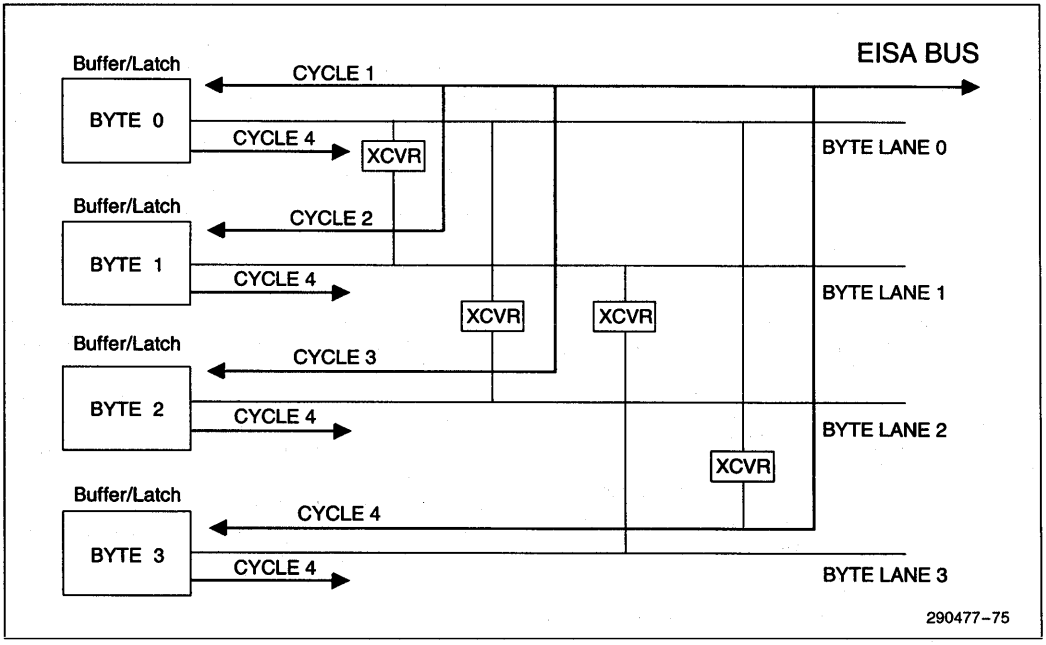


Figure 8-3. Re-Drive Function: 32-bit EISA Master Accessing an 8-bit EISA or ISA Slave—32-bit Read/BE[3:0]# = 0000

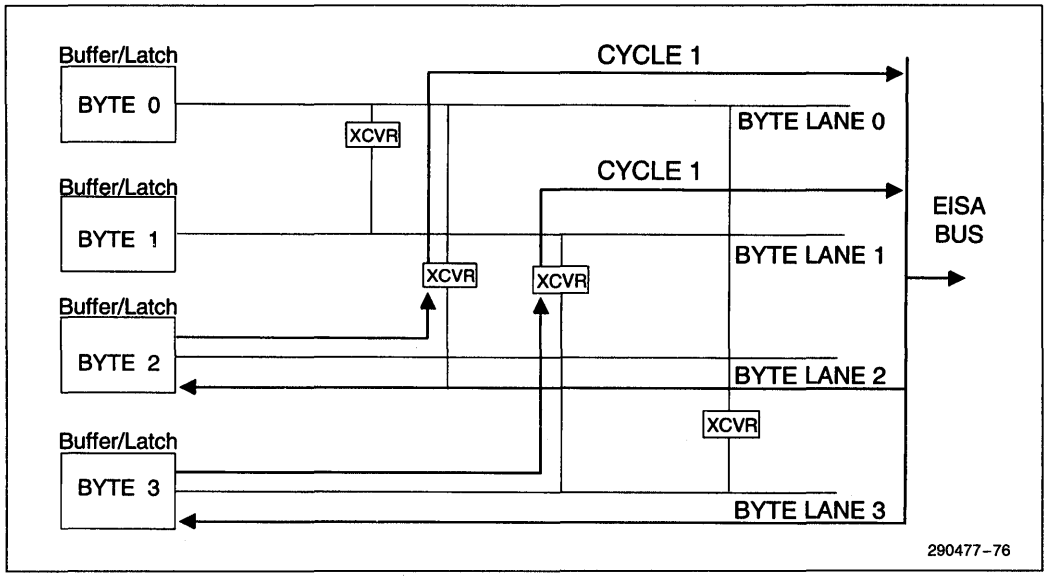


Figure 8-4. Copy with Re-Drive: 32-bit EISA Master Accessing a 16-bit EISA or ISA Slave—One Word Write/BE[3:0]# = 0011

9.0 BIOS TIMER

The PCEB provides a system BIOS Timer that decrements at each edge of its 1.03 MHz/1.04 MHz clock (derived from the 8.25 MHz/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 MHz 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 milliseconds. The accuracy of the counter is ± 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 10-1) 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.

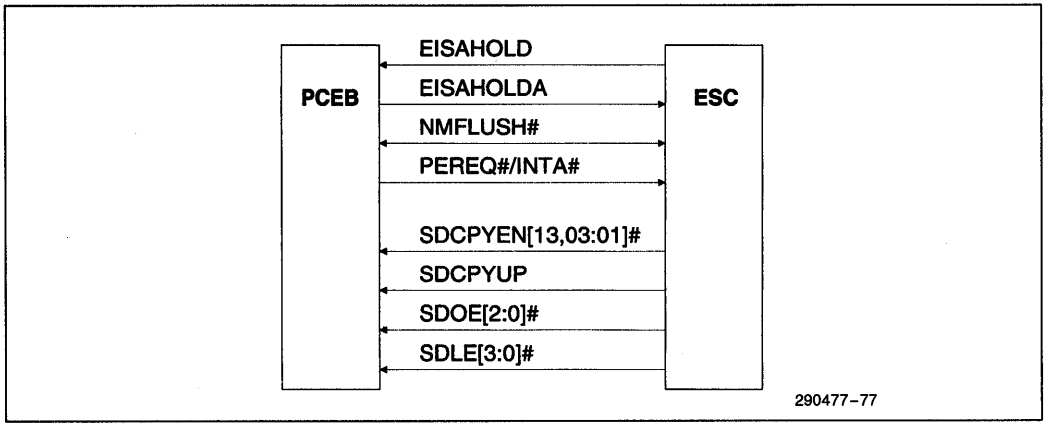


Figure 10-1. PCEB/ESC Interface Signals

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. 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.

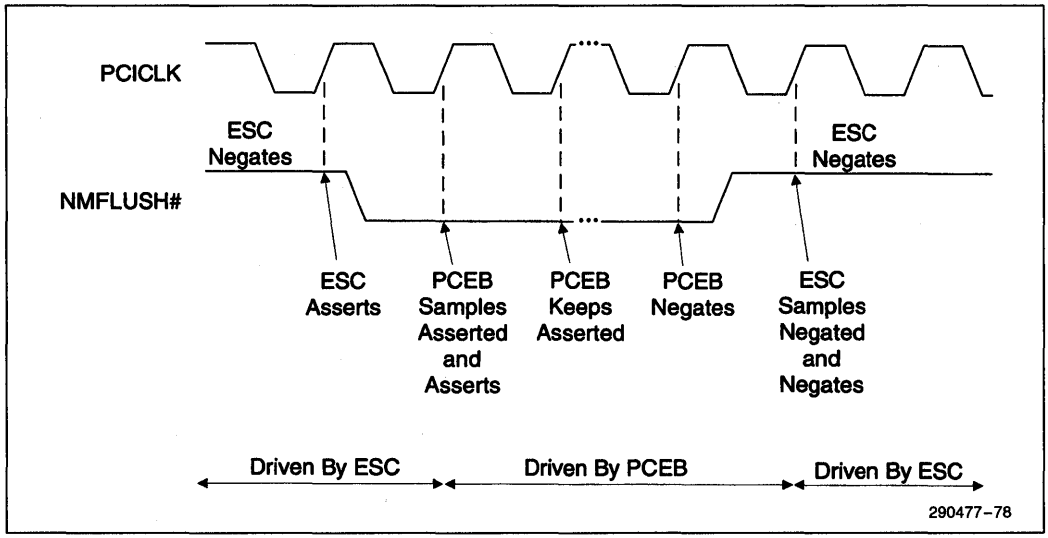


Figure 10-2. NMFLUSH# Protocol

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 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 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 10-1. 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 10-1. 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 [23:26])
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 10-2). 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 10-2. 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.

10.3 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 . . . . . 0.95 watts fully loaded
0.75 watts with four slots

**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.*

11.2 D.C. Characteristics

11.2.1 JUNCTION TEMPERATURE SPECIFICATIONS

The junction temperature for the PCEB is 95°C with a case temperature of 85°C. To guarantee device operation at 85°C case temperature the ambient temperature allowable is shown in Table 11-1.

Table 11-1. PCEB Maximum Allowable Ambient Temperature/Air Flow Rates

EISA Loading	Still	100 lfpm	200 lfpm	400 lfpm
4 Slots	53°C	58°C	61°C	66°C
8 Slots	44°C	49°C	54°C	59°C

11.2.2 EISA BUS D.C. SPECIFICATIONS

EISA Signals
BCLK(in), START#(t/s), CMD#(in), M/IO#(t/s), W/R#(t/s), EXRDY(o/d), EX32#(o/d), EX16#(in), MSBURST#(t/s), SLBURST#(t/s), LOCK#(t/s), BE[3:0]#(t/s), LA[31:2](t/s), SD[31:0](t/s), REFRESH#(in)

ISA Signals
IO16(out)

Interchip Signals
SDCPYEN[13,03:01]#(in), SDCPYUP(in), SDOE[2:0]#(in), SDLE[3:0]#(in), EISAHOLD(in), EISAHOLDA(out), PEREQ#/INTA#(out), INTCHIP0(t/s), NMFLUSH#(t/s)

Table 11-2. EISA/ISA Bus D.C. Specifications (V_{DD} = 5V ±5%, T_{case} = 0°C to +85°C)

Symbol	Parameter	Min	Max	Test Conditions	Notes
V_{IL1}	Input Low Voltage		0.8V		
V_{IH1}	Input High Voltage	2.0V			
V_{IL2}	Input Low Voltage		0.8V		1
V_{IH2}	Input High Voltage	V_{CC} − 0.8V			1
V_{OL1}	Output Low Voltage		0.45V	I_{OL} = 24 mA	2
V_{OH1}	Output High Voltage	2.4V		I_{OH} = −5.0 mA	2
V_{OL2}	Output Low Voltage		0.45V	I_{OL} = 1 mA	3
V_{OH2}	Output High Voltage	V_{CC} − 0.45V		I_{OH} = −1 mA	3
I_{LI}	Input Leakage Current		±15 μA	$0V < V_{IN} < V_{CC}$	
I_{LO}	Output Leakage Current		±15 μA	$0.45V < V_{IN} < V_{CC}$	
C_{IN}	Capacitance Input		8 pF		
C_{OUT}	Capacitance Output		15 pF	at 1 MHz	
I_{CC}	V_{CC} Supply Current		TBD	TBD	

NOTES:
1. All EISA Bus signals use V_{IL1}, V_{IH1} for input levels except for the Interchip signals: SDCPYEN#, SDCPYUP, SDOE#, SDLE#, EISAHOLD, EISAHLDA, PEREQ#/INTA#, INTCHIP0, NMFLUSH#.
2. BE[3:0]#, EX16#, EX32#, EXRDY, LA[31:2], LOCK#, M/IO#, MSBURST#, SD[31:0], SLBURST#, START#, W/R#.
3. SDCPYEN#, SDCPYUP, SDOE#, SDLE#, EISAHOLD, EISAHLDA, PEREQ#/INTA#, INTCHIP0, NMFLUSH#.

11.2.3 PCI BUS D.C. SPECIFICATIONS

PCI System Signals
PCICLK(in), PCIRST(in)

PCI Shared Signals
AD[31:0](t/s), C/BE[3:0]#(t/s), FRAME#(s/t/s), TRDY(s/t/s), IRDY(s/t/s), TOP#(s/t/s), PCILOCK#(s/t/s), IDSEL(in), DEVSEL#(s/t/s), PAR(t/s), PERR#(s/t/s)

PCI Sideband Signals
CPUREQ#(in), REQ1#(in), REQ0#/PCEBGNT#(in), REQ[2:3]#(in), CPUGNT#(out), GNT0#/PCEBREQ# (out), GNT1#/RESUME#(out), GNT[2:3]#(out), MEMREQ#(out), FLSHREQ#(out), MEMACK#(in), MEMCS#(out), PIODEC#(in)

Table 11-3. PCI Bus D.C. Specifications (V_{DD} = 5V ±5%, T_{case} = 0°C to +85°C)

Symbol	Parameter	Min	Max	Test Conditions	Notes
V_{IL}	Input Low Voltage		0.8V		
V_{IH}	Input High Voltage	2.0V	5.5 V		
V_{OL}	Output Low Voltage		0.55V	I_{OL1} = 3 mA, $I_{OL}2$ = 6 mA	1
V_{OH}	Output High Voltage	2.4 V		I_{OH} = −2.0 mA	
I_{IL}	Low-level Input Current		−70 μA	V_{IN} = 0.5V	
I_{IH}	High-level Input Current		70 μA	V_{IN} = 2.7V	
$C_{I/O}$	Input/Output Capacitance		10 pF	at 1 MHz	
C_{CLK}	PCICLK Signal Input Capacitance		17 pF	at 1 MHz	
I_{CC}	V_{CC} Supply Current		TBD	TBD	

NOTE:
1. I_{OL2} applies to those signals requiring external pull-ups: FRAME#, TRD#, IRDY#, STOP#, LOCK#, DEVSEL#.

11.3 A.C. Characteristics

The Symbol column shows the timing variable used in the A.C. timing waveforms. The parameter column contains the description of the timing and its reference signal. If the timing is for a particular bus cycle, the cycles are listed in parenthesis. Burst cycles include standard timings for their requirements. The Min column lists either the minimum delay time, set-up time, or hold time requirement in nano-seconds, unless stated otherwise. The Max column lists the maximum delay time, also in nano-seconds. The Figure column shows what A.C. timing waveforms the parameter can be found. The Note column may contain a number to refer to a specific note found at the end of the table.

The A.C. specifications are based upon the specified capacitive loading values in the table below. The minimum capacitive loading value is 50 pF for all signals.

Table 11-4. Capacitive Loading Table

Signals	Loading
BE[3:0]#, EX16#, EX32#, EXRDY, LA[31:2], LOCK#, M/IO#, MSBURST#, SD[31:0], SLBURST#, START#, W/R#	240 pF
SDCPYEN#, SDCPYUP, SDOE#, SDLE#, EISAHOLD, EISAHLDA, PEREQ#/INTA#, INTCHIP0, NMFLUSH#, PCICLK, PCIRST, AD[31:0], C/BE[3:0]#, FRAME#, TRDY, IRDY, STOP#, PCILOCK#, IDSEL, DEVSEL#, PAR, PERR#, CPUREQ#, REQ1#, REQ0#/PCEBGNT#, REQ[2:3]#, CPUGNT#, GNT0#/PCEBREQ#, GNT1#/RESUME#, GNT[2:3]#, MEMREQ#, FLSHREQ#, MEMACK#, MEMCS#, PIODEC#	50 pF

11.3.1 CLOCK SIGNALS A.C. SPECIFICATIONS

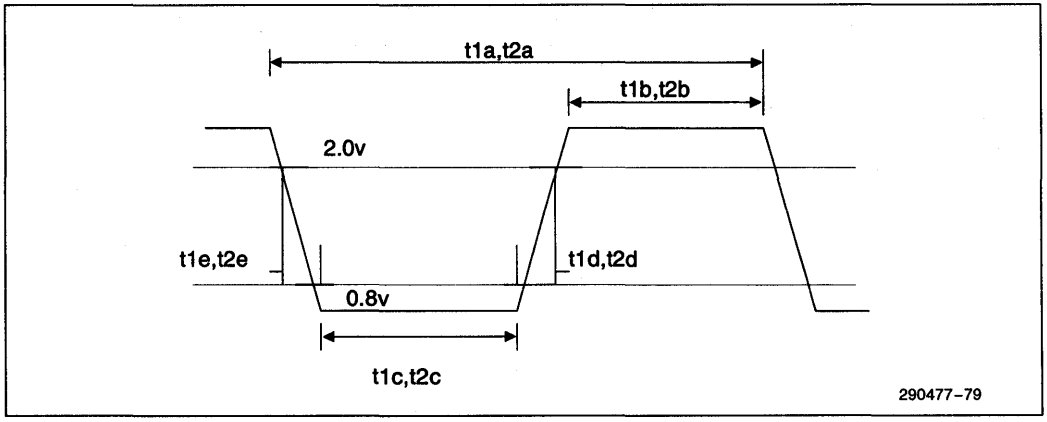


Figure 11-1. PCICLK, BCLK Timing

Table 11-5. Clock Signals A.C. Specifications (V_{DD} = 5V ±5%, T_{case} = 0° to +85°C)

Symbol	Parameter	Min	Max	Figures	Notes
PCICLK					
t1a	Cycle Time	30		11-1	
t1b	High Time (at 2.0V)	40% * T_{cyc}		11-1	
t1c	Low Time (at 0.8V)	40% * T_{cyc}		11-1	
t1d	Rise Time (0.8V to 2.0V)	3		11-1	
t1e	Fall Time (2.0V to 0.8V)	3		10-0	
BCLK					
t2a	Clock Period (0.8V to 0.8V)	120		10-1	
t2b	Low Time (at 0.8V)	55		10-1	
t2c	High Time (at 2.0V)	56		10-1	
t2d	Rise Time (0.8V to 2.0V)		7	10-1	
t2e	Fall Time (2.0V to 0.8V)		6	10-1	

11.3.2 PCI A.C. SPECIFICATIONS

Table 11-6. PCI A.C. Specifications (V_{DD} = 5V ±5%, T_{case} = 0° to +85°C)

Symbol	Parameter	Min	Max	Figures	Notes
PCIRST#					
t3a	Pulsewidth	1 ms		11-11	
t3b	PCICLK Active Setup to PCIRST# Deasserted	100 μs		11-11	
AD[31:0], C/BE[3:0]#, FRAME#, IRDY#,PAR, PERR#, TRDY#, DEVSEL#, STOP#, PCILOCK#, IDSEL					
t4a	Delay from PCICLK Rising	2	11	11-10	
t4b	Setup to PCICLK Rising	7		11-10	
t4c	Hold from PCICLK Rising	0		11-10	
REQ[3:0]#/GNT[3:0]#, PCEBREQ#, RESUME#, CPUREQ#, CPUGNT#					
t5a	GNT Delay from PCICLK Rising	2	12	11-9	
t5b	REQ Setup to PCICLK Rising	12		11-9	
t5c	REQ Hold from PCICLK Rising	0		11-9	
MEMREQ#, FLSHREQ#, MEMACK#					
t36a	MEMREQ#, FLSHREQ# Delay from PCICLK Rising	2	12	11-12	
t36b	MEMACK# Setup to PCICLK Rising	12		11-12	
t36c	MEMACK# Hold from PCICLK Rising	12		11-12	

11.3.3 EISA INTERFACE A.C. SPECIFICATIONS

Table 11-7. EISA Master Interface A.C. Specifications (V_{DD} = 5V ±5%, T_{case} = 0°C to +85°C)

Symbol	Parameter	Min	Max	Figures	Notes
EISA MASTER					
LA[31:2]					
t6a	Delay from BCLK Falling	2	30	11-2	
t6b	Float Delay from BCLK Falling (End of Master Mode)	0	40	11-13	
BE[3:0]#					
t7a	Delay from BCLK Rising	0	25	11-2	
t7b	Float Delay from BCLK Falling (End of Master Mode)	0	40	11-13	
t7c	Float Delay from BCLK Falling (Backoff Cycle)	0	40	11-6	
M/IO					
t8a	Delay from BCLK Falling	2	30	11-2	
t8b	Float Delay from BCLK Falling (End of Master Mode)	0	40	11-13	
W/R					
t9a	Delay from BCLK Rising	0	25	11-2	
t9b	Float Delay from BCLK Falling (End of Master Mode)	0	40	11-13	
START#					
t10a	Delay from BCLK Rising	2	25	11-2	
t10b	Float Delay from BCLK Falling (Backoff Cycle)	0	40	11-6	
t10c	Float Delay from BCLK Falling (End of Master Mode)	0	40	11-13	
EX32#, EX16#					
t11a	Setup to BCLK Rising	25		11-2	
t11b	Setup to BCLK Rising (End of Backoff Cycle)	15		11-6	
t11c	Hold from BCLK Rising	55		11-2	
t11d	Hold from BCLK Rising (End of Backoff Cycle)	50		11-6	
EXRDY					
t12a	Setup to BCLK Falling	15		11-2	
t12b	Hold from BCLK Falling	5		11-2	
LOCK					
t13a	Delay from BCLK Rising	2	60	11-2	
t13b	Float Delay from BCLK Falling (End of Master Mode)	0	40	11-13	
SD[31:0]					
t14a	Delay from BCLK Falling (Write)	5	40	11-2	
t14b	Setup Time to BCLK Rising (Read)	12		11-2, 4	
t14c	Hold Time from BCLK Rising (Read)	4		11-2, 4	
t14d	Float Delay from BCLK Falling (Backoff Write Cycles)	0	50	11-6	
t14e	Float Delay from BCLK Falling (End of Master Mode)			11-13	

Table 11-7. EISA Master Interface A.C. Specifications
(V_{DD} = 5V ±5%, T_{case} = 0°C to +85°C) (Continued)

Symbol	Parameter	Min	Max	Figures	Notes
EISA MASTER BURST CYCLES					
LA[31:2]					
t15	Delay from BCLK Falling	2	30	11-4	
BE[3:0]#					
t16	Delay from BCLK Falling	2	30	11-4	
MSBURST#					
t17a	Delay from BCLK Falling	2	35	11-4	
t17b	Float Delay from BCLK Falling (End of Master Mode)	0	40	11-13	
SLBURST#					
t18a	Setup to BCLK Rising	15		11-4	
t18b	Hold from BCLK Rising	55		11-4	
SD[31:0]					
t19a	Delay from BCLK Rising (Write)	5	40	11-4	
t19b	Setup to BCLK Rising (Read)	15		11-4	
t19c	Hold from BCLK Rising (Read)	5		11-4	

Table 11-8. EISA Slave Interface A.C. Specifications (V_{DD} = 5V ±5%, T_{case} = 0° to +85°C)

Symbol	Parameter	Min	Max	Figures	Notes
EISA SLAVE					
LA[31:2]					
t20a	Setup to BCLK Rising	10		11-3	
t20b	Hold from BCLK Rising	20		11-3	
BE[3:0]#					
t21a	Setup to BCLK Rising at CMD# Time	80		11-3	
t21b	Hold from BCLK Rising	20		11-3	
M/IO					
t22a	Setup to BCLK Rising	10		11-3	
t22b	Hold from BCLK Rising	20		11-3	
W/R					
t23a	Setup to BCLK Rising at CMD# Time	80		11-3	
t23b	Hold from BCLK Rising	20		11-3	
START#					
t24a	Setup to BCLK Rising	88		11-3	
t24b	Asserted Pulsewidth	115		11-3	
EX32#, EX16#					
t25a	Delay from LA Valid	2	54	11-3	
t25b	Delay from PIODEC# Asserted		15	11-14	1

Table 11-8. EISA Slave Interface A.C. Specifications (V_{DD} = 5V ±5%, T_{case} = 0° to +85°C) (Continued)

Symbol	Parameter	Min	Max	Figures	Notes
EXRDY					
t26a	Negate Delay from BCLK Rising		35	11-3	
t26b	Float Delay from BCLK Rising	3	34	11-3	
LOCK					
t27a	Setup to BCLK Rising	54		11-3	
t27b	Hold from BCLK Rising	2		11-3	
SD[31:0]					
t28a	Delay from BCLK Rising (Read)	0	50	11-3	
t28b	Setup to BCLK Rising (Write)	110		11-3, 5	
t28c	Hold from BCLK Rising (Write)	25		11-3, 5	
EISA SLAVE BURST CYCLES					
LA[31:2]					
t29a	Setup to BCLK Rising	5		11-5	
t29b	Hold from BCLK Rising	2		11-5	
BE[3:0]#					
t30a	Setup to BCLK Rising	5		11-5	
t30b	Hold from BCLK Rising	2		11-5	
MSBURST#					
t31a	Setup to BCLK Rising	14		11-5	
t31b	Hold from BCLK Rising	45		11-5	
SLBURST#					
t32	Delay from LA Valid	2	55	11-5	
SD[31:0]					
t33a	Delay from BCLK Rising (Read)	35	80	11-5	
t33b	Setup to BCLK Rising (Write)	55		11-5	
t33c	Hold from BCLK Rising (Write)	5		11-5	

11.3.4 ISA INTERFACE A.C. SPECIFICATIONS

Table 11-9. ISA Interface A.C. Specifications (V_{DD} = 5V ±5%, T_{case} = 0° to +85°C)

Symbol	Parameter	Min	Max	Figures	Notes
REFRESH#					
t34	Setup to START# Asserted	5		11-15	
IO16#					
t35a	Assert Delay from BCLK Rising		70	11-15	
t35b	Assert Delay from PIODEC Asserted		15	11-15	2
t35c	Setup to CMD# Asserted	10		11-18	
t35d	Hold from CMD# Negated	0		11-18	

11.3.5 DATA SWAP BUFFERS A.C. SPECIFICATIONS

Table 11-10. Data Swap A.C. Specifications (V_{DD} = 5V ±5%, T_{case} = 0° to +85°C)

Symbol	Parameter	Min	Max	Figures	Notes
SDCPYEN[13,03:01]#, SDCPYUP#					
t37a	Asserted to Valid data Delay	2	21	11-16	
t37b	SDCPYUP# Setup to SDCPYEN# Asserted	0		11-16	
SD[x] to SD[x]					
t37c	Copy Buffer Propogation Delay		16.5	11-16	
SDLE[3:0]#					
t38a	Data Setup Time to SDLE# Rising	3.5		11-16	
t38b	Data Hold Time from SDLE# Rising	3.5		11-16	
SDOE[2:0]#					
t39	Asserted to Valid Data Delay		17.5	11-16	

11.3.6 ARBITRATION AND INTERRUPT ACKNOWLEDGE A.C. SPECIFICATIONS

Table 11-11. Arbitration and Interrupt Acknowledge A.C. Specifications
(V_{DD} = 5V ±5%, T_{case} = 0° to +85°C)

Symbol	Parameter	Min	Max	Figures	Notes
EISAHOLD					
t40a	Setup to PCICLK Rising	11		11-7	
t40b	Hold from PCICLK Rising	0		11-7	
EISAHOLDA					
t41	Delay from PCICLK Rising	2	12	11-7	
PEREQ#/INTA#					
t42	Delay from PCICLK Rising	2	12	11-7	

11.3.7 BUFFER COHERENCY A.C. SPECIFICATIONS

Table 11-12. Buffer Coherency A.C. Specifications (V_{DD} = 5V ±5%, T_{case} = 0° to +85°C)

Symbol	Parameter	Min	Max	Figures	Notes
INTCHIP0, NMFLUSH#					
t43a	Delay from PCICLK Rising (Acknowledge from PCEB)	2	13	11-8	
t43b	Setup to PCICLK Rising (Request from ESC)	12		11-8	
t43c	Hold from PCICLK Rising	0		11-8	

11.3.8 ADDRESS DECODER A.C. SPECIFICATIONS

Table 11-13. Address Decoder A.C. Specifications (V_{DD} = 5V ±5%, T_{case} = 0° to +85°C)

Symbol	Parameter	Min	Max	Figures	Notes
MEMCS#					
t44	Delay from PCICLK Rising (Acknowledge from PECB	2	17	11-17	

NOTES:
1. EX32# must still meet the propogation requirement from LA valid.
2. PIODEC must be provided in time to meet the EISA specification.

11.3.9 A.C. TIMING WAVEFORMS

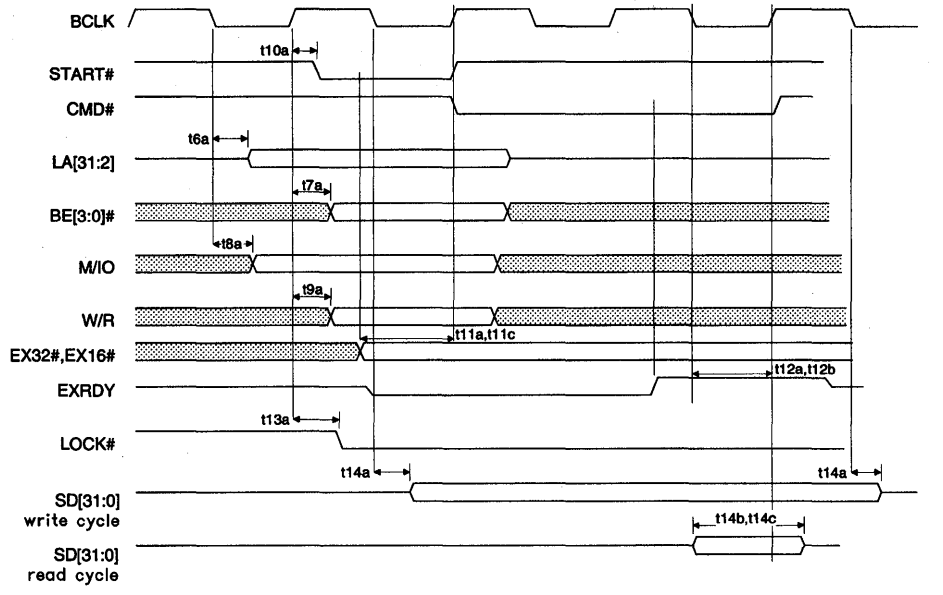


Figure 11-2. EISA Master Cycle

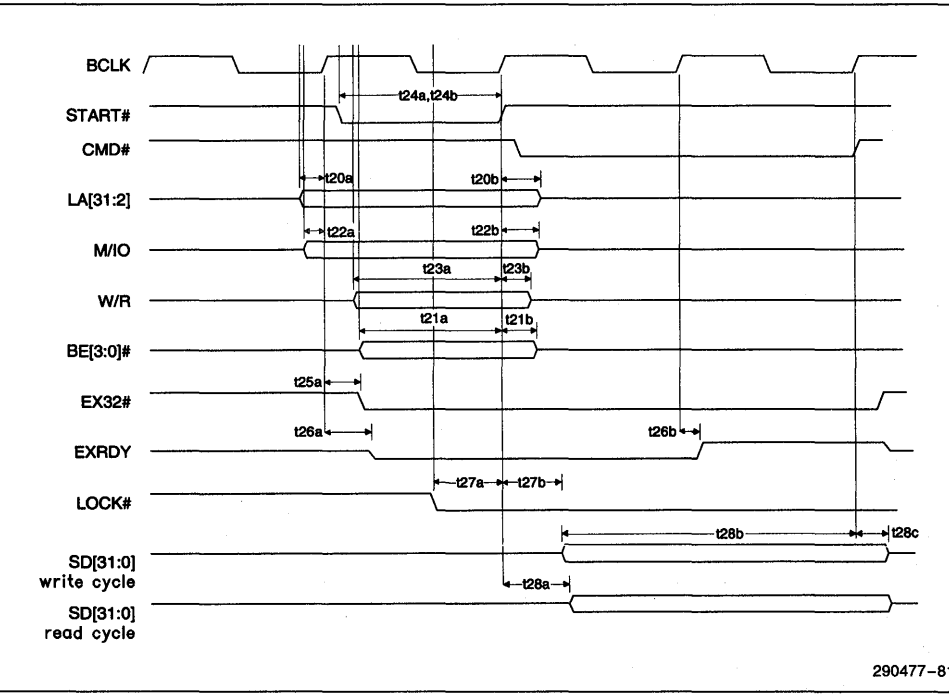


Figure 11-3. EISA Slave

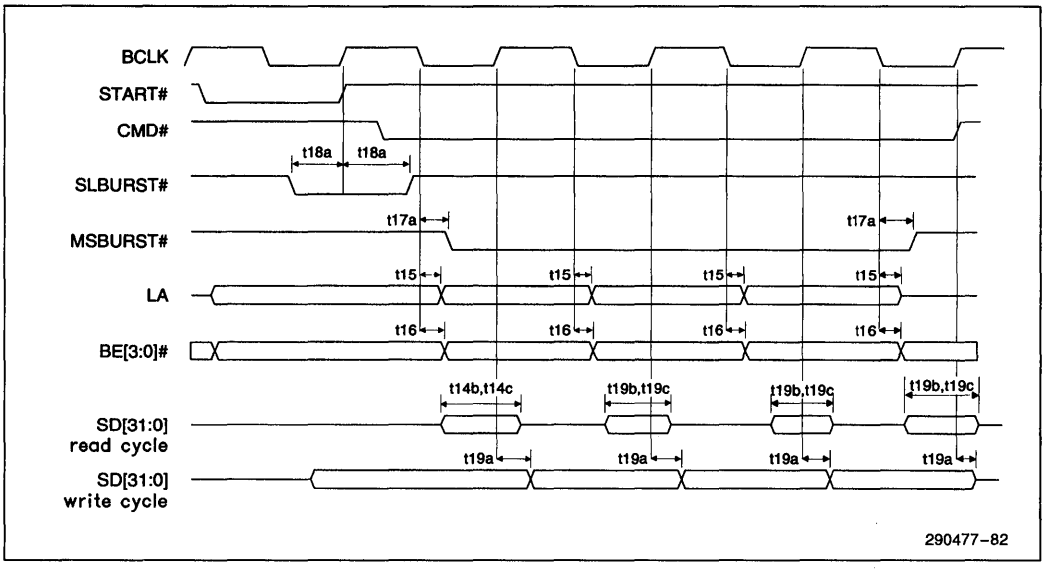


Figure 11-4. EISA Master Burst

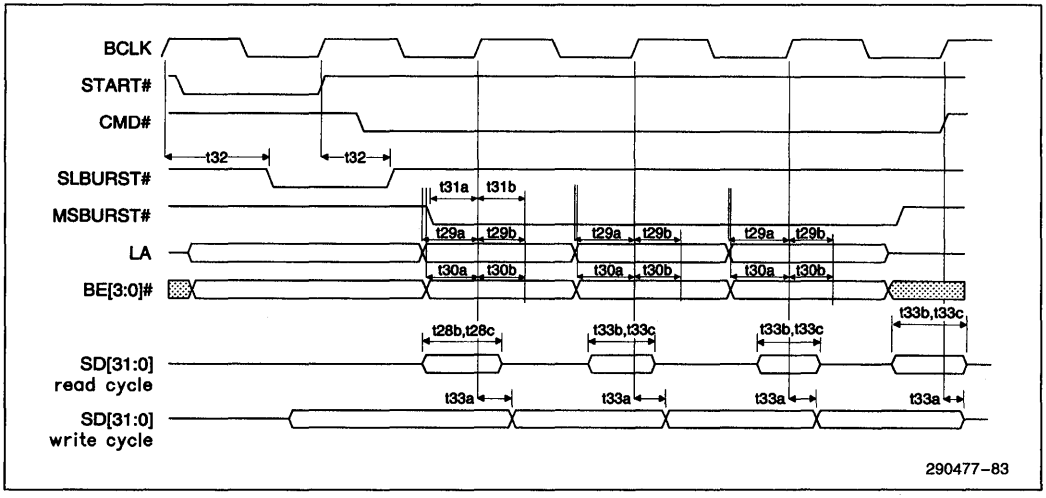


Figure 11-5. EISA Slave Burst

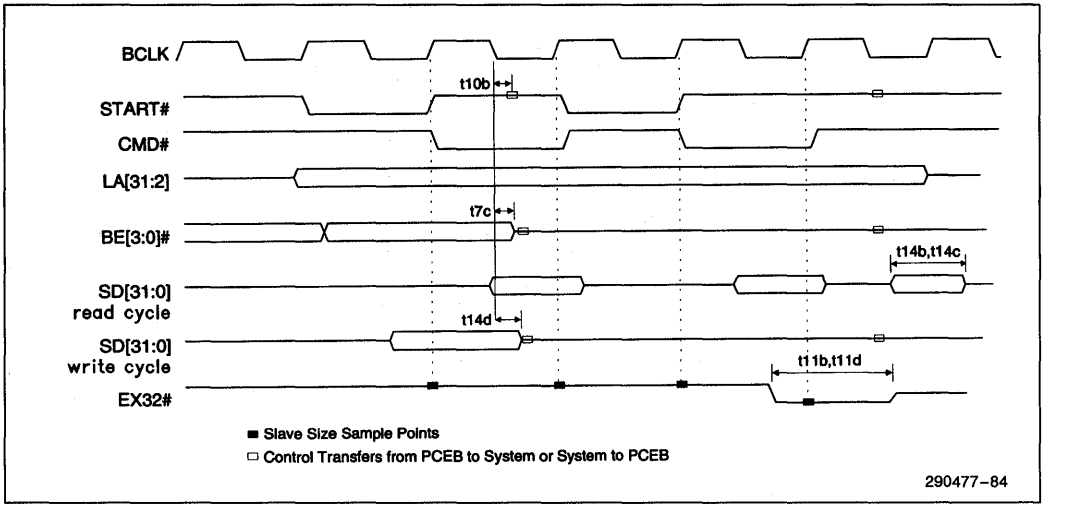


Figure 11-6. Backoff Cycles

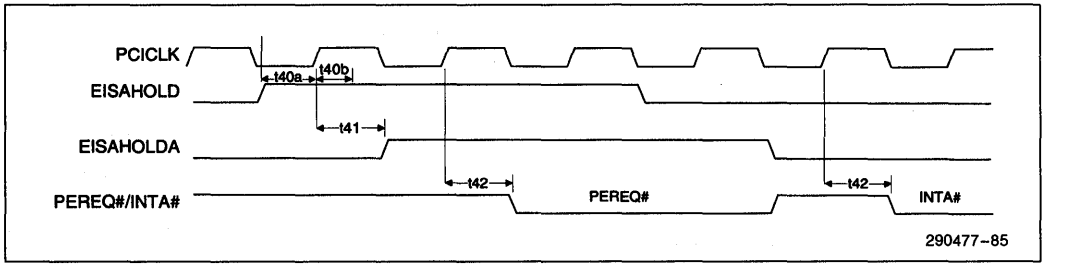


Figure 11-7. EISAHOLD Signals

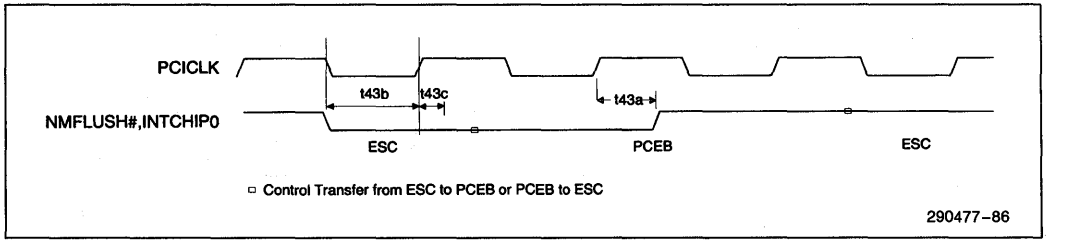


Figure 11-8. Flush Requests

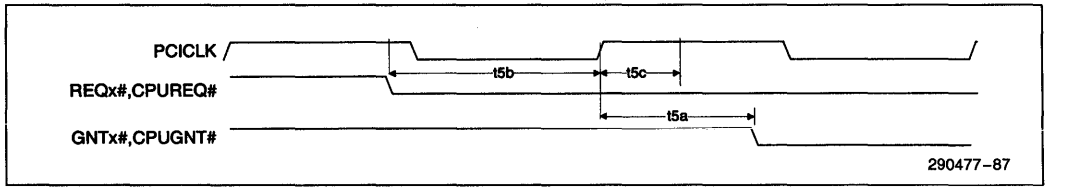


Figure 11-9. PCI Bus Arbitration

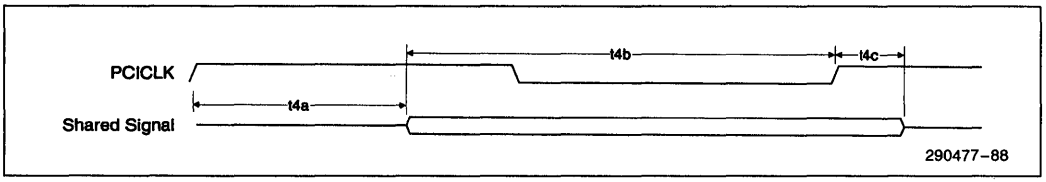


Figure 11-10. PCI Shared Signals

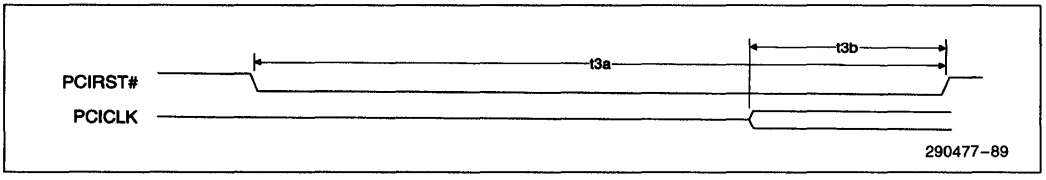


Figure 11-11. PCIRST# Signal

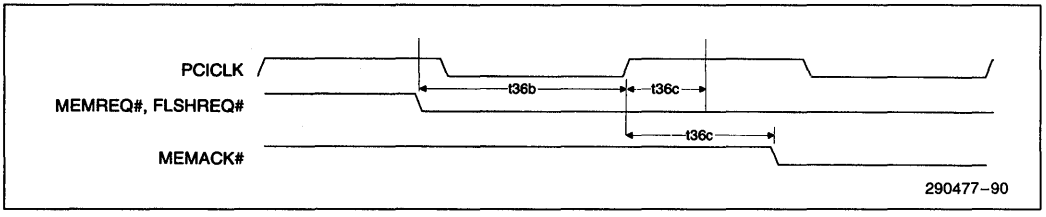


Figure 11-12. MEMREQ#, FLSHREQ#, MEMACK# Signals

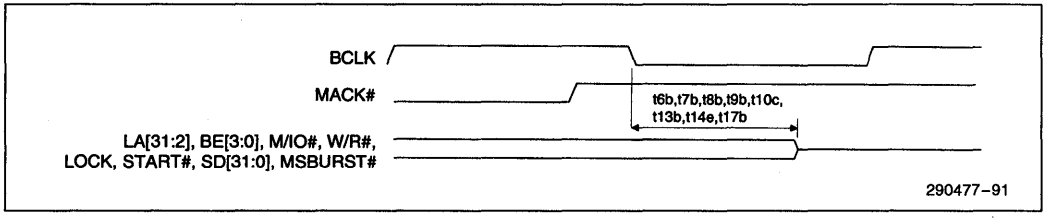


Figure 11-13. End of Master Mode

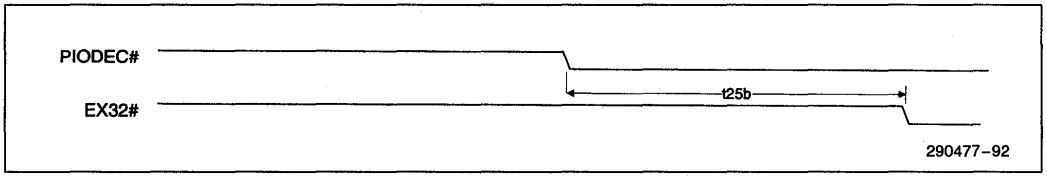


Figure 11-14. PIODEC# to EX32# Propogation

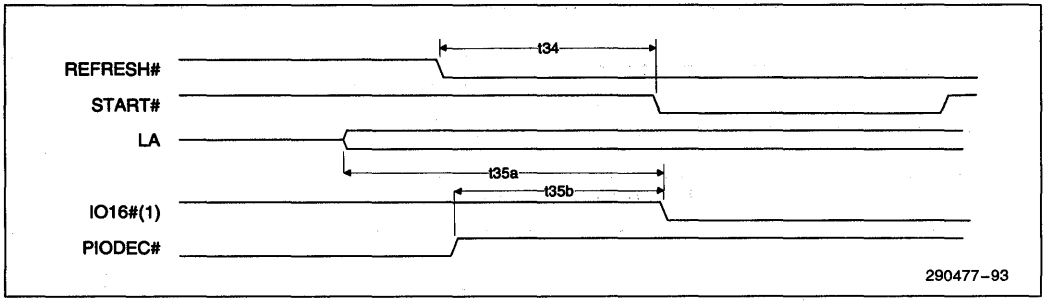


Figure 11-15. ISA Interface Signals

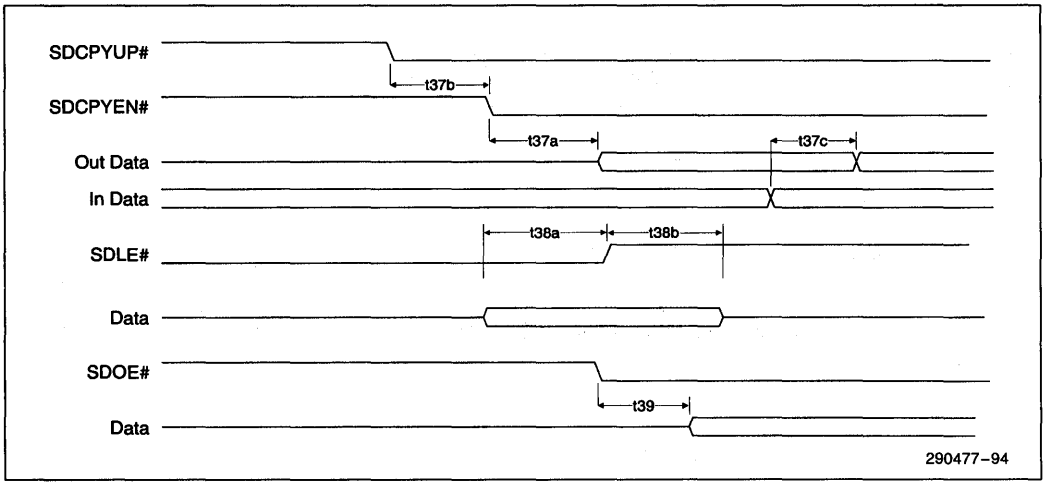


Figure 11-16. Data Swap Buffers

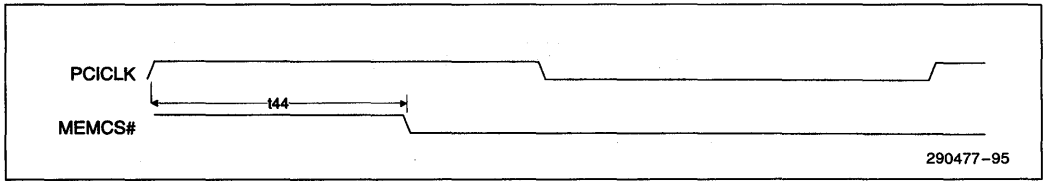


Figure 11-17. MEMCS# Signal

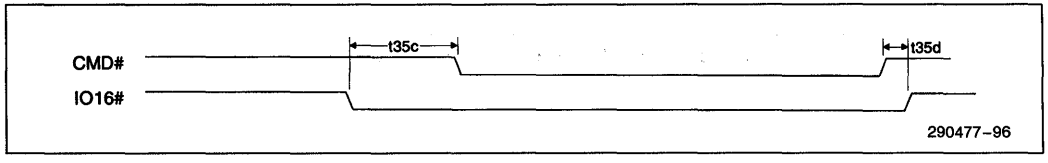
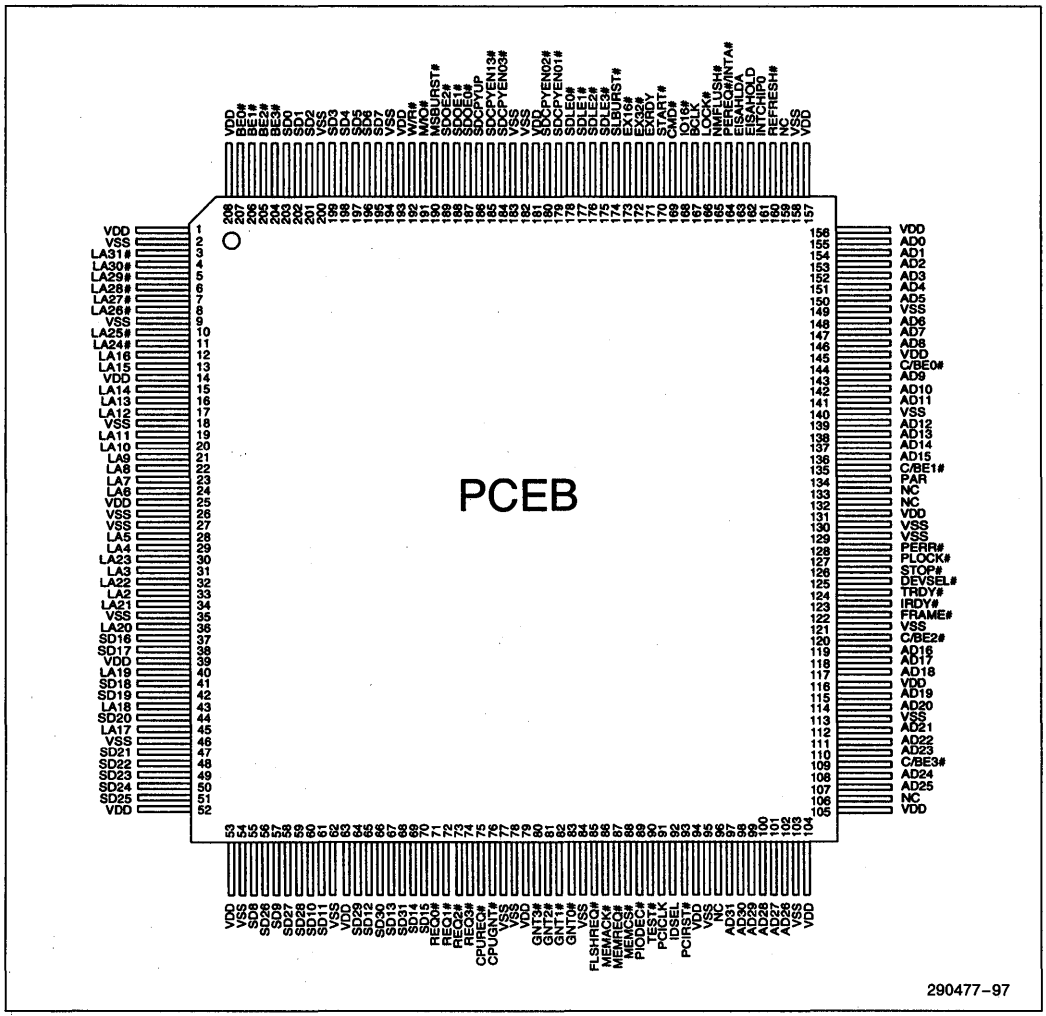


Figure 11-18. IO16# Signal

12.0 PINOUT AND PACKAGE INFORMATION

12.1 Pin Assignment



290477-97

Figure 12-1. Pinout

Table 12-1. 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
AD31	97	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

Name	Pin#	Type
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
INTCHIP0	161	t/s
IO16#	168	o/d
IRDY#	123	s/t/s
LA2	33	t/s
LA3	31	t/s
LA4	29	t/s
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

Name	Pin#	Type
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
MSBURST#	190	t/s
NC	96	NC
NC	106	NC
NC	132	NC
NC	133	NC
NC	159	NC
NMFLUSH#	165	t/s
PAR	134	t/s
PCICLK	91	in
PCIRST#	93	in
PEREQ#/ INTA#	164	out
PERR#	128	s/o/d
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 12-1. 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

Name	Pin#	Type
SDOE0#	187	in
SDOE1#	188	in
SDOE2#	189	in
SLBURST#	174	t/s
START#	170	t/s
STOP#	126	s/t/s
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

Name	Pin#	Type
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
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
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 12-2. 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
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
44	SD20	t/s
45	LA17	t/s

Pin#	Name	Type
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
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

Pin#	Name	Type
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
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/o/d
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

Table 12-2. Numerical PCEB Pin Assignment (Continued)

Pin#	Name	Type
136	AD15	t/s
137	AD14	t/s
138	AD13	t/s
139	AD12	t/s
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	NC	NC
160	REFRESH#	in

Pin#	Name	Type
161	INTCHIP0	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
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

Pin#	Name	Type
185	SDCPYEN13#	in
186	SDCPYUP	in
187	SDOE0#	in
188	SDOE1#	in
189	SDOE2#	in
190	MSBURST#	t/s
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

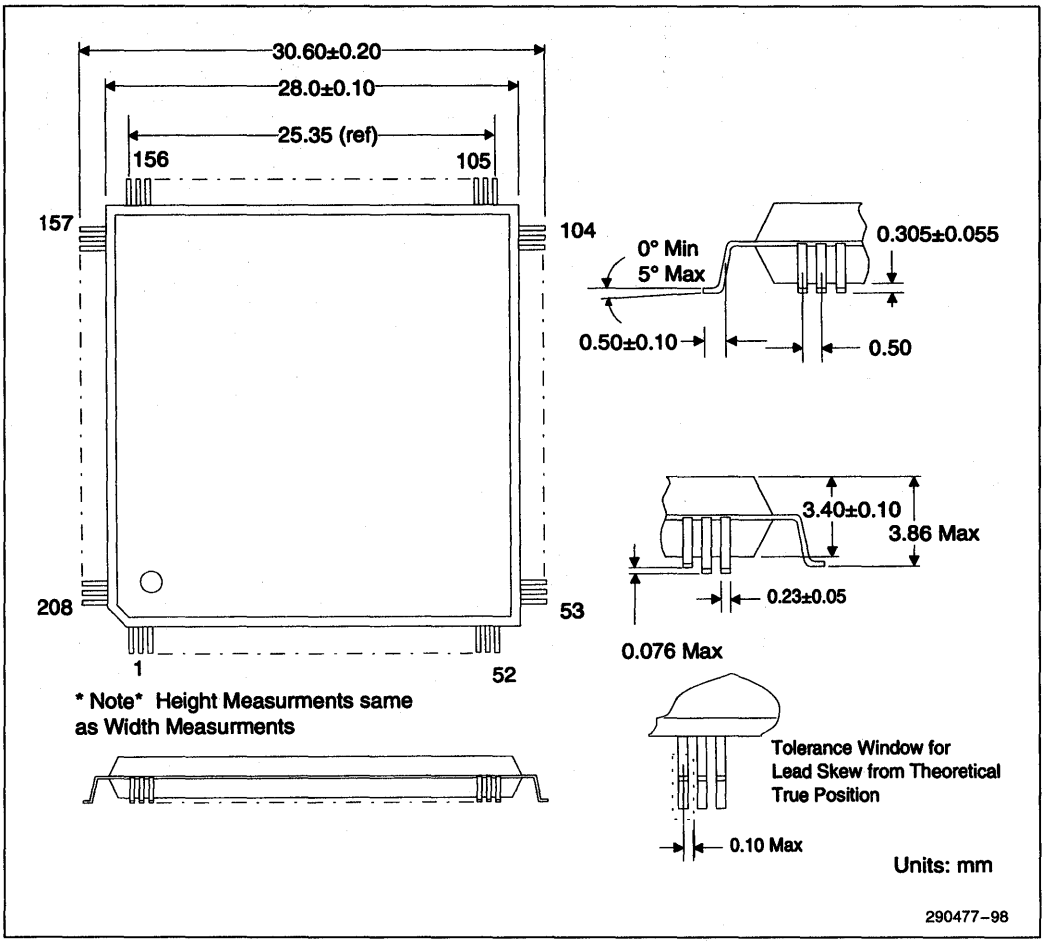


Figure 12-2. 208 Pin Plastic Quad Flat Pack (PQFP) Dimensions

Terminology

Assembly/Disassembly: This occurs when the master/slave data sizes are mismatched. The ESC/PCEB runs multiple cycles to route bytes to the appropriate byte lanes (byte swapping). For example, if a PCI agent (i.e., 32-bit master) is accessing an EISA or an ISA 8-bit slave, the ESC/PCEB will run four cycles to the ISA 8-bit slave and route the bytes to appropriate byte lanes.

Bus Lock Mode: This is the mode when the entire PCI Bus is locked.

Data Line: In the case of the PCEB, data line (or line) denotes one line of the 4-line internal PCEB Line Buffer.

Data Size Translation: This is performed by the PCEB/ESC when the master and slave data bus sizes do not match (i.e., 32-bit master/8-bit slave). During a data size translation, the PCEB/ESC will perform one or more of the following operations, depending on the master/slave data size combination, master/slave type (PCI/EISA/ISA), transfer direction (Read/Write), and the number of byte enables asserted: data assembly, data coping (up or down), or data re-drive.

DMA Device: A DMA device requests service by asserting DREQx. During DMA transfers, the data is transferred a memory slave (i.e., DMA Buffers internal to PCEB or memory on EISA/ISA bus) and an I/O slave (the I/O device is always on EISA/ISA bus). The I/O slave device is referred to as the DMA device.

EISA Bus: The EISA Bus (Extended ISA Bus) is a superset of the ISA bus. It includes all ISA bus features, along with extensions to enhance performance and capabilities.

EISA Master: A 16-bit or 32-bit bus master that uses the EISA signal set to generate memory or I/O cycles. The ESC component converts the EISA control signals to ISA signals, when necessary.

EISA Slave: An 8-bit, 16-bit, or 32-bit memory I/O slave device that uses the extended signals set of the EISA Bus to accept cycles from various masters. An EISA slave returns information about its type and data width using extended and ISA signals.

ESC: Integrated EISA Peripherals. This component is connected to the EISA/ISA bus and to the PCEB component. The ESC works in tandem with the PCEB component to translate bus protocols between the PCI and EISA/ISA busses.

Flush: Transfer the contents of the buffer to its destination. In the case of the Line Buffers, this term means, transfer the content of the Line Buffer to PCI memory. In case of the Posted Write Buffers (PWBs), it means transfer the contents to its EISA/ISA memory destination.

GAT Mode: Guaranteed Access Time Mode. This mode is used to guarantee that the ISA 2.1 μs CHRDY is not violated.

Initiator: A PCI agent that initiates a PCI cycle (i.e., a PCI master).

I/O Subsystem: This refers to the PCI-EISA Bridge and all the I/O and memory devices attached to the EISA Bus. The PCEB and ESC are components of this I/O subsystem.

ISA Bus: The bus used in the Industry Standard Architecture compatible computers. In the context of an EISA system, it refers to the ISA subset of the EISA Bus. For more details, refer to the IEEE P996 document.

ISA Master: A 16-bit master that uses the ISA subset of the EISA Bus for generation of memory or I/O cycles. This device must understand 8-bit or 16-bit ISA slaves, and route data to the appropriate byte lanes. It is not required to handle any of the signals associated with the extended portion of the EISA Bus. The ESC converted the ISA control signals to EISA signals, when necessary.

ISA Slave: An 8-bit or 16-bit slave that uses the ISA subset of the EISA Bus to accept cycles from various masters. It returns ISA signals to indicate its type and data width.

Line Buffer: This denotes the 4x16 byte internal PCEB Line Buffers.

Memory Subsystem: This consists of the second level (L2) Cache (Cache Controller and SRAMs), the memory (DRAMs, DRAM Controller, Data Buffers), and PCI Bridge.

Mis-matched Data Size: A master and slave that do not have equal data sizes (i.e., PCI agent accessing an 8-bit ISA slave, or a 16-bit DMA device accessing a 32-bit EISA memory slave).

PCEB: PCI-EISA Bridge. This component is connected to the PCI and EISA Buses. The PCEB works in tandem with the ESC component to translate bus protocols between the PCI and the EISA/ISA bus.

PCI: Peripheral Component Interconnect. This is the physical interconnect mechanism intended for use between highly integrated peripheral controller components and processor/memory systems. PCI is a multiplexed version of the Intel 486 bus, with control mechanism modified and extended for optimal I/O support. Refer to PCI Specification Revision 1.0 for more details.

PCI Agent: A 32-bit master that resides on the PCI Bus.

Posted Write Buffers: These are 4-byte wide unidirectional buffers used in the PCEB for PCI master access to EISA/ISA slaves.

Re-drive: This occurs when both the master and slave are on the EISA/ISA bus, and the master/slave data size combination is mismatched (i.e., 32-bit EISA master accessing an 8-bit ISA slave). During a re-drive cycle, the data is latched from the EISA/ISA bus and then driven back onto the appropriate EISA/ISA byte lane.

Read State: Data residing in the Line Buffers has been READ from the PCI memory.

Target: The destination of the PCI cycle (i.e., a PCI slave).

Target Abort: This resembles the RETRY, though the Target must deassert DEVSEL# along with the assertion of STOP#.

Turn-around Cycle: This is a required PCI cycle when one or more signals are driven by more than one agent. This cycle is required to avoid contention when one agent stops driving a signal and another agent begins. A turn-around cycle must last at least one clock.

82378IB
SYSTEM I/O (SIO)

- **Provides the Bridge Between the PCI Bus and ISA Bus**
- **100% PCI and ISA Compatible**
  - **— PCI and ISA Master/Slave Interface**
  - **— Directly Drives 10 PCI Loads and 6 ISA Slots**
  - **— Supports PCI at 25 MHz and 33 MHz**
  - **— Supports ISA from 6 MHz to 8.33 MHz**
- **Enhanced DMA Functions**
  - **— Scatter/Gather**
  - **— Fast DMA Type A, B, and F**
  - **— Compatible DMA Transfers**
  - **— 32-bit Addressability**
  - **— Seven Independently Programmable Channels**
  - **— Functionality of Two 82C37A DMA Controllers**
- **Integrated Data Buffers to Improve Performance**
  - **— 8-byte DMA/ISA Master Line Buffer**
  - **— 32-bit Posted Memory Write Buffer to ISA**
- **Integrated 16-bit BIOS Timer**
- **Arbitration for PCI Devices**
  - **— Four PCI Masters are Supported**
  - **— Fixed, Rotating, or a Combination of the Two**
- **Arbitration for ISA Devices**
  - **— ISA Masters**
  - **— DMA and Refresh**
- **Utility Bus (X-Bus) Peripheral Support**
  - **— Provides Chip Select Decode**
  - **— Controls Lower X-Bus Data Byte Transceiver**
  - **— Integrates Port 92, Mouse Interrupt, Coprocessor Error Reporting**
- **Integrates the Functionality of one 82C54 Timer**
  - **— System Timer**
  - **— Refresh Request**
  - **— Speaker Tone Output**
- **Integrates the Functionality of two 82C59 Interrupt Controllers**
  - **— 14 Interrupts supported**
- **Non-Maskable Interrupts (NMI)**
  - **— PCI System Errors**
  - **— ISA Parity Errors**

The 82378IB System I/O (SIO) component provides the bridge between the PCI bus and the ISA expansion bus. The SIO also integrates many of the common I/O functions found in today's ISA based PC systems. The SIO incorporates the logic for a PCI interface (master and slave), ISA interface (master and slave), enhanced seven channel DMA controller that supports fast DMA transfers and Scatter/Gather, data buffers to isolate the PCI bus from the ISA bus and to enhance performance, PCI and ISA arbitration, 14 level interrupt controller, a 16-bit BIOS timer, three programmable timer/counters, and non-maskable-interrupt (NMI) control logic. The SIO also provides decode for peripheral devices such as the Flash BIOS, Real Time Clock, Keyboard/Mouse Controller, Floppy Controller, two Serial Ports, one Parallel Port, and IDE Hard Disk Drive.

April 1993
Order Number: 290473-002

82378IB System I/O (SIO)

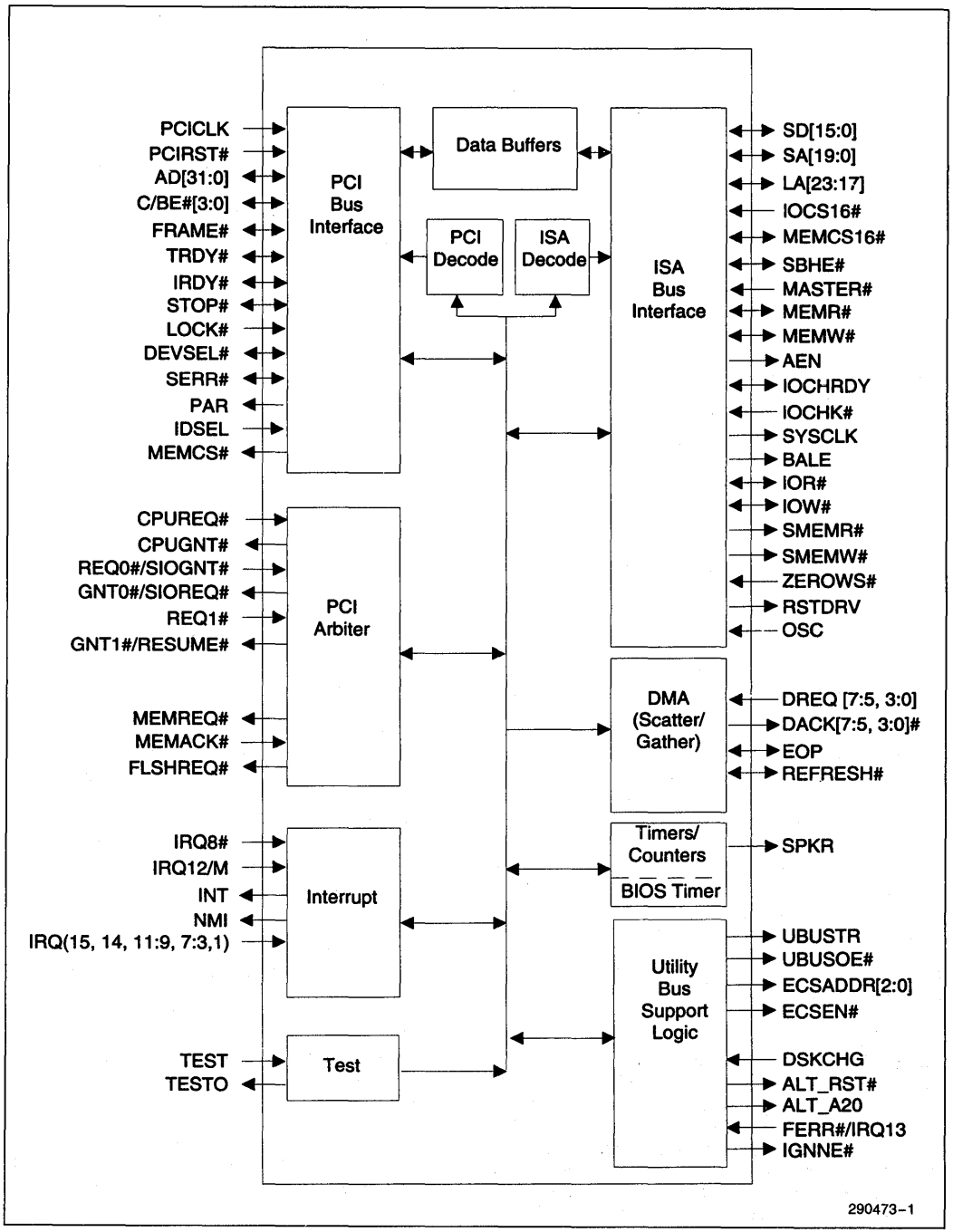
CONTENTS

CONTENTS

CONTENTS PAGE

CONTENTS

CONTENTS



SIO Component Block Diagram

1.0 ARCHITECTURAL OVERVIEW

The major functions of the SIO component are broken up into blocks as shown in the preceding figure. A description of each block is provided below.

PCI Bus Interface:
The PCI Bus Interface provides the interface between the SIO and the PCI bus. The SIO provides both a master and slave interface to the PCI bus. As a PCI master, the SIO runs cycles on behalf of DMA, ISA masters, and the internal data buffer management logic when buffer flushing is required. The SIO will burst a maximum of two dwords when reading from PCI memory, and one dword when writing to PCI memory. The SIO does not generate PCI I/O cycles as a master. As a PCI slave, the SIO accepts cycles initiated by PCI masters targeted for the SIO's internal register set or the ISA bus. The SIO will accept a maximum of one data transaction before terminating the transaction. This supports the Incremental Latency Mechanism as defined in the Peripheral Component Interconnect (PCI) Specification.

As a master, the SIO generates address and command signal (C/BE#) parity for read and write cycles, and data parity for write cycles. As a slave, the SIO generates data parity for read cycles. Parity checking is not supported. The SIO also provides support for system error reporting by generating a Non-Maskable- Interrupt (NMI) when SERR# is driven active.

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

The SIO directly supports the PCI Interface running at either 25 Mhz or 33 Mhz. If a frequency of less than 33 Mhz is required (not including 25 Mhz), a SYSCLK divisor value (as indicated in the ISA Clock Divisor Register) must be selected that guarantees that the ISA bus frequency does not violate the 6 Mhz to 8.33 Mhz SYSCLK range.

PCI Arbiter:
The PCI arbiter provides support for four PCI masters; the Host Bridge, SIO, and two PCI masters. The arbiter can be programmed for a purely rotating scheme, fixed, or a combination of the two. The Arbiter can also be programmed to support bus parking. This gives the Host Bridge default access to the PCI bus when no other device is requesting service. The arbiter can be disabled if an external arbiter is used.

PCI Decode/ISA Decode:
The SIO contains two address decoders; one to decode PCI initiated cycles and one to decode ISA master and DMA initiated cycles. Two decoders are used to allow the PCI and ISA busses to run concurrently.

The SIO is also programmable to provide address decode on behalf of the Host Bridge. When programmed, the SIO monitors the PCI and ISA address busses, and generates a memory chip select signal (MEMCS#) indicating that the current cycle is targeted for system memory residing behind the Host Bridge. This feature can be disabled through software.

Data Buffers:
To isolate the slower ISA bus from the PCI bus, the SIO provides two types of data buffers. One dword deep posted write buffer is provided for the posting of PCI initiated memory write cycles to the ISA bus. The second buffer is a bi- directional, 8 byte line buffer used for ISA master and DMA accesses to the PCI bus. All DMA and ISA master read and write cycles go through the 8 byte line buffer.

The data buffers also provide the data assembly or disassembly when needed for transactions between the PCI and ISA busses.

Buffering is programmable and can be enabled or disabled through software.

ISA Bus Interface:
The SIO incorporates a fully ISA-bus compatible master and slave interface. The SIO directly drives six ISA slots without external data or address buffering. The ISA interface also provides byte swap logic, I/O recovery support, wait- state generation, and SYSCLK generation. The SIO supports ISA bus frequencies from 6 to 8.33 Mhz.

As an ISA master, the SIO generates cycles on behalf of DMA, Refresh, and PCI master initiated cycles. The SIO supports compressed cycles when accessing ISA slaves (i.e. ZEROWS# asserted). As an ISA slave, the SIO accepts ISA master accesses targeted for the SIO's internal register set or ISA master memory cycles targeted for the PCI bus. The SIO does not support ISA master initiated I/O cycles targeted for the PCI bus.

The SIO also monitors ISA master to ISA slave cycles to generate SMEMR# or SMEMW#, and to support data byte swapping, if necessary.

DMA:
The DMA controller incorporates the functionality of two 82C37 DMA controllers with seven independently programmable channels. Each channel can be programmed for 8-bit or 16-bit DMA device size, and ISA-compatible or fast DMA type "A", type "B", or type F" timings. Full 32-bit addressing is supported as an extension of the ISA-compatible specification. The DMA controller is also responsible for generating ISA 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 system 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 read.

Timer Block:
The timer block contains three counters that are equivalent in function to those found in one 82C54 programmable interval timer. These three counters are combined to provide the System Timer function, Refresh Request, and speaker tone. The three counters use the 14.31818 Mhz OSC input for a clock source.

In addition to the three counters, the SIO provides a programmable 16-bit BIOS timer. This timer can be used by BIOS software to implement timing loops. The timer uses the ISA system clock (SYSCLK) divided by 8 as a clock source. An 8 to 1 ratio between the SYSCLK and the BIOS timer clock is always maintained. The accuracy of the BIOS timer is ± 1 msec.

Utility Bus (X-Bus) Logic:
The SIO provides four encoded chip selects that are decoded externally to provide chip selects for Flash BIOS, Real Time Clock, Keyboard/Mouse Controller, Floppy Controller, two Serial Ports, one Parallel Port, and an IDE Hard Disk Drive. The SIO provides the control for the buffer that isolates the lower 8-bits of the Utility Bus from the lower 8-bits of the ISA bus.

In addition to providing the encoded chip selects and Utility Bus buffer control, the SIO also provides Port 92 functions (Alternate Reset and Alternate A20), Coprocessor error reporting, the Floppy DSKCHG function, and a mouse interrupt input.

Interrupt Controller Block:
The SIO provides an ISA compatible interrupt controller that incorporates the functionality of two 82C59 interrupt controllers. The two interrupt controllers are cascaded so that 14 external and two internal interrupts are possible.

Test:
The test block provides the interface to the test circuitry within the SIO. The Test input can be used to tri-state all of the SIO outputs.

2.0 PIN ASSIGNMENT

The SIO package is a 208 pin plastic Quad flat pack (PQFP). The package signals are shown in Figure 2-1 and listed in Table 2-1. The following notations are used to describe pin types.

Signal	Description
I	**Input** is a standard input-only signal.
O	**Totem Pole Output** is a standard active driver.
OD	**Open Drain** Input/Output
IO	**Input/Output** is a bi-directional, tri-state 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. A pull-up sustains the inactive state until another agent drives it and is provided by the central resource.
t/s/o	Tri-State Output

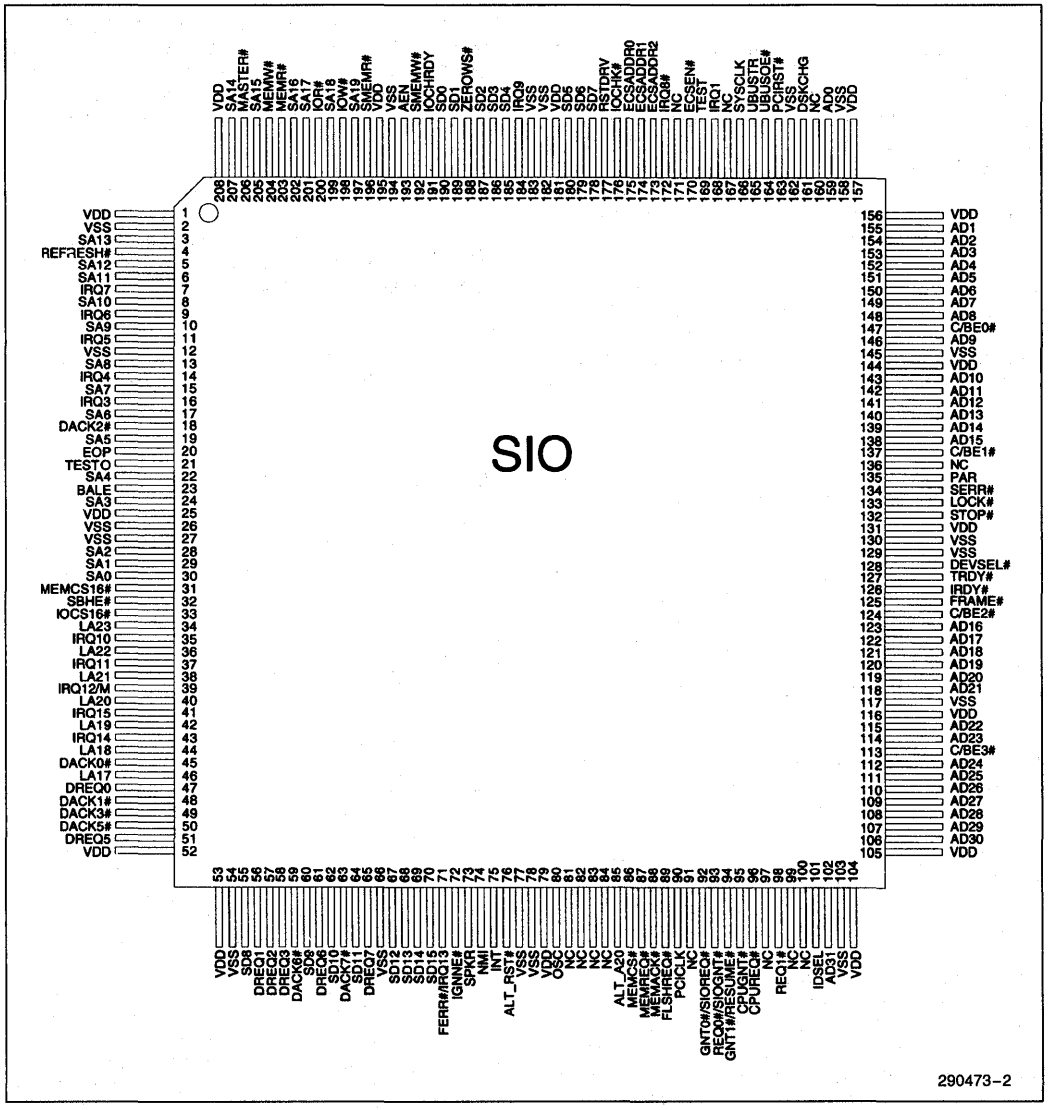


Figure 2-1. SIO Package Pinout Diagram

Table 2-1. Alphabetical SIO Pin Assignment List

Pin Name	Pin #	Type
AD0	159	I/O
AD1	155	I/O
AD2	154	I/O
AD3	153	I/O
AD4	152	I/O
AD5	151	I/O
AD6	150	I/O
AD7	149	I/O
AD8	148	I/O
AD9	146	I/O
AD10	143	I/O
AD11	142	I/O
AD12	141	I/O
AD13	140	I/O
AD14	139	I/O
AD15	138	I/O
AD16	123	I/O
AD17	122	I/O
AD18	121	I/O
AD19	120	I/O
AD20	119	I/O
AD21	118	I/O
AD22	115	I/O
AD23	114	I/O
AD24	112	I/O
AD25	111	I/O
AD26	110	I/O
AD27	109	I/O
AD28	108	I/O
AD29	107	I/O
AD30	106	I/O
AD31	102	I/O
AEN	193	O
ALT_A20	85	O
ALT_RST#	76	O
BALE	23	O

Pin Name	Pin #	Type
C/BE0#	147	I/O
C/BE1#	137	I/O
C/BE2#	124	I/O
C/BE3#	113	I/O
CPUGNT#	95	t/s/o
CPUREQ#	96	I
DACK0#	45	O
DACK1#	48	O
DACK2#	18	O
DACK3#	49	O
DACK5#	50	O
DACK6#	59	O
DACK7#	63	O
DEVSEL#	128	I/O (s/t/s)
DREQ0	47	I
DREQ1	56	I
DREQ2	57	I
DREQ3	58	I
DREQ5	51	I
DREQ6	61	I
DREQ7	65	I
DSKCHG	161	I
ECSADDR0	175	O
ECSADDR1	174	O
ECSADDR2	173	O
ECSEN#	170	O
EOP	20	I/O
FERR#/ IRQ13	71	I
FLSHREQ#	89	t/s/o
FRAME#	125	I/O (s/t/s)
GNT0#/ SIOREQ#	92	t/s/o
GNT1#/ RESUME#	94	t/s/o

Pin Name	Pin #	Type
IDSEL	101	I
IGNNE#	72	O
INT	75	O
IOCHK#	176	I
IOCHRDY	191	I/O
IOCS16#	33	I
IOR#	200	I/O
IOW#	198	I/O
IRDY#	126	I/O (s/t/s)
IRQ1	168	I
IRQ3	16	I
IRQ4	14	I
IRQ5	11	I
IRQ6	9	I
IRQ7	7	I
IRQ8#	172	I
IRQ9	184	I
IRQ10	35	I
IRQ11	37	I
IRQ12/M	39	I
IRQ14	43	I
IRQ15	41	I
LA17	46	I/O
LA18	44	I/O
LA19	42	I/O
LA20	40	I/O
LA21	38	I/O
LA22	36	I/O
LA23	34	I/O
LOCK#	133	I (s/t/s)
MASTER#	206	I
MEMACK#	88	I
MEMCS#	86	O
MEMCS16#	31	I/O (o/d)

Table 2-1. Alphabetical SIO Pin Assignment List (Continued)

Pin Name	Pin #	Type
MEMR#	203	I/O
MEMREQ#	87	t/s/o
MEMW#	204	I/O
NMI	74	O
OSC	80	I
PAR	135	O
PCICLK	90	I
PCIRST#	163	I
REFRESH#	4	I/O
REQ0#/ SIOGNT#	93	I
REQ1#	98	I
Reserved	91	NC
Reserved	97	NC
Reserved	81	NC
Reserved	82	NC
Reserved	83	NC
Reserved	84	NC
Reserved	99	NC
Reserved	100	NC
Reserved	136	NC
Reserved	160	NC
Reserved	167	NC
Reserved	171	NC
RSTDRV	177	O
SA0	30	I/O
SA1	29	I/O
SA2	28	I/O
SA3	24	I/O
SA4	22	I/O
SA5	19	I/O
SA6	17	I/O
SA7	15	I/O
SA8	13	I/O
SA9	10	I/O
SA10	8	I/O

Pin Name	Pin #	Type
SA11	6	I/O
SA12	5	I/O
SA13	3	I/O
SA14	207	I/O
SA15	205	I/O
SA16	202	I/O
SA17	201	I/O
SA18	199	I/O
SA19	197	I/O
SBHE#	32	I/O
SD0	190	I/O
SD1	189	I/O
SD2	187	I/O
SD3	186	I/O
SD4	185	I/O
SD5	180	I/O
SD6	179	I/O
SD7	178	I/O
SD8	55	I/O
SD9	60	I/O
SD10	62	I/O
SD11	64	I/O
SD12	67	I/O
SD13	68	I/O
SD14	69	I/O
SD15	70	I/O
SERR#	134	I
SMEMR#	196	O
SMEMW#	192	O
SPKR	73	O
STOP#	132	I/O (s/t/s)
SYSCLK	166	O
TEST	169	I
TESTO	21	O
TRDY#	127	I/O (s/t/s)

Pin Name	Pin #	Type
UBUSOE#	164	O
UBUSTR	165	O
ZEROWS#	188	I
V_{DD}	1	V
V_{DD}	25	V
V_{DD}	52	V
V_{DD}	53	V
V_{DD}	79	V
V_{DD}	104	V
V_{DD}	105	V
V_{DD}	116	V
V_{DD}	131	V
V_{DD}	144	V
V_{DD}	156	V
V_{DD}	157	V
V_{DD}	181	V
V_{DD}	195	V
V_{DD}	208	V
V_{SS}	2	V
V_{SS}	12	V
V_{SS}	26	V
V_{SS}	27	V
V_{SS}	54	V
V_{SS}	66	V
V_{SS}	77	V
V_{SS}	78	V
V_{SS}	103	V
V_{SS}	117	V
V_{SS}	129	V
V_{SS}	130	V
V_{SS}	145	V
V_{SS}	158	V
V_{SS}	162	V
V_{SS}	182	V
V_{SS}	183	V
V_{SS}	194	V

Table 2-2. Numerical SIO Pin Assignment List

Pin Name	Pin #	Type
V_{DD}	1	V
V_{SS}	2	V
SA13	3	I/O
REFRESH#	4	I/O
SA12	5	I/O
SA11	6	I/O
IRQ7	7	I
SA10	8	I/O
IRQ6	9	I
SA9	10	I/O
IRQ5	11	I
V_{SS}	12	V
SA8	13	I/O
IRQ4	14	I
SA7	15	I/O
IRQ3	16	I
SA6	17	I/O
DACK2#	18	O
SA5	19	I/O
EOP	20	I/O
TESTO	21	O
SA4	22	I/O
BALE	23	O
SA3	24	I/O
V_{DD}	25	V
V_{SS}	26	V
V_{SS}	27	V
SA2	28	I/O
SA1	29	I/O
SA0	30	I/O
MEMCS16#	31	I/O (o/d)
SBHE#	32	I/O
IOCS16#	33	I
LA23	34	I/O
IRQ10	35	I
LA22	36	I/O

Pin Name	Pin #	Type
IRQ11	37	I
LA21	38	I/O
IRQ12/M	39	I
LA20	40	I/O
IRQ15	41	I
LA19	42	I/O
IRQ14	43	I
LA18	44	I/O
DACK0#	45	O
LA17	46	I/O
DREQ0	47	I
DACK1#	48	O
DACK3#	49	O
DACK5#	50	O
DREQ5	51	I
V_{DD}	52	V
V_{DD}	53	V
V_{SS}	54	V
SD8	55	I/O
DREQ1	56	I
DREQ2	57	I
DREQ3	58	I
DACK6#	59	O
SD9	60	I/O
DREQ6	61	I
SD10	62	I/O
DACK7#	63	O
SD11	64	I/O
DREQ7	65	I
V_{SS}	66	V
SD12	67	I/O
SD13	68	I/O
SD14	69	I/O
SD15	70	I/O
FERR#/ IRQ13	71	I

Pin Name	Pin #	Type
IGNNE#	72	O
SPKR	73	O
NMI	74	O
INT	75	O
ALT_RST#	76	O
V_{SS}	77	V
V_{SS}	78	V
V_{DD}	79	VI
OSC	80	I
Reserved	81	NC
Reserved	82	NC
Reserved	83	NC
Reserved	84	NC
ALT_A20	85	O
MEMCS#	86	O
MEMREQ#	87	t/s/o
MEMACK#	88	I
FLSHREQ#	89	t/s/o
PCICLK	90	I
Reserved	91	NC
GNT0#/ SIOREQ#	92	t/s/o
REQ0#/ SIOGNT#	93	I
GNT1#/ RESUME#	94	t/s/o
CPUGNT#	95	t/s/o
CPUREQ#	96	I
Reserved	97	NC
REQ1#	98	I
Reserved	99	NC
Reserved	100	NC
IDSEL	101	I
AD31	102	I/O
V_{SS}	103	V
V_{DD}	104	V
V_{DD}	105	V

Table 2-2. Numerical SIO Pin Assignment List (Continued)

Pin Name	Pin #	Type
AD30	106	I/O
AD29	107	I/O
AD28	108	I/O
AD27	109	I/O
AD26	110	I/O
AD25	111	I/O
AD24	112	I/O
C/BE3#	113	I/O
AD23	114	I/O
AD22	115	I/O
V_{DD}	116	V
V_{SS}	117	V
AD21	118	I/O
AD20	119	I/O
AD19	120	I/O
AD18	121	I/O
AD17	122	I/O
AD16	123	I/O
C/BE2#	124	I/O
FRAME#	125	I/O (s/t/s)
IRDY#	126	I/O (s/t/s)
TRDY#	127	I/O (s/t/s)
DEVSEL#	128	I/O (s/t/s)
V_{SS}	129	V
V_{SS}	130	V
V_{DD}	131	V
STOP#	132	I/O (s/t/s)
LOCK#	133	I (s/t/s)
SERR#	134	I
PAR	135	O
Reserved	136	NC
C/BE1#	137	I/O

Pin Name	Pin #	Type
AD15	138	I/O
AD14	139	I/O
AD13	140	I/O
AD12	141	I/O
AD11	142	I/O
AD10	143	I/O
V_{DD}	144	V
V_{SS}	145	V
AD9	146	I/O
C/BE0#	147	I/O
AD8	148	I/O
AD7	149	I/O
AD6	150	I/O
AD5	151	I/O
AD4	152	I/O
AD3	153	I/O
AD2	154	I/O
AD1	155	I/O
V_{DD}	156	V
V_{DD}	157	V
V_{SS}	158	V
AD0	159	I/O
Reserved	160	NC
DSKCHG	161	I
V_{SS}	162	V
PCIRST#	163	I
UBUSOE#	164	O
UBUSTR	165	O
SYSCLK	166	O
Reserved	167	NC
IRQ1	168	I
TEST	169	I
ECSEN#	170	O
Reserved	171	NC
IRQ8#	172	I

Pin Name	Pin #	Type
ECSADDR2	173	O
ECSADDR1	174	O
ECSADDR0	175	O
IOCHK#	176	I
RSTDRV	177	O
SD7	178	I/O
SD6	179	I/O
SD5	180	I/O
V_{DD}	181	V
V_{SS}	182	V
V_{SS}	183	V
IRQ9	184	I
SD4	185	I/O
SD3	186	I/O
SD2	187	I/O
ZEROWS#	188	I
SDI	189	I/O
SD0	190	I/O
IOCHRDY	191	I/O
SMEMW#	192	O
AEN	193	O
V_{SS}	194	V
V_{DD}	195	V
SMEMR#	196	O
SA19	197	I/O
IOW#	198	I/O
SA18	199	I/O
IOR#	200	I/O
SA17	201	I/O
SA16	202	I/O
MEMR#	203	I/O
MEMW#	204	I/O
SA15	205	I/O
MASTER#	206	I
SA14	207	I/O
V_{DD}	208	V

3.0 SIGNAL DESCRIPTION

This section contains a detailed description of each signal. The signals are arranged in functional groups according to the interface.

Note that 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.

3.1 PCI Bus Interface Signals

Signal Name	Type	Description
PCICLK	I	**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 SIO include 25 MHz and 33 MHz.
PCIRST#	I	**PCI RESET**: PCIRST# forces the SIO to a known state. AD[31:0], C/BE[3:0]#, and PAR are always driven low by the SIO synchronously from the leading edge of PCIRST#. The SIO always tri-states these signals from the trailing edge of PCIRST#. If the internal arbiter is enabled (CPUREQ# sampled high on the trailing edge of PCIRST#), the SIO will drive these signals low again (synchronously 2-5 PCICLKs later) until the bus is given to another master. If the internal arbiter is disabled (CPUREQ# sampled low on the trailing edge of PCIRST#), these signals remain tri-stated until the SIO is required to drive them valid as a master or slave. FRAME#, IRDY#, TRDY#, STOP#, DEVSEL#, MEMREQ# FLSHREQ#, CPUGNT#, GNT0#/SIOREQ#, and GNT1#/RESUME# are tri-stated from the leading edge of PCIRST#. FRAME#, IRDY#, TRDY#, STOP#, and DEVSEL# remain tri-stated until driven by the SIO as either a master or a slave. MEMREQ#, FLSHREQ#, CPUGNT#, GNT0#/SIOREQ#, and GNT1#/RESUME# are tri-stated until driven by the SIO. After PCIRST#, MEMREQ# and FLSHREQ# are driven inactive asynchronously from PCIRST# inactive. CPUGNT#, GNT0#/SIOREQ#, and GNT1#/RESUME# are driven based on the arbitration scheme and the asserted REQx#'s. All registers are set to their default values. PCIRST# may be asynchronous to PCICLK when asserted or negated. Although asynchronous, negation must be a clean, bounce-free edge. Note that PCIRST# must be asserted for more than 1 μs.

3.1 PCI Bus Interface Signals (Continued)

Signal Name	Type	Description
AD[31:0]	I/O	**PCI ADDRESS/DATA**: AD[31:0] is a multiplexed address and data bus. During the first clock of a transaction, AD[31:0] contains a physical byte address (32 bits). During subsequent clocks, AD[31:0] contains data. An SIO 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). When the SIO is a target, AD[31:0] are inputs during the address phase of a transaction. During the following data phase(s), the SIO may be asked to supply data on AD[31:0] for a PCI read, or accept data for a PCI write. As a master, the SIO drives a valid address on AD[31:2] during the address phase, and drives write or latches read data on AD[31:0] during the data phase. The SIO always drives AD[1:0] low as a master during the address phase. AD[31:0] are always driven low by the SIO synchronously from the leading edge of PCIRST#. The SIO always tri-states AD[31:0] from the trailing edge of PCIRST#. If the internal arbiter is enabled (CPUREQ# sampled high on the trailing edge of PCIRST#), the SIO drives AD[31:0] low again (synchronously 2-5 PCICLKs later) until the bus is given to another master. If the internal arbiter is disabled (CPUREQ# sampled low on the trailing edge of PCIRST#), AD[31:0] remain tri-stated until the SIO is required to drive them valid as a master or slave. When the internal arbiter is enabled, the SIO acts as the central resource responsible for driving the AD[31:0] signals when no one is granted the PCI Bus and the bus is idle. When the internal arbiter is disabled, the SIO does not drive AD[31:0] as the central resource. The SIO is always responsible for driving AD[31:0] when it is granted the bus (SIOGNT# and idle bus) and as appropriate when it is the master of a transaction.
C/BE[3:0]#	I/O	**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. 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, C/BE[1] to byte 1, C/BE[2] to byte 2, and C/BE#[3] to byte 3. The SIO drives C/BE[3:0]# as an initiator of a PCI Bus cycle and monitors C/BE[3:0]# as a Target. C/BE[3:0]# are always driven low by the SIO synchronously from the leading edge of PCIRST#. The SIO always tri-states C/BE[3:0]# from the trailing edge of PCIRST#. If the internal arbiter is enabled (CPUREQ# sampled high on the trailing edge of PCIRST#), the SIO drives C/BE[3:0]# low again (synchronously 2-5 PCICLKs later) until the bus is given to another master. If the internal arbiter is disabled (CPUREQ# sampled low on the trailing edge of PCIRST#), C/BE[3:0]# remain tri-stated until the SIO is required to drive them valid as a master or slave. When the internal arbiter is enabled, the SIO acts as the central resource responsible for driving the C/BE[3:0]# signals when no one is granted the PCI Bus and the bus is idle. When the internal arbiter is disabled, the SIO does not drive C/BE[3:0]# as the central resource. The SIO is always responsible for driving C/BE[3:0]# when it is granted the bus (SIOGNT# and idle bus) and as appropriate when it is the master of a transaction.

3.1 PCI Bus Interface Signals (Continued)

Signal Name	Type	Description
FRAME#	I/O (s/t/s)	**CYCLE FRAME**: FRAME# is driven by the current master to indicate the beginning and duration of an access. FRAME# is asserted to indicate a bus transaction is beginning. While FRAME# is asserted data transfers continue. When FRAME# is negated the transaction is in the final data phase. FRAME# is an input to the SIO when the SIO is the target. FRAME# is an output when the SIO is the initiator. FRAME# is tri-stated from the leading edge of PCIRST#. FRAME# remains tri-stated until driven by the SIO as either a master or a slave.
TRDY#	I/O (s/t/s)	**TARGET READY**: TRDY# indicates a SIO's ability to complete the current data phase of the transaction. TRDY# is used in conjunction with IRDY#. A data phase is completed when both TRDY# and IRDY# are sampled asserted. During a read, TRDY# indicates that the SIO, as a target, has place valid data on AD[31:0]. During a write, it indicates the SIO, as a target is prepared to latch data. TRDY is an input to the SIO when the SIO is the initiator and an output when the SIO is a target. TRDY# is tri-stated from the leading edge of PCIRST#. TRDY# remains tri-stated until driven by the SIO as either a master or a slave.
IRDY#	I/O (s/t/s)	**INITIATOR READY**: IRDY# indicates the SIO's ability, as an initiator, to complete the current data phase of the transaction. It is used in conjunction with TRDY#. A data phase is completed on any clock both IRDY# and TRDY# are sampled asserted. During a write, IRDY# indicates the SIO has valid data present on AD[31:0]. During a read, it indicates the SIO is prepared to latch data. IRDY# is an input to the SIO when the SIO is the target and an output when the SIO is an initiator. IRDY# is tri-stated from the leading edge of PCIRST#. IRDY# remains tri-stated until driven by the SIO as either a master or a slave.
STOP#	I/O (s/t/s)	**STOP**: STOP# indicates that the SIO, as a target, is requesting a master to stop the current transaction. As a master, STOP# causes the SIO to stop the current transaction. STOP# is an output when the SIO is a target and an input when the SIO is an initiator. STOP# is tri-stated from the leading edge of PCIRST#. STOP# remains tri-stated until driven by he SIO as either a master or a slave.
LOCK#	I	**LOCK**: LOCK# indicates an atomic operation that may require multiple transactions to complete. LOCK# is always an input to the SIO. When the SIO is the target of a transaction and samples LOCK# negated during the address phase of a transaction, the SIO considers itself a locked resource until it samples LOCK# and FRAME# negated. When other masters attempt accesses while the SIO is locked, the SIO responds with a retry termination.

3.1 PCI Bus Interface Signals (Continued)

Signal Name	Type	Description
IDSEL	I	**INITIALIZATION DEVICE SELECT**: IDSEL is used as a chip select during configuration read and write transactions. The SIO samples IDSEL during the address phase of a transaction. If IDSEL is sampled active, and the bus command is a configuration read or write, the SIO responds by asserting DEVSEL# on the next cycle.
DEVSEL#	I/O (s/t/s)	**DEVICE SELECT**: The SIO asserts DEVSEL# to claim a PCI transaction through positive or subtractive decoding. As an output, the SIO asserts DEVSEL# when it samples IDSEL active in configuration cycles to SIO configuration registers. The SIO also asserts DEVSEL# when an internal SIO address is decoded or when the SIO subtractively decodes a cycle. As an input, DEVSEL# indicates the response to a SIO master-initiated transaction. The SIO also samples this signal for all PCI transactions to decide to subtractively decode the cycle. DEVSEL# is tri-stated from the leading edge of PCIRST#. DEVSEL# remains tri-stated until driven by the SIO as either a master or a slave.
PAR	O	**CALCULATED PARITY SIGNAL**: PAR is "even" parity and is calculated on 36 bits - AD[31:0] plus C/BE[3:0]#. "Even" parity means that the number of "1"s within the 36 bits plus PAR are counted and the sum is always even. PAR is always calculated on 36 bits regardless of the valid byte enables. PAR is generated for address and data phases and is only guaranteed to be valid one PCI clock after the corresponding address or data phase. PAR is driven and tri-stated identically to the AD[31:0] lines except that PAR is delayed by exactly one PCI clock. PAR is an output during the address phase (delayed one clock) for all SIO master transactions. It is also an output during the data phase (delayed one clock) when the SIO is the master of a PCI write transaction, and when it is the target of a read transaction. PAR is always driven low by the SIO synchronously from the leading edge of PCIRST#. The SIO always tri-states PAR from the trailing edge of PCIRST#. If the internal arbiter is enabled (CPUREQ# sampled high on the trailing edge of PCIRST#), the SIO drives PAR low again (synchronously 2-5 PCICLKs later) until the bus is given to another master. If the internal arbiter is disabled (CPUREQ# sampled low on the trailing edge of PCIRST#), PAR remains tri-stated until the SIO is required to drive them valid as a master or slave. When the internal arbiter is enabled, the SIO acts as the central resource responsible for driving PAR when no device is granted the PCI Bus and the bus is idle. When the internal arbiter is disabled, the SIO does not drive PAR as the central resource. The SIO is always responsible for driving PAR when it is granted the bus (SIOGNT# and idle bus) and as appropriate when it is the master of a transaction.
SERR#	I	**SYSTEM ERROR**: SERR# can be pulsed active by any PCI device that detects a system error condition. Upon sampling SERR# active, the SIO generates a non-maskable interrupt (NMI) to the CPU.

3.2 PCI Arbiter Signals

Signal Name	Type	Description
CPUREQ#	I	**CPU REQUEST**: This signal provides the following functions: 1. If CPUREQ# is sampled high on the trailing edge of PCIRST#, the internal arbiter is enabled. If CPUREQ# is sampled low on the trailing edge of PCIRST#, the internal arbiter is disabled. This requires that the host bridge drive CPUREQ# high during PCIRST#. 2. If the SIO's internal arbiter is enabled, this pin is configured as CPUREQ#. An active low assertion indicates that the CPU initiator desires the use of the PCI Bus. If the internal arbiter is disabled, this pin is meaningless after reset.
REQ0#/ SIOGNT#	I	**REQUEST 0/SIO GRANT**: If the SIO's internal arbiter is enabled, this pin is configured as REQ0#. An active low assertion indicates that Initiator0 desires the use of the PCI Bus. If the internal arbiter is disabled, this pin is configured as SIOGNT#. When asserted, SIOGNT# indicates that the external PCI arbiter has granted use of the bus to the SIO.
REQ1#	I	**REQUEST 1**: If the SIO's internal arbiter is enabled through the Arbiter Configuration Register, then this signal is configured as REQ1#. An active low assertion indicates that Initiator1 desires the use of the PCI Bus. If the internal arbiter is disabled, the SIO ignores REQ1# after reset.
CPUGNT#	t/s/o	**CPU GRANT**: If the SIO's internal arbiter is enabled, this pin is configured as CPUGNT#. The SIO's internal arbiter asserts CPUGNT# to indicate that the CPU initiator has been granted the PCI Bus. If the internal arbiter is disabled, this signal is meaningless. CPUGNT# is tri-stated from the leading edge of PCIRST#. CPUGNT# is tri-stated until driven by the SIO CPUGNT# is driven based on the arbitration scheme and the asserted REQx#'s.
GNT0#/ SIOREQ#	t/s/o	**GRANT 0/SIO REQUEST**: If the SIO's internal arbiter is enabled, this pin is configured as GNT0#. The SIO's internal arbiter asserts GNT0# to indicate that Initiator0 has been granted the PCI Bus. If the internal arbiter is disabled, this pin is configured as SIOREQ#. The SIO asserts SIOREQ# to request the PCI Bus. GNT0#/SIOREQ# is tri-stated until driven by the SIO. GNT0#/SIOREQ# is driven based on the arbitration scheme and the asserted REQx#'s.
GNT1#/ RESUME#	t/s/o	**GRANT 1/RESUME**: If the SIO's internal arbiter is enabled, this pin is configured as GNT1#. The SIO's internal arbiter asserts GNT1# to indicate that Initiator1 has been granted the PCI Bus. If the internal arbiter is disabled, this pin is configured as RESUME#. The SIO asserts RESUME# to indicate that the conditions causing the SIO to retry the cycle has passed. GNT1#/RESUME# is tri-stated until driven by the SIO. GNT1#/RESUME# is driven based on the arbitration scheme and the asserted REQx#'s.

3.2 PCI Arbiter Signals (Continued)

Signal Name	Type	Description
MEMREQ#	t/s/o	**MEMORY REQUEST**: If the SIO is configured in Guaranteed Access Time (GAT) Mode, MEMREQ# will be asserted when an ISA master or DMA is requesting the ISA Bus (along with FLSHREQ#) to indicate that the SIO requires ownership of the main memory. MEMREQ# is tri-stated from the leading edge of PCIRST#. MEMREQ# remains tri-stated until driven by the SIO. After PCIRST, MEMREQ# is driven inactive asynchronously from PCIRST# inactive. The SIO asserts FLSHREQ# concurrently with asserting MEMREQ#. **FLSHREQ# MEMREQ# Meaning** 1 1 Idle 0 1 Flush buffers pointing towards PCI to avoid ISA deadlock 1 0 Reserved 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#	t/s/o	**FLUSH REQUEST**: FLSHREQ# is generated by the SIO to command all of the system's posted write buffers pointing towards the PCI Bus to be flushed. This is required before granting the ISA Bus to an ISA master or the DMA. FLSHREQ# is tri-stated from the leading edge of PCIRST#. FLSHREQ# remains tri-stated until driven by the SIO. After PCIRST, FLSHREQ# is driven inactive asynchronously from PCIRST# inactive.
MEMACK#	I	**MEMORY ACKNOWLEDGE**: MEMACK# is the response handshake that indicates to the SIO that the function requested over the MEMREQ# and/or FLSHREQ# signals has been completed. In GAT mode (MEMREQ# and FLSHREQ# asserted), the main memory bus is dedicated to the PCI Bus and the system's posted write buffers pointing towards the PCI Bus have been flushed and are disabled. In non-GAT mode (FLSHREQ# asserted alone), this means the system's posted write buffers have been flushed and are disabled. In either case the SIO can now grant the ISA Bus to the requester.

3.3 Address Decoder Signals

Signal Name	Type	Description
MEMCS#	O	**MEMORY CHIP SELECT**: MEMCS# is a programmable address decode signal provided to a Host CPU bridge. A CPU bridge can use MEMCS# to forward a PCI cycle to main memory behind the bridge. MEMCS# is driven one PCI clock after FRAME# is sampled active (address phase) and is valid for one clock cycle before going inactive. MEMCS# is high upon reset.

3.4 ISA Interface Signals

Signal Name	Type	Description
AEN	O	**ADDRESS ENABLE**: AEN is asserted during DMA cycles to prevent I/O slaves from misinterpreting DMA cycles as valid I/O cycles. When negated, AEN indicates that an I/O slave may respond to address and I/O commands. When asserted, AEN informs I/O resources on the ISA Bus that a DMA transfer is occurring. This signal is also driven high during refresh cycles. AEN is driven low upon reset.
BALE	O	**BUS ADDRESS LATCH ENABLE**: BALE is an active high signal asserted by the SIO to indicate that the address (SA[19:0], LA[23:17]), AEN and SBHE# signal lines are valid. The LA[23:17] address lines are latched on the trailing edge of BALE. BALE remains asserted throughout DMA and ISA master cycles. BALE is driven low upon reset.
SYSCLK	O	**SYSTEM CLOCK**: SYSCLK is an output of the SIO component. The frequencies supported are 6 to 8.33 Mhz.
IOCHRDY	I/O	**I/O CHANNEL READY**: Resources on the ISA Bus assert IOCHRDY to indicate that additional time (wait states) is required to complete the cycle. This signal is normally high on the ISA Bus. IOCHRDY is an input when the SIO owns the ISA Bus and a PCI agent is accessing an ISA slave or during compatible DMA transfers (compatible cycles only). IOCHRDY is output when an external ISA Bus Master owns the ISA Bus and is accessing a PCI slave or an SIO register. As an SIO output, IOCHRDY is driven inactive (low) from the falling edge of the ISA commands. After data is available for an ISA master read or the SIO latches the data for a write cycle, IOCHRDY is asserted for 70 ns. After 70 ns, the SIO floats IOCHRDY. The 70 ns includes both the drive time and the time it takes the SIO to float IOCHRDY. The SIO does not drive this signal when an ISA Bus master is accessing an ISA Bus slave. IOCHRDY is tri-stated upon reset.
IOCS16#	I	**16-BIT I/O CHIP SELECT**: This signal is driven by I/O devices on the ISA Bus to indicate that they support 16-bit I/O bus cycles.
IOCHK#	I	**IO CHANNEL CHECK**: IOCHK# can be driven by any resource on the ISA Bus. When asserted, it indicates that a parity or an un-correctable error has occurred for a device or memory on the ISA Bus. A NMI will be generated to the CPU if the NMI generation is enabled.
IOR#	I/O	**I/O READ**: IOR# is the command to an ISA I/O slave device that the slave may drive data on to the ISA data bus (SD[15:0]). The I/O slave device must hold the data valid until after IOR# is negated. IOR# is an output when the SIO owns the ISA Bus. IOR# is an input when an external ISA master owns the ISA Bus. IOR# is driven high upon reset.
IOW#	I/O	**I/O WRITE**: IOW# is the command to an ISA I/O slave device that the slave may latch data from the ISA data bus (SD[15:0]). IOW# is an output when the SIO owns the ISA Bus. IOW# is an input when an external ISA master owns the ISA Bus. IOW# is driven high upon reset.
LA[23:17]	I/O	**UNLATCHED ADDRESS**: The LA[23:17] address lines are bi-directional. These address lines allow accesses to physical memory on the ISA Bus up to 16 MBytes. LA[23:17] are outputs when the SIO owns the ISA Bus. The LA[23:17] lines become inputs whenever an ISA master owns the ISA Bus. These signals are undefined during DMA type A, B, and F cycles. The LA[23:17] signals are at an unknown state upon reset.
SA[19:0]	I/O	**SYSTEM ADDRESS BUS**: These bi-directional address lines define the selection with the granularity of one byte within the one Megabyte section of memory defined by the LA[23:17] address lines. The address lines SA[19:17] that are coincident with LA[19:17] are defined to have the same values as LA[19:17] for all memory cycles. For I/O accesses, only SA[15:0] are used. SA[19:0] are outputs when the SIO owns the ISA Bus. SA[19:0] are inputs when an external ISA Master owns the ISA Bus. SA[19:0] are undefined during DMA type A, B, or F cycles. SA[19:0] are at an unknown state upon reset.

3.4 ISA Interface Signals (Continued)

Signal Name	Type	Description
SBHE#	I/O	**SYSTEM BYTE HIGH ENABLE**: SBHE# indicates, when asserted, that a byte is being transferred on the upper byte (SD[15:8]) of the data bus. SBHE# is negated during refresh cycles. SBHE# is an output when the SIO owns the ISA Bus. SBHE# is an input when an external ISA master owns the ISA Bus. SBHE# is at an unknown state upon reset.
MEMCS16#	OD	**MEMORY CHIP SELECT 16**: MEMCS16# is a decode of LA[23:17] without any qualification of the command signal lines. ISA slaves that are 16-bit memory devices drive this signal low. The SIO ignores MEMCS16# during I/O access cycles and refresh cycles. During DMA cycles, this signal is only used by the byte swap logic. MEMCS16# is an input when the SIO owns the ISA Bus. MEMCS16# is an output when an ISA Bus master owns the ISA Bus. The SIO drives this signal low during ISA master to PCI memory cycles. MEMCS16# is at an unknown state upon RESET.
MASTER#	I	**MASTER**: An ISA Bus master asserts MASTER# to indicate that it has control of the ISA Bus. Before the ISA master can assert MASTER#, it must first sample DACK# active. Once MASTER# is asserted, the ISA master has control of the ISA Bus until it negates MASTER#.
MEMR#	I/O	**MEMORY READ**: MEMR# is the command to a memory slave that it may drive data onto the ISA data bus. MEMR# is an output when the SIO is a master on the ISA Bus. MEMR# is an input when an ISA master, other than the SIO, owns the ISA Bus. This signal is also driven by the SIO during refresh cycles. For compatible timing mode DMA cycles, the SIO, as a master, asserts MEMR# if the address is less than 16 MBytes. This signal is not generated for accesses to addresses greater than 16 MByte. MEMR# is not driven active during DMA type A, B, or F cycles.
MEMW#	I/O	**MEMORY WRITE**: MEMW# is the command to a memory slave that it may latch data from the ISA data bus. MEMW# is an output when the SIO owns the ISA Bus. MEMW# is an input when an ISA master, other than the SIO, owns the ISA Bus. For compatible timing mode DMA cycles, the SIO, as a master, asserts MEMW# if the address is less than 16 MBytes. This signal is not generated for accesses to addresses greater than 16 MByte. MEMW# is not driven active during DMA type A, B, or F cycles.
SMEMW#	O	**SYSTEM MEMORY WRITE**: The SIO asserts SMEMW# to request a memory slave to accept data from the data lines. If the access is below the 1 MByte range (00000000h - 000FFFFFh) during DMA compatible, SIO master, or ISA master cycles, the SIO asserts SMEMW#. SMEMW# is a delayed version of MEMW#. SMEMW# is driven high upon reset.
SMEMR#	O	**SYSTEM MEMORY READ**: The SIO asserts SMEMR# to request a memory slave to accept data from the data lines. If the access is below the 1 MByte range (00000000h - 000FFFFFh) during DMA compatible, SIO master, or ISA master cycles, the SIO asserts SMEMR#. SMEMR# is a delay version of MEMR#. Upon PCIRST# this signal is low. SMEMR# is driven high upon reset.

3.4 ISA Interface Signals (Continued)

Signal Name	Type	Description
ZEROWS#	I	**ZERO WAIT STATES**: An ISA slave asserts ZEROWS# after its address and command signals have been decoded to indicate that the current cycle can be shortened. A 16-bit ISA memory cycle can be reduced to two SYSCLKs. An 8-bit memory or I/O cycle can be reduced to three SYSCLKs. ZEROWS# has no effect during 16-bit I/O cycles. If IOCHRDY and ZEROWS# are both asserted during the same clock, then ZEROWS# is ignored and wait states are added as a function of IOCHRDY (i.e., IOCHRDY has precedence over ZEROWS#).
OSC	I	**OSCILLATOR**: OSC is the 14.31818 MHz ISA clock signal. It is used by the internal 8254 Timer, counters 0, 1, and 2.
RSTDRV	O	**RESET DRIVE**: The SIO asserts RSTDRV to reset devices that reside on the ISA Bus. The SIO asserts this signal when PCIRST# (PCI Reset) is asserted. In addition, the SIO can be programmed to assert RSTDRV by writing to the ISA Clock Divisor Register. Software should assert the RSTDRV during configuration to reset the ISA Bus when changing the clock divisor. Note that when RSTDRV is generated via the ISA Clock Divisor Register, software must ensure that RSTDRV is driven active for a minimum of 1 μs.
SD[15:0]	I/O	**SYSTEM DATA**: SD[15:0] provide the 16-bit data path for devices residing on the ISA Bus. SD[15:8] correspond to the high order byte and SD[7:0] correspond to the low order byte. SD[15:0] are undefined during refresh. The SIO tri-states SD[15:0] during reset.

3.5 DMA Signals

Signal Name	Type	Description
DREQ [3:0, 7:5]	I	**DMA REQUEST**: The DREQ lines are used to request DMA service from the SIO's DMA controller or for a 16-bit master to gain control of the ISA expansion bus. The active level (high or low) is programmed via the DMA Command Register (bit 6). When the bit 6 = 0, DREQ[3:0,7:5] are active high and when bit 6 = 1, the signals are active low. All inactive to active edges of DREQ are assumed to be asynchronous. The request must remain active until the appropriate DACK signal is asserted.
DACK# [3:0, 7:5]	O	**DMA ACKNOWLEDGE**: The DACK output lines indicate that a request for DMA service has been granted by the SIO or that a 16-bit master has been granted the bus. The active level (high or low) is programmed via the DMA Command Register (bit 7). When bit 7 = 0, DACK# [3:0,7:5] are active low and when bit 7 = 1, the signals are active high. These lines should be used to decode the DMA slave device with the IOR# or IOW# line to indicate selection. If used to signal acceptance of a bus master request, this signal indicates when it is legal to assert MASTER#. If the DMA controller has been programmed for a timing mode other than compatible mode, and another device has requested the bus, and a 4 μsec time has elapsed, this line will be negated and the transfer stopped before the transfer is complete. In this case, the transfer is re-started at the next arbitration period that the channel wins the bus. Upon PCIRST#, these lines are set inactive (high).
EOP	I/O	**END OF PROCESS**: EOP is bi-directional, acting in one of two modes, and is directly connected to the TC line of the ISA Bus. DMA slaves assert EOP to the SIO to terminate DMA cycles. The SIO asserts EOP to DMA slaves as a terminal count indicator. **EOP-In Mode** For all transfer types during DMA, the SIO samples EOP. If it is sampled asserted, the transfer is terminated. **TC-Out Mode** The SIO asserts EOP after a new address has been output, if the byte count expires with that transfer. The EOP (TC) remains asserted until AEN is negated, unless AEN is negated during an autoinitialization. EOP (TC) is negated before AEN is negated during an autoinitialization. When all the DMA channels are not in use, the EOP signal is in output mode and negated (low). After PCIRST#, EOP is in output mode and inactive.
REFRESH#	I/O	**REFRESH**: As an output, REFRESH# is used by the SIO to indicate when a refresh cycle is in progress. It should be used to enable the SA[15:0] address to the row address inputs of all banks of dynamic memory on the ISA Bus. Thus, when MEMR# is asserted, the entire expansion bus dynamic memory is refreshed. Memory slaves must not drive any data onto the bus during refresh. As an output, this signal is driven directly onto the ISA Bus. This signal is an output only when the SIO DMA refresh is a master on the bus responding to an internally generated request for refresh. As an input, REFRESH# is driven by 16-bit ISA Bus masters to initiate refresh cycles. Upon PCIRST#, this signal is tri-stated.

3.6 Timer Signals

Signal Name	Type	Description
SPKR	O	**SPEAKER DRIVE**: The SPKR signal is the output of counter 2 and is "ANDed" with Port 061h bit 1 to provide Speaker Data Enable. This signal drives an external speaker driver device, which in turn drives the ISA system speaker. SPKR has a 24 ma drive capability. Upon reset, its output state is 0.

3.7 Interrupt Controller Signals

Signal Name	Type	Description
IRQ[15,14, 11:9, 7:3,1]	I	**INTERRUPT REQUEST**: The IRQ signals provide both system board components and ISA Bus I/O devices with a mechanism for asynchronously interrupting the CPU. The assertion mode of these inputs depends on the programming of LTIM, bit-3 of ICW1 on both Controller-1 and Controller-2. When LTIM is programmed to a 0, a low-to-high transition on any of that controller's IRQ lines is recognized as an interrupt request. This is "edge-triggered" mode. Edge-triggered mode is the SIO default. When LTIM is programmed to a 1, a high level on any of that controller's IRQ lines is recognized as an interrupt request. This mode is "level-triggered" mode. Upon PCIRST#, the IRQ lines are placed in edge-triggered mode. An active IRQ input must remain asserted until after the interrupt is acknowledged. If the input goes inactive before this time, a DEFAULT IRQ7 occurs when the CPU acknowledges the interrupt. **Note:** Refer to the Utility Bus Signal descriptions for IRQ12 and IRQ13 signal descriptions.
IRQ8#	I	**INTERRUPT REQUEST EIGHT SIGNAL**: IRQ8# is an active low interrupt input. The assertion mode of this input depends on the programming of the LTIM bit of ICW1 on both Controller-1 and Controller-2. When the LTIM = 0, a high-to-low transition on IRQ8# is recognized as an interrupt request. This is "edge-triggered" mode. Edge triggered mode is the SIO default. When the LTIM = 1, a low level on IRQ8# is recognized as an interrupt request. This mode is "level-triggered" mode. Upon PCIRST#, IRQ8# will be placed in edge- triggered mode. IRQ8# must remain asserted until after the interrupt is acknowledged. If the input goes inactive before this time, a DEFAULT IRQ7 will occur when the CPU acknowledges the interrupt.
INT	O	**CPU INTERRUPT**: INT is driven by the SIO to signal the CPU that an interrupt request is pending and needs to be serviced. It is asynchronous with respect to SYSCLK or PCICLK and is always an output. The interrupt controller must be programmed following a reset to ensure that INT is at a known state. Upon PCIRST#, INT is driven low.
NMI	O	**NON-MASKABLE INTERRUPT**: NMI is used to force a non-maskable interrupt to the CPU. The SIO generates an NMI when either SERR# or IOCHK# is asserted, depending on how the NMI Status and Control Register is programmed. The CPU detects an NMI when it detects a rising edge on NMI. After the NMI interrupt routine processes the interrupt, the NMI status bits in the NMI Status and Control Register are cleared by software. The NMI interrupt routine must read this register to determine the source of the interrupt. The NMI is reset by setting the corresponding NMI source enable/disable bit in the NMI Status and Control Register. To enable NMI interrupts, the two NMI enable/disable bits in the register must be set to 0, and the NMI mask bit in the NMI Enable/Disable and Real-Time Clock Address Register must be set to 0. Upon PCIRST#, this signal is driven low.

3.8 Utility Bus Signals

Signal	Type	Description
UBUSTR	O	**UTILITY DATA BUS TRANSMIT/RECEIVE**: UBUSTR is tied directly to the direction control of a 74F245 that buffers the utility data bus, UD[7:0]. UBUSTR is asserted for all I/O read cycles (regardless if a Utility Bus device has been decoded). UBUSTR is asserted for memory cycles only if BIOS space has been decoded. For PCI and ISA master-initiated read cycles, UBUSTR is asserted from the falling edge of either IOR# or MEMR#, depending on the cycle type (driven from MEMR# only if BIOS space has been decoded). When the rising edge of IOR# or MEMR# occurs, the SIO negates UBUSTR. For DMA read cycles from the Utility Bus, UBUSTR is asserted when DACKx# is asserted and negated when DACKx# is negated. At all other times, UBUSTR is negated. Upon PCIRST#, this signal is driven low.
UBUSOE#	O	**UTILITY DATA BUS OUTPUT ENABLE**: UBUSOE# is tied directly to the output enable of a 74F245 that buffers the utility data bus, UD[7:0], from the system data bus, SD[7:0]. UBUSOE# is asserted anytime a SIO supported Utility Bus device is decoded, and the devices decode is enabled in the Utility Bus Chip Select Enable Registers. UBUSOE# is asserted from the falling edge of the ISA commands (IOR#, IOW#, MEMR#, or MEMW#) for PCI and ISA master-initiated cycles. UBUSOE# is negated from the rising edge of the ISA command signals for SIO-initiated cycles and the SA[16:0] and LA[23:17] address for ISA master-initiated cycles. For DMA cycles, UBUSOE# is asserted when DACK2# is asserted and negated when DACK2# negated. UBUSOE# is not driven active under the following conditions: 1. during an I/O access to the floppy controller, if DSKCHG is sampled low at reset. 2. if the Digital Output Register is programmed to ignore DACK2#. 3. during an I/O read access to floppy location 3F7h (primary) or 377h (secondary), if the IDE decode space is disabled (i.e. IDE is not resident on the Utility Bus). 4. during any access to a utility bus peripheral in which its decode space has been disabled. Upon a PCIRST#, this signal is driven inactive (high).
ECSADDR [2:0]	O	**ENCODED CHIP SELECTS**: ECSADDR[2:0] are the encoded chip selects and/or control signals for the Utility Bus peripherals supported by the SIO. The binary code formed by the three signals indicates which Utility Bus device is selected. These signals tie to the address inputs of two external 74F138 decoder chips and are driven valid/invalid from the SA[16:0] and LA[23:17] address lines. Upon PCIRST#, these signals are driven high.
ECSEN#	O	**ENCODED CHIP SELECT ENABLE**: ECSEN# is used to determine which of the two external 74F138 decoders is to be selected. ECSEN# is driven low to select decoder 1 and driven high to select decoder 2. This signal is driven valid/invalid from the SA[16:0] and LA[23:17] address lines (except for the generation of RTCALE#, in which case, ECSEN# is driven active based on IOW# falling, and remains active for two SYSCLKs). During a non-valid address or during an access not targeted for the Utility Bus, this signal is driven high. Upon PCIRST#, this signal is driven high.
ALT_RST#	O	**ALTERNATE RESET**: ALT_RST# is used to reset the CPU under program control. This signal is AND'ed together externally with the reset signal (KBDRST#) from the keyboard controller to provide a software means of resetting the CPU. This provides a faster means of reset than is provided by the keyboard controller. Writing a 1 to bit 0 in the Port 92 Register causes this signal to pulse low for approximately 4 SYSCLKs. Before another ALT_RST# pulse can be generated, bit 0 must be set to 0. Upon PCIRST#, this signal is driven inactive high (bit 0 in the Port 92 Register is set to 0).

3.8 Utility Bus Signals (Continued)

<table>
<tr><th>Signal</th><th>Type</th><th>Description</th></tr>
<tr><td>ALT_A20</td><td>O</td><td>ALTERNATE A20. ALT_A20 is used to force A20M# to the CPU low for support of real mode compatible software. This signal is externally OR'ed with the A20GATE signal from the keyboard controller and CPURST to control the A20M# input of the CPU. Writing a 0 to bit 1 of the Port 92 Register forces ALT_A20 low. ALT_A20 low drives A20M# to the CPU low, if A20GATE from the keyboard controller is also low. Writing a 1 to bit 1 of the Port 92 Register force ALT_A20 high. ALT_A20 high drives A20M# to the CPU high, regardless of the state of A20GATE from the keyboard controller. Upon reset, this signal is driven low.</td></tr>
<tr><td>DSKCHG</td><td>I</td><td>DISK CHANGE: DSKCHG is tied directly to the DSKCHG signal of the floppy controller. This signal is inverted and driven on SD7 during I/O read cycles to floppy address locations 3F7h (primary) or 377h (secondary) as shown in the table below. Note that the primary and secondary locations are programmed in the Utility Bus Address Decode Enable/Disable Register "A".
<table>
<tr><th>FLOPPYCS# Decode</th><th>IDECSx# Decode</th><th>State of SD7 (Output)</th><th>State of UBUSOE#</th></tr>
<tr><td>Enabled</td><td>Enabled</td><td>Tri-stated</td><td>Enabled</td></tr>
<tr><td>Enabled</td><td>Disabled</td><td>Driven via DSKCHG</td><td>Disabled</td></tr>
<tr><td>Disabled</td><td>Enabled</td><td>Tri-stated</td><td>Disabled (note)</td></tr>
<tr><td>Disabled</td><td>Disabled</td><td>Tri-stated</td><td>Disabled</td></tr>
</table>
NOTE:
This mode is not supported because of potential contention between the Utility Bus buffer and a floppy on the ISA Bus driving the system bus at the same time during shared I/O accesses.
This signal is also used to determine if the floppy controller is present on the Utility Bus. It is sampled on the trailing edge of PCIRST#, and if high, the Floppy is present. For systems that do not support a Floppy via the SIO, this pin should be strapped low. If sampled low, the SD7 function, UBUSOE#, and ECSADDR[2:0] signals will not be enabled for DMA or programmed I/O accesses to the floppy disk controller. This condition overrides the floppy decode enable bits in the Utility Bus Chip Select A.</td></tr>
<tr><td>FERR#/ IRQ13</td><td>I</td><td>NUMERIC COPROCESSOR ERROR/IRQ13: This signal has two separate functions, depending on bit 5 in the ISA Clock Divisor Register. This pin functions as a FERR# signal supporting coprocessor errors, if this function is enabled (bit 5 = 1), or as an external IRQ13, if the coprocessor error function is disabled (bit 5 = 0).
If programmed to support coprocessor error reporting, this signal is tied to the coprocessor error signal on the CPU. If FERR# is asserted by the coprocessor inside the CPU, the SIO generates an internal IRQ13 to its interrupt controller unit. The SIO then asserts the INT output to the CPU. Also, in this mode, FERR# gates the IGNNE# signal to ensure that IGNNE# is not asserted to the CPU unless FERR# is active. When FERR# is asserted, the SIO asserts INT to the CPU as an IRQ13. IRQ13 continues to be asserted until a write to F0h has been detected.
If the Coprocessor error reporting is disabled, FERR# can be used by the system as IRQ13. Upon PCIRST#, this signal provides the standard IRQ13 function.
This signal should be pulled high with an external 8.2K Ω pull-up resistor if the IRQ13 mode is used or the pin is left floating.</td></tr>
</table>

3.8 Utility Bus Signals (Continued)

Signal	Type	Description
IGNNE#	O	**IGNORE ERROR**: This signal is connected to the ignore error pin of the CPU. IGNNE# is only used if the SIO coprocessor error reporting function is enabled in the ISA Clock Divisor Register (bit 5 = 1). If FERR# is active, indicating a coprocessor error, a write to the Coprocessor Error Register (F0h) causes the IGNNE# to be asserted. IGNNE# remains asserted until FERR# is negated. If FERR# is not asserted when the Coprocessor Error Register is written, the IGNNE# is not asserted. IGNNE# is driven high upon a reset.
IRQ12/M	I	**INTERRUPT REQUEST/MOUSE INTERRUPT**: In addition to providing the standard interrupt function as described in the pin description for IRQ[15,14, 11:9, 7:3, 1], this pin also provides a mouse interrupt function. Bit 4 in the ISA Clock Divisor Register determines the functionality of IRQ12/M. When bit 4 = 0, the standard interrupt function is provided and this pin can be tied to the ISA connector. When bit 4 = 1, the mouse interrupt function is provided and this pin can be tied to the DIRQ12 output of the keyboard controller. When the mouse interrupt function is selected, a low to high transition on this signal is latched by the SIO and an INT is generated to the CPU as IRQ12. An interrupt will continue to be generated until a PCIRST# or an I/O read access to address 60h (falling edge of IOR#) is detected. After a PCIRST#, this pin provides the standard IRQ12 function.

3.9 Test Signals

Pin Name	Type	Description
TEST	I	**TEST**: The TEST signal is used to tri-state all of the SIO outputs. During normal operation, this input should be tied to ground.
TESTO	O	**TEST OUTPUT:** This is the output pin used during NAND tree testing.

4.0 REGISTER DESCRIPTION

The SIO contains both PCI configuration registers and non-configuration registers. The configuration registers (Table 4-1) are located in PCI configuration space and are only accessible from the PCI Bus. Addresses for configuration registers are offset values that appear on AD[7:2] and C/BE#[3:0]. The configuration registers (Section 4-1) 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 non-configuration registers (Table 4-2) include DMA Registers (Section 4-2), Timer Registers (Section 4-3), Interrupt Controller Registers (Section 4-4), and Non-Maskable Interrupt and Utility Bus Support Registers (Section 4-5). All of these registers are accessible from the PCI Bus. In addition, some of the registers are accessible from the ISA Bus. Table 4-2 indicates the bus access for each register. Except for the DMA scatter/gather registers and the BIOS timer registers, the non-configuration registers can only be accessed as byte quantities. If a PCI master attempts a multi-byte access (i.e., more than one Byte Enable signal asserted), the SIO responds with a target-abort. The scatter/gather registers and BIOS timer registers can be accessed as byte, word, or dword quantities.

Some of the SIO configuration and non-configuration registers contain reserved bits. These bits are labeled "Reserved". Software must take care to deal correctly with bit-encoded fields that are reserved.

In addition to reserved bits within a register, the SIO contains address locations in the PCI configuration space that are marked "Reserved" (Table 4-1). The SIO responds to accesses to these address locations by completing the PCI cycle. However, reads of reserved address locations yield all zeroes and writes have no affect on the SIO.

The SIO, upon receiving a hard reset (PCIRST# signal), sets its internal registers to pre-determined default states. The default values are indicated in the individual register descriptions.

Table 4-1. Configuration Registers

Configuration Offset	Register	Register Access	Bus Access
00h-01h	Vendor Identification	RO	PCI Only
02h-03h	Device Identification	RO	PCI Only
04h-05h	Command	R/W	PCI Only
06h-07h	Device Status	R/W	PCI Only
08h	Revision Identification	RO	PCI Only
09h-3Fh	Reserved	–	PCI Only
40h	PCI Control	R/W	PCI Only
41h	PCI Arbiter Control	R/W	PCI Only
42h	PCI Arbiter Priority Control	R/W	PCI Only
43h	Reserved	–	PCI Only
44h	MEMCS# Control	R/W	PCI Only
45h	MEMCS# Bottom of Hole	R/W	PCI Only
46h	MEMCS# Top of Hole	R/W	PCI Only
47h	MEMCS# Top of Memory	R/W	PCI Only
48h	ISA Address Decoder Control	R/W	PCI Only
49h	ISA Address Decoder ROM Block Enable	R/W	PCI Only
4Ah	ISA Address Decoder Bottom of Hole	R/W	PCI Only
4Bh	ISA Address Decoder Top of Hole	R/W	PCI Only
4Ch	ISA Controller Recovery Timer	R/W	PCI Only
4Dh	ISA Clock Divisor	R/W	PCI Only
4Eh	Utility Bus Chip Select Enable A	R/W	PCI Only
4Fh	Utility Bus Chip Select Enable B	R/W	PCI Only
50h-53h	Reserved	–	PCI Only
54h	MEMCS# Attribute Register #1	R/W	PCI Only
55h	MEMCS# Attribute Register #2	R/W	PCI Only
56h	MEMCS# Attribute Register #3	R/W	PCI Only
57h	Scatter/Gather Relocation Base Address	R/W	PCI Only
58h-7Fh	Reserved	–	PCI Only
80h-81h	BIOS Timer Base Address	R/W	PCI Only
82h-FFh	Reserved	–	PCI Only

Table 4-2. Non-Configuration Registers

Address	Function Unit	Register	Register Access	Bus Access
0000h	DMA	DMA1 CH0 Base and Current Address	R/W	PCI Only
0001h	DMA	DMA1 CH0 Base and Current Count	R/W	PCI Only
0002h	DMA	DMA1 CH1 Base and Current Address	R/W	PCI Only
0003h	DMA	DMA1 CH1 Base and Current Count	R/W	PCI Only
0004h	DMA	DMA1 CH2 Base and Current Address	R/W	PCI Only
0005h	DMA	DMA1 CH2 Base and Current Count	R/W	PCI Only
0006h	DMA	DMA1 CH3 Base and Current Address	R/W	PCI Only
0007h	DMA	DMA1 CH3 Base and Current Count	R/W	PCI Only
0008h	DMA	DMA1 Status(r) Command(w)	R/W	PCI Only
0009h	DMA	DMA1 Write Request	WO	PCI Only
000Ah	DMA	DMA1 Write Single Mask Bit	WO	PCI Only
000Bh	DMA	DMA1 Write Mode	WO	PCI Only
000Ch	DMA	DMA1 Clear Byte Pointer	WO	PCI Only
000Dh	DMA	DMA1 Master Clear	WO	PCI Only
000Eh	DMA	DMA1 Clear Mask	WO	PCI Only
000Fh	DMA	DMA1 Read/Write All Mask Register Bits	R/W	PCI Only
0020h	Interrupt	INT 1 Control	R/W	PCI /ISA
0021h	Interrupt	INT 1 Mask	R/W	PCI /ISA
0040h	Timer	Timer Counter 1 - Counter 0 Count	R/W	PCI /ISA
0041h	Timer	Timer Counter 1 - Counter 1 Count	R/W	PCI /ISA
0042h	Timer	Timer Counter 1 - Counter 2 Count	R/W	PCI /ISA
0043h	Timer	Timer Counter 1 Command Mode	WO	PCI /ISA
0060h[2]	Control	Reset UBus IRQ12	RO	PCI /ISA
0061h	Control	NMI Status and Control	R/W	PCI /ISA
0070h[2]	Control	CMOS RAM Address and NMI Mask	WO	PCI /ISA
0078h-007Bh[3,4,5]	Timer	BIOS Timer	R/W	PCI Only
0080h[1]	DMA	DMA Page Register Reserved	R/W	PCI/ISA
0081h	DMA	DMA Channel 2 Page Register	R/W	PCI /ISA
0082h	DMA	DMA Channel 3 Page Register	R/W	PCI /ISA
0083h	DMA	DMA Channel 1 Page Register	R/W	PCI /ISA
0084h[1]	DMA	DMA Page Register Reserved	R/W	PCI/ISA
0085h[1]	DMA	DMA Page Register Reserved	R/W	PCI/ISA
0086h[1]	DMA	DMA Page Register Reserved	R/W	PCI/ISA
0087h	DMA	DMA Channel 0 Page Register	R/W	PCI /ISA
0088h[1]	DMA	DMA Page Register Reserved	R/W	PCI/ISA
0089h	DMA	DMA Channel 6 Page Register	R/W	PCI /ISA

Table 4-2. Non-Configuration Registers (Continued)

Address	Function Unit	Register	Register Access	Bus Access
008Ah	DMA	DMA Channel 7 Page Register	R/W	PCI /ISA
008Bh	DMA	DMA Channel 5 Page Register	R/W	PCI /ISA
008Ch[1]	DMA	DMA Page Register Reserved	R/W	PCI/ISA
008Dh[1]	DMA	DMA Page Register Reserved	R/W	PCI/ISA
008Eh[1]	DMA	DMA Page Register Reserved	R/W	PCI/ISA
008Fh	DMA	DMA Low Page Register Refresh	R/W	PCI/ISA
0090h[1]	DMA	DMA Page Register Reserved	R/W	PCI/ISA
0092h[2]	Control	Port 92 Register	R/W	PCI/ISA
0094h[1]	DMA	DMA Page Register Reserved	R/W	PCI/ISA
0095h[1]	DMA	DMA Page Register Reserved	R/W	PCI/ISA
0096h[1]	DMA	DMA Page Register Reserved	R/W	PCI/ISA
0098h[1]	DMA	DMA Page Register Reserved	R/W	PCI/ISA
009Ch[1]	DMA	DMA Page Register Reserved	R/W	PCI/ISA
009Dh[1]	DMA	DMA Page Register Reserved	R/W	PCI/ISA
009Eh[1]	DMA	DMA Page Register Reserved	R/W	PCI/ISA
009Fh	DMA	DMA Low Page Register Refresh	R/W	PCI/ISA
00A0h	Interrupt	INT 2 Control Register	R/W	PCI /ISA
00A1h	Interrupt	INT 2 Mask Register	R/W	PCI /ISA
00C0h	DMA	DMA2 CH0 Base and Current Address	R/W	PCI Only
00C2h	DMA	DMA2 CH0 Base and Current Count	R/W	PCI Only
00C4h	DMA	DMA2 CH1 Base and Current Address	R/W	PCI Only
00C6h	DMA	DMA2 CH1 Base and Current Count	R/W	PCI Only
00C8h	DMA	DMA2 CH2 Base and Current Address	R/W	PCI Only
00CAh	DMA	DMA2 CH2 Base and Current Count	R/W	PCI Only
00CCh	DMA	DMA2 CH3 Base and Current Address	R/W	PCI Only
00CEh	DMA	DMA2 CH3 Base and Current Count	R/W	PCI Only
00D0h	DMA	DMA2 Status(r) Command(w) Register	R/W	PCI Only
00D2h	DMA	DMA2 Write Request Register	WO	PCI Only
00D4h	DMA	DMA2 Write Single Mask Bit Register	WO	PCI Only
00D6h	DMA	DMA2 Write Mode Register	WO	PCI Only
00D8h	DMA	DMA2 Clear Byte Pointer Register	WO	PCI Only
00DAh	DMA	DMA2 Master Clear Register	WO	PCI Only
00DCh	DMA	DMA2 Clear Mask Register	WO	PCI Only
00DEh	DMA	DMA2 Read/Write All Mask Register Bits	R/W	PCI Only
00F0h[2]	Control	Coprocessor Error Register	WO	PCI /ISA
0372h[2]	Control	Secondary Floppy Disk Digital Output Register	WO	PCI /ISA

Table 4-2. Non-Configuration Registers (Continued)

Address	Function Unit	Register	Register Access	Bus Access
03F2h[2]	Control	Primary Floppy Disk Digital Output Register	WO	PCI /ISA
040Ah[3]	DMA	Scatter/Gather Interrupt Status Register	RO	PCI Only
040Bh	DMA	DMA1 Extended Mode Register	WO	PCI /ISA
0410h[3,4]	DMA	CH0 Scatter/Gather Command	WO	PCI Only
0411h[3,4]	DMA	CH1 Scatter/Gather Command	WO	PCI Only
0412h[3,4]	DMA	CH2 Scatter/Gather Command	WO	PCI Only
0413h[3,4]	DMA	CH3 Scatter/Gather Command	WO	PCI Only
0415h[3,4]	DMA	CH5 Scatter/Gather Command	WO	PCI Only
0416h[3,4]	DMA	CH6 Scatter/Gather Command	WO	PCI Only
0417h[3,4]	DMA	CH7 Scatter/Gather Command	WO	PCI Only
0418h[3,4]	DMA	CH0 Scatter/Gather Status	RO	PCI Only
0419h[3,4]	DMA	CH1 Scatter/Gather Status	RO	PCI Only
041Ah[3,4]	DMA	CH2 Scatter/Gather Status	RO	PCI Only
041Bh[3,4]	DMA	CH3 Scatter/Gather Status	RO	PCI Only
041Dh[3,4]	DMA	CH5 Scatter/Gather Status	RO	PCI Only
041Eh[3,4]	DMA	CH6 Scatter/Gather Status	RO	PCI Only
041Fh[3,4]	DMA	CH7 Scatter/Gather Status	RO	PCI Only
0420h-0423h[3,4]	DMA	CH0 Scatter/Gather Descriptor Table Pointer	R/W	PCI Only
0424h-0427h[3,4]	DMA	CH1 Scatter/Gather Descriptor Table Pointer	R/W	PCI Only
0428h-042Bh[3,4]	DMA	CH2 Scatter/Gather Descriptor Table Pointer	R/W	PCI Only
042Ch-042Fh[3,4]	DMA	CH3 Scatter/Gather Descriptor Table Pointer	R/W	PCI Only
0434h-0437h[3,4]	DMA	CH5 Scatter/Gather Descriptor Table Pointer	R/W	PCI Only
0438h-043Bh[3,4]	DMA	CH6 Scatter/Gather Descriptor Table Pointer	R/W	PCI Only
043Ch-043Fh[3,4]	DMA	CH7 Scatter/Gather Descriptor Table Pointer	R/W	PCI Only
0481h	DMA	DMA CH2 High Page Register	R/W	PCI /ISA
0482h	DMA	DMA CH3 High Page Register	R/W	PCI /ISA
0483h	DMA	DMA CH1 High Page Register	R/W	PCI /ISA
0487h	DMA	DMA CH0 High Page Register	R/W	PCI /ISA
0489h	DMA	DMA CH6 High Page Register	R/W	PCI /ISA
048Ah	DMA	DMA CH7 High Page Register	R/W	PCI /ISA
048Bh	DMA	DMA CH5 High Page Register	R/W	PCI /ISA
04D6h	DMA	DMA2 Extended Mode Register	WO	PCI /ISA

NOTES:
1. PCI write cycles to these address locations flow through to the ISA Bus. PCI read cycles to these address locations do not flow through to the ISA Bus.
2. PCI read and write cycles to these address locations flow through to the ISA Bus.
3. The I/O address of this register is relocatable. The value shown in this table is the default address location.
4. This register can be accessed as a byte, word, or dword quantity.
5. If this register location is enabled, PCI accesses to the BIOS Timer Register do not flow through to the ISA Bus. If disabled, accesses to this address location flow through to the ISA Bus.

4.1 SIO Configuration Register Description

This section describes the SIO configuration registers. These registers include the Mandatory Header Registers (located in the first 64 bytes of configuration space) and the SIO specific registers (located from configuration offset 40h - 56h).

4.1.1 VID - VENDOR IDENTIFICATION REGISTER

Register Name: Vendor Identification Register
Address Offset: 00h, 01h
Default Value: 8086h
Attribute: Read Only
Size: 16 bits

The VID Register contains the vendor identification number. This 16-bit register combined with the Device Identification Register uniquely identifies any PCI device. Writes to this register have no effect.

Table 4-3. Vendor Identification Register

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

4.1.2 DID - DEVICE IDENTIFICATION REGISTER

Register Name: Device Identification Register
Address Offset: 02h, 03h
Default Value: 0484h
Attribute: Read Only
Size: 16 bits

The DID Register contains the device identification number. This register, along with the Vendor ID, uniquely identifies the SIO. Writes to this register have no effect. The DID fields are shown in Table 4-4.

Table 4-4. Device Identification Register

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

4.1.3 COM - COMMAND REGISTER

Register Name: Command Register
Address Offset: 04h - 05h
Default Value: 0007h
Attribute: Read/Write
Size: 16 bits

The SIO does not implement the Command Register defined in the PCI specification. This 16-bit register is shown for completeness.

Table 4-5. Command Register

Bit	Description
15:5	**Reserved.** Read 0.
4	**PMWE (Postable Memory Write Enable).** Enable Postable memory write, memory write and invalidate, and memory read Pre-fetch commands. The SIO does not support these commands as a master or slave so this bit is not implemented. This bit will always be read as a 0.
3	**SCE (Special Cycle Enable).** Since the SIO does not respond to any type of special cycle on the PCI Bus, this bit is not implemented. This bit is hardwired to 0 and will always be read as 0.
2	**BME (Bus Master Enable).** Since the SIO always requests the PCI Bus on behalf of ISA masters, DMA, or line buffer PCI requests, this bit is hardwired to a 1 and will always be read as a 1.
1	**MSE (Memory Space Enable).** Enables SIO to accept a PCI-originated memory cycle. Since the SIO always responds to PCI-originated memory cycles (and ISA-bound cycles) by asserting DEVSEL#, this bit is hardwired to a 1 and will always be read as a 1.
0	**IOSE (I/O Space Enable).** Enable SIO to accept a PCI-originated I/O cycle. Since the SIO always responds to a master I/O cycle, this bit is hardwired to a 1 and will always be read as a 1.

4.1.4 DS - DEVICE STATUS REGISTER

Register Name Device Status Register
Address Offset 06h, 07h
Default Value 0200h
Attribute Read/Write
Size 16 bits

DSR is a 16-bit status register that reports the occurrence of a PCI master-abort by the SIO or a PCI target-abort when the SIO is a master. The register also indicates the SIO DEVSEL# signal timing that is hardwired in the SIO. The DS fields are shown in Table 4-6.

Table 4-6. Device Status Register

Bit	Description
15	**Reserved.** Read as 0.
14	**SERRS (SERR# Status).** This bit is set by the PCI devices that assert the SERR# signal. Since SERR# is only an input to the SIO, this bit is not implemented and will always be read as 0.
13	**MA (Master-Abort Status).** When the SIO, as a master, generates a master-abort, MA is set to a 1. Software sets MA to 0 by writing a 1 to this bit location.
12	**RTA (Received Target-Abort Status).** When the SIO is a master on the PCI Bus and receives a target-abort, this bit is set to a 1. Software sets RTA to 0 by writing a 1 to this bit location.
11	**STA (Signaled Target-Abort Status).** This bit is set to a 1 by the SIO when it generates a target-abort.
10:9	**DEVT (SIO DEVSEL# Timing Status).** This 2-bit field defines the timing for DEVSEL# assertion. These read only bits indicate the SIO's DEVSEL# timing when performing a positive decode. Since the SIO always generates DEVSEL# with medium timing, DEVT = 01. This DEVSEL# timing does not include Configuration cycles.
8:0	**Reserved.** Read as 0's.

4.1.5 RID - REVISION IDENTIFICATION REGISTER

Register Name: Revision Identification Register
Address Offset: 08h
Default Value: xxh
Attribute: Read Only
Size: 8 bits

This 8-bit register contains the revision number for the SIO. These bits are read only and writes to this register have no effect. The RID fields are shown in Table 4-7.

Table 4-7. Revision Identification Register

Bit	Description
7:0	**Revision Identification Number.** This is an 8-bit value that indicates the revision identification number for the SIO.

4.1.6 PCICON - PCI CONTROL REGISTER

Register Name: PCI Control Register
Address Offset: 40h
Default Value: 20h
Attribute: Read/Write
Size: 8 bits

This 8-bit register controls the Line Buffer operation, the SIO's PCI Posted Write Buffer enabling, and the DEVSEL# signal sampling point. The PCICON Register also controls how the SIO responds to INTA cycles on the PCI Bus. The PCICON fields are shown in Table 4-8.

Table 4-8. PCI Control Register

Bit	Description
7:6	**Reserved.** Read as 0.
5	**IAE (Interrupt Acknowledge Enable).** When IAE = 0, the SIO ignores INTA cycles generated on the PCI Bus. However, when disabled, the SIO still responds to accesses to the 8259's register set and allows poll mode functions. When IAE = 1, the SIO responds to INTA cycles in the normal fashion. This bit defaults to a 1 (respond to INTA cycles).
4:3	**SDSP (Subtractive Decoding Sample Point).** The SDSP field determines the DEVSEL# sample point, after which an inactive DEVSEL# results in the SIO forwarding the unclaimed PCI cycle to the ISA Bus (subtractive decoding). This setting should match the slowest device in the system. **Bit 4 3 Operation** 0 0 Slow sample point 0 1 Typical sample point 1 0 Fast sample point 1 1 Reserved
2	**PPBE (PCI Posted Write Buffer Enable).** When PPBE = 0, the PCI posted write buffer is disabled. When PPBE = 1, the PCI posted write buffer is enabled. This bit defaults to disabled mode (PPBE = 0).
1	**ILBC (ISA Master Line Buffer Configuration).** When ILBC = 0, the Line Buffer is in single transaction mode. When ILBC = 1, the Line Buffer is in 8 byte mode. This bit applies only to ISA Master transfers. This bit defaults to single transaction mode (ILBC = 0).
0	**DLBC (DMA Line Buffer Configuration).** When DLBC = 0, the Line Buffer is in single transaction mode. When DLBC = 1, the Line Buffer is in 8-byte mode. This bit applies only to DMA transfers. This bit defaults to single transaction mode (DLBC = 0).

4.1.7 PAC - PCI ARBITER CONTROL REGISTER

Register Name: PCI Arbiter Control
Address Offset: 41h
Default Value: 00h
Attribute: Read/Write
Size: 8 bits

This 8-bit register controls the operation of the PCI arbiter. The PAC register enables/disables the guaranteed access time mode, controls bus lock cycles, enables/disables CPU bus parking, and controls the master retry timer. The PAC fields are shown in Table 4-9.

Table 4-9. PCI Arbiter Control Register

Bit	Description
7:5	**Reserved.** Read as 0's.
4:3	**MRT (Master Retry Timer).** This 2-bit field determines the number of PCICLKs after the first retry that a PCI initiator's Bus request will be unmasked. **Bit 4 3 Operation** 0 0 Timer disabled, Retries never masked. 0 1 Retries unmasked after 16 PCICLK's. 1 0 Retries unmasked after 32 PCICLK's. 1 1 Retries unmasked after 64 PCICLK's.
2	**BP (Bus Park).** Set to a 1 the SIO will park CPUREQ# on the PCI bus when it detects the PCI bus idle. If Bus Park is disabled, the SIO takes responsibility for driving AD, C/BE# and PAR upon detection of bus idle state if the internal arbiter is enabled.
1	**BL (Bus Lock).** This bit selects between bus lock and resource lock. When BO = 1, Bus Lock is selected. The arbiter considers the entire PCI bus locked upon initiation of any locked transaction. When BO = 0, resource lock is enabled. A locked agent is considered a locked resource and other agents may continue normal PCI transactions.
0	**GAT (Guaranteed Access Time).** This bit enables/disables the guaranteed access time mode. When GAT = 1, the SIO is configured for Guaranteed Access Time mode. This mode is available in order to guarantee the 2.5 μs CHRDY time-out specification for the ISA Bus. When the SIO is an Initiator on behalf of an ISA master, the PCI and memory busses are arbitrated for in serial and must be owned before the ISA master is given ownership of the ISA Bus. When GAT = 0, the guaranteed access time mode is disabled. When guaranteed access time mode is disabled, the ISA master is first granted the ISA Bus and then the SIO arbitrates for the PCI Bus.

4.1.8 PAPC - PCI ARBITER PRIORITY CONTROL REGISTER

Register Name: PCI Arbiter Priority Control
Address Offset: 42h
Default Value: 04h
Attribute: Read /Write
Size: 8 bits

This register controls the PCI arbiter priority scheme. The arbiter supports four masters arranged through three switching banks. This permits the four masters to be arranged in a purely rotating priority scheme, one of eight fixed priority schemes, or a hybrid combination of the fixed and rotating priority schemes. Bits 0-2 determine which input for that particular bank has priority. This is shown in Figure 5-20. Bits 4-6 enable/disable rotate priority for each bank. For each bit, A 1 enables the mode and a 0 disables the mode. If rotate mode is enabled for a particular bank, the bank will operate in rotate mode and the fixed priority mode select for that bank is ignored. The PAPC Register fields are shown in Table 4-10.

Table 4-10. PCI Arbiter Priority Control Register

Bit	Description
7	Reserved. Read as 0.
6	Bank 2 Rotate Control
5	Bank 1 Rotate Control
4	Bank 0 Rotate Control
3	Reserved
2	Bank 2 Fixed Priority mode select
1	Bank 1 Fixed Priority mode select
0	Bank 0 Fixed Priority mode select

Fixed Priority Mode:

The fixed bank control bits select which requester is the highest priority device within that particular bank. Bits 4-6 must all be programmed to 0's (rotate mode disabled) to get these combinations. The eight selectable fixed priority schemes are listed in Table 4-11:

Table 4-11. Fixed Mode Bank Control Bits

Mode	Bank			Priority			
	2	1	0	Highest			Lowest
0	0	0	0	SIOREQ#	REQ0#	CPUREQ#	REQ1#
1	0	0	1	REQ0#	SIOREQ#	CPUREQ#	REQ1#
2	0	1	0	SIOREQ#	REQ0#	REQ1#	CPUREQ#
3	0	1	1	REQ0#	SIOREQ#	REQ1#	CPUREQ#
4	1	0	0	CPUREQ#	REQ1#	SIOREQ#	REQ0#
5	1	0	1	CPUREQ#	REQ1#	REQ0#	SIOREQ#
6	1	1	0	REQ1#	CPUREQ#	SIOREQ#	REQ0#
7	1	1	1	REQ1#	CPUREQ#	REQ0#	SIOREQ#

Rotating Priority Mode:

When any Bank Rotate Control bit is set to a one that particular bank rotates between the two requesting inputs. Any or all banks can be set in rotate mode. If all three banks are set in rotate mode, then the four supported masters are all rotated and the arbiter is a pure rotating priority scheme. If, within a rotating bank, the highest priority input does not have an active request, then the lower priority input will be granted the bus. However, this does not change the rotation scheme. When the bank toggles, the previously lowest priority input will become the highest priority input. Because of this, the maximum latency a device may encounter would be two complete rotations.

4.1.9 MCSCON - MEMCS# CONTROL REGISTER

Register Name: MEMCS# Control
Address Offset: 44h
Default value: 00h
Attribute: Read/Write
Size: 8 bits

Bits 0-2 of this register enable MEMCS# blocks. PCI addresses within the enabled blocks result in the generation of MEMCS#. Note that the 0-512 KByte segment does not have RE and WE bits. The 0 -512 KByte segment can only be turned off with the MEMCS# Master Enable bit (bit 4). Note also, that when the RE and WE bits are both 0 for a particular segment, the PCI master can not access the segment. The MCSCON fields are shown in Table 4-12.

Table 4-12. MEMCS# Control Register

Bit	Description
7:5	**Reserved.** Read as 0's.
4	**MEMCS# Master Enable.** When the MEMCS# master enable bit is set to a 1, the SIO asserts MEMCS# for all accesses to the defined MEMCS# region (that have been programmed in this register and the MAR1, MAR2, and MAR3 Registers). Also, when this bit is a 1, the positive decoding functions enabled by having the ISA Clock Divisor Register bit 6 = 1 and the Utility Bus Chip Select Register "A" bit 6 = 1 are ignored. Subtractive decoding is provided for these memory areas, instead. When the MEMCS# master enable bit is set to a 0, the entire MEMCS# function is disabled. When this bit is 0, MEMCS# will never be asserted.
3	**Write Enable For 0F0000h - 0FFFFFh (Upper 64 KByte BIOS).** When this bit is set to a 1, the SIO generates MEMCS# for PCI master memory write accesses to the address range 0F0000h - 0FFFFFh. When this bit is set to a 0, the SIO does not generate MEMCS# for PCI master memory write accesses to the address range 0F0000h - 0FFFFFh.
2	**Read Enable For 0F0000h - 0FFFFFh (Upper 64 KByte BIOS).** When this bit is set to a 1, the SIO generates MEMCS# for PCI master memory read accesses to the address range 0F0000h - 0FFFFFh. When this bit is set to a 0, the SIO does not generate MEMCS# for PCI master memory read accesses to the address range 0F0000h - 0FFFFFh.
1	**Write Enable For 080000h - 09FFFFh (512 KByte - 640 KByte).** When this bit is set to a 1, the SIO generates MEMCS# for PCI master memory write accesses to the address range 080000h - 09FFFFh. When this bit is set to a 0, the SIO does not generate MEMCS# for PCI master memory write accesses to the address range 080000h - 09FFFFh.
0	**Read Enable For 080000h - 09FFFFh (512 KByte - 640 KByte).** When this bit is set to a 1, the SIO generates MEMCS# for PCI master memory read accesses to the address range 080000h - 09FFFFh. When this bit is set to a 0, the SIO does not generate MEMCS# for PCI master memory read accesses to the address range 080000h - 09FFFFh.

4.1.10 MCSBOH - MEMCS# BOTTOM OF HOLE REGISTER

Register Name: MEMCS# Bottom of Hole
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. The MCSBOH fields are shown in Table 4-13.

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 effectively disabled. 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 effectively disables the hole.

Table 4-13. MEMCS# Bottom of Hole Register

Bit	Description
7	AD23
6	AD22
5	AD21
4	AD20
3	AD19
2	AD18
1	AD17
0	AD16

4.1.11 MCSTOH - MEMCS# TOP OF HOLE REGISTER

Register Name: MEMCS# Top of Hole
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. The MCSTOH fields are shown in Table 4-14.

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 effectively disabled. 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 effectively disables the hole.

Table 4-14. MEMCS# Top of Hole Register

Bit	Description
7	AD23
6	AD22
5	AD21
4	AD20
3	AD19
2	AD18
1	AD17
0	AD16

4.1.12 MCSTOM - MEMCS# TOP OF MEMORY REGISTER

Register Name: MEMCS# Top of Memory
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 up to 512 MBytes, in 2 MByte increments. This register is typically set to the top of main memory. Accesses ≥ 2 MByte and ≤ top of memory boundary results in the assertion of the MEMCS# signal (unless the address resides in the hole programmed by the MCSBOH and MCSTOH Registers). A value of 00h disables this 2 MByte-to-top memory region. A value of 00h assigns the top of memory to include 2 MByte - 1. A value of FFh assigns the top of memory to include 512 MByte - 1. The MCSTOM fields are shown in Table 4-15.

4.1.13 IADCON - ISA ADDRESS DECODER CONTROL REGISTER

Register Name: ISA Address Decoder Control
Address Offset: 48h
Default value: 01h
Attribute: Read/Write
Size: 8 bits

This register enables the forwarding of ISA or DMA memory cycles to the PCI Bus. In addition, this register sets the top of the "1 MByte to top of main memory" region. The IADCON fields are shown in Table 4-16.

Table 4-15. MEMCS# Top of Memory Register

Bit	Description
7	AD28
6	AD27
5	AD26
4	AD25
3	AD24
2	AD23
1	AD22
0	AD21

Table 4-16. ISA Address Decoder Control Register

Bit	Description
7:4	The top can be assigned in 1 MByte increments from 1 MByte up to 16 MByte. ISA master or DMA accesses within this region are forwarded to PCI unless they are within the hole. **Bits 7 6 5 4 — Top of Memory** 0 0 0 0 — 1 MByte 0 0 0 1 — 2 MByte 0 0 1 0 — 3 MByte 0 0 1 1 — 4 MByte 0 1 0 0 — 5 MByte 0 1 0 1 — 6 MByte 0 1 1 0 — 7 MByte 0 1 1 1 — 8 MByte 1 0 0 0 — 9 MByte 1 0 0 1 — 10 MByte 1 0 1 0 — 11 MByte 1 0 1 1 — 12 MByte 1 1 0 0 — 13 MByte 1 1 0 1 — 14 MByte 1 1 1 0 — 15 MByte 1 1 1 1 — 16 MByte
3:0	**ISA and DMA Memory Cycle To PCI Bus Enables.** The memory block is enabled by writing a 1 to the corresponding bit position. Setting the bit to 0 disables the corresponding block. ISA or DMA memory cycles to the enabled blocks result in the ISA cycle being forwarded to the PCI Bus. The BIOSCS# enable bit (bit 6 in the UBCSA Register) for the 896K-960K region overrides the function of bit 3 of this register. If the BIOSCS# bit is set to a 1, the ISA or DMA memory cycle is always contained to ISA, regardless of the setting of bit 3 in this register. If the BIOSCS# bit is disabled, the cycle is forwarded to the PCI bus if bit 3 in this register is enabled. Refer to Section 5 for a complete description of BIOS decoding. **Bit — Memory Block** 0 — 0-512 KByte Memory 1 — 512-640 KByte Memory 2 — 640-768 KByte VGA Memory 3 — 896-960 KByte Low BIOS

4.1.14 LADRBE - ISA ADDRESS DECODER ROM BLOCK ENABLE REGISTER

Register Name: ISA Address Decoder - ROM Block Enable (IADRBE)

Address Offset: 49h

Default value: 00h

Attribute: Read/Write

Size: 8 bits

ISA addresses within the enabled ranges result in the ISA memory cycle being forwarded to the PCI Bus. For each bit position, the memory block is enabled if the bit is set to 1 and is disabled if the bit is set to 0. If the memory block is disabled, the ISA cycle is not forwarded to the PCI Bus. The LADRBE fields are shown in Table 4-17.

Table 4-17. ISA Address Decoder ROM Block Enable Register

Bit	Description
7	880-896K Memory Enable
6	864-880K Memory Enable
5	848-864K Memory Enable
4	832-848K Memory Enable
3	816-832K Memory Enable
2	800-816K Memory Enable
1	784-800K Memory Enable
0	768-784K Memory Enable

4.1.15 LADBOH - ISA ADDRESS DECODER BOTTOM OF HOLE REGISTER

Register Name: ISA Address Decoder - Bottom of Hole
Address Offset: 4Ah
Default value: 10h
Attribute: Read/Write
Size: 8 bits

This register defines the bottom of the ISA Address Decoder hole. The hole is defined by the following equation: TOH ≥ address ≥ BOH, where BOH is the bottom of the hole address programmed into this register and TOH is the top of the hole address programmed into the IADTOH Register. ISA master or DMA addresses falling within the hole will not be forwarded to the PCI Bus. The hole can be sized in 64 KByte increments and placed anywhere between 1 MByte and 16 MByte on any 64 KByte boundary. It is the responsibility of the programmer to guarantee that the BOH is at or above 1 MByte. A[31:24] must be 0's for the hole, meaning the hole is restricted to be under the 16 MByte boundary. When TOH < BOH, the hole is effectively disabled. The default value for the BOH and TOH disables the hole. The COM fields are shown in Table 4-16.

For example, to program the BOH at 1 MByte, this register should be set to 10h. To program the BOH at 2 MBytes, this register should be set to 20h. To program the BOH at 8 MBytes, this register should be set to 80h. These settings are shown in the Figure 4-1.

Table 4-18. ISA Address Decoder Bottom of Hole Register

Bit	Description
7	A23
6	A22
5	A21
4	A20
3	A19
2	A18
1	A17
0	A16

4.1.16 IADTOH - ISA ADDRESS DECODER TOP OF HOLE REGISTER

Register Name: ISA Address Decoder - Top of Hole
Address Offset: 4Bh
Default value: 0Fh
Attribute: Read/Write
Size: 8 bits

This register defines the top of the ISA Address Decoder hole. The hole is defined by the following equation: TOH ≥ address ≥ BOH, where BOH is the bottom of the hole address programmed into the LADBOH Register and TOH is the top of the hole address programmed into this Register. ISA master or DMA addresses falling within the hole will not be forwarded to the PCI Bus. The hole can be sized in 64 KByte increments and placed anywhere between 1 MByte and 16 MByte on any 64 KByte boundary. It is the responsibility of the programmer to guarantee that the TOH is at or above 1 MByte. A[31:24] must be 0's for the hole, meaning the hole is restricted to be under the 16 MByte boundary. When TOH < BOH, the hole is disabled. The default value for the BOH and TOH disables the hole. The LADTOH field is shown in Table 4-19.

For example, to program the TOH at 1 MByte + 64 KByte, this register should be set to 10h. To program the TOH at 2 MByte + 128 KByte, this register should be set to 21h. To program the TOH at 12 MByte, this register should be set to BFh. These settings are shown in the Figure 4-1.

Table 4-19. ISA Address Decoder Top of Hole Register

Bit	Description
7	A23
6	A22
5	A21
4	A20
3	A19
2	A18
1	A17
0	A16

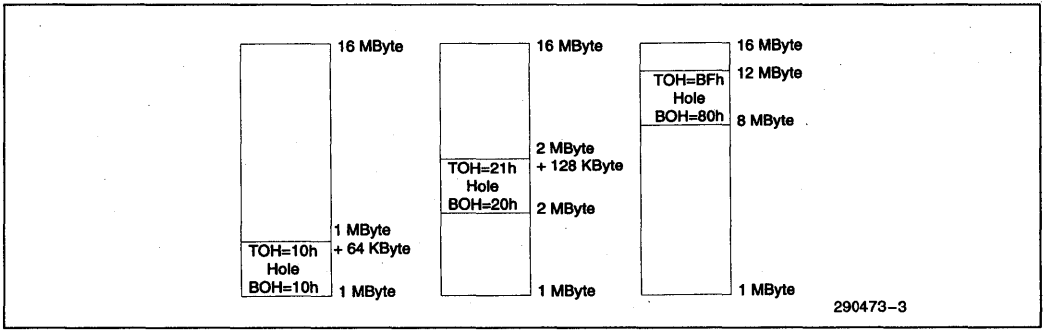


Figure 4-1. ISA Address Decoder Hole Examples

Table 4-20. Examples of ISA Decoding

Test Case Description	TOM (48h)	TOH (4Bh)	BOH (4Ah)	Address (hex)	Address	Result
8MB TOM, no hole @ 1M	7xh	0Fh	10h	01000000h	16MB	To PCI
				00FFFFFFh	16MB-1	ISA
				00800000h	8MB	ISA
				007FFFFFh	8MB-1	To PCI
				00100000h	1MB	To PCI
				000FFFFFh	1MD-1	ISA (BIOS)
4MB TOM, no hole @ 2M	3xh	1Fh	20h	01000000h	16MB	To PCI
				00FFFFFFh	16MB-1	ISA
				00400000h	4MB	ISA
				003FFFFFh	4MB-1	To PCI
				00200000h	2MB	To PCI
				001FFFFFh	2MB-1	To PCI
				00100000h	1MB	To PCI
1MB TOM, no hole @ 1M	0xh	0Fh	10H	01000000h	16MB	To PCI
				00FFFFFFh	16MB-1	ISA
				00100000h	1MB	ISA
				000FFFFFh	1MB-1	ISA (BIOS)
16MB TOM, 64KB hole @ 15MB	Fxh	F0h	F0h	01000000h	16MB	To PCI
				00FFFFFFh	16MB-1	To PCI
				00F10000h	15MB + 64KB	To PCI
				00F0FFFFh	15MB + 64KB-1	ISA
				00F00000h	15MB	ISA
				00EFFFFFh	15MB-1	To PCI
				00E10000h	14MB + 64KB	To PCI
				00E0FFFFh	14MB + 64KB-1	To PCI
				00E00000h	14MB	To PCI
				00DFFFFFh	14MB-1	To PCI
12MB TOM, 2MB + 128KB hole @ 2MB	Bxh	21h	20h	010000000h	16MB	To PCI
				00FFFFFFh	16MB-1	ISA
				00C00000h	12MB	ISA
				00BFFFFFh	12MB-1	To PCI
				00220000h	2MB + 128KB	To PCI
				0021FFFFh	2MB + 128KB-1	ISA
				00210000h	2MB + 64KB	ISA
				0020FFFFh	2MB + 64KB-1	ISA
				00200000h	2MB	ISA
				001FFFFFh	2MB-1	To PCI
				00100000h	1MB	To PCI
5MB TOM, 3MB hole @ 1.5MB	4xh	47H	18h	01000000h	16MB	To PCI
				00FFFFFFh	16MB-1	ISA
				00500000h	5MB	ISA
				004FFFFFh	5MB-1	To PCI
				00480000h	4.5MB	To PCI
				0047FFFFh	4.5MB-1	ISA
				00180000h	1.5MB	ISA
				0017FFFFh	1.5MB-1	To PCI
				00100000h	1MB	To PCI

4.1.17 ICRT - ISA CONTROLLER RECOVERY TIMER REGISTER

Register Name: ISA Controller Recovery Time
Address Offset: 4Ch
Default Value: 56h
Attribute: Read/Write
Size: 8 bits

The I/O recovery mechanism in the SIO is used to add additional recovery delay between PCI originated 8-bit and 16-bit I/O cycles to the ISA bus. The SIO automatically forces a minimum delay of four SYSCLKs between back-to-back 8 and 16 bit I/O cycles to the ISA bus. The delay is measured from the rising edge of the I/O command (IOR# or IOW#) to the falling edge of the next BALE. If a delay of greater than four SYSCLKs is required, the ISA I/O Recovery Time Register can be programmed to increase the delay in increments of SYSCLKs. 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 added to the standard I/O recovery. The ICRT fields are shown in Table 4-21.

Table 4-21. ISA Controller Recovery Timer Register

Bit	Description
7	**Reserved.** Read as 0.
6	**8-Bit I/O Recovery Enable.** This bit enables the recovery times programmed into bits [5:3] 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 ISA clock cycles (SYSCLK) added to the four minimum. The selected delay programmed into this field is enabled/disabled via bit 6 of this register.

Bit 5	4	3	SYSCLKs Added	Total SYSCLKs
0	0	1	+1	5
0	1	0	+2	6
0	1	1	+3	7
1	0	0	+4	8
1	0	1	+5	9
1	1	0	+6	10
1	1	1	+7	11
0	0	0	+8	12

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 ISA clock cycles (SYSCLK) added to the four minimum. The selected delay programmed into this field is enabled/disabled via bit 2 of this register.

Bit 1	0	SYSCLKs Added	Total SYSCLKs
0	1	+1	5
1	0	+2	6
1	1	+3	7
0	0	+4	8

4.1.18 ICD - ISA CLOCK DIVISOR REGISTER

Register Name: ISA Clock Divisor
Address Offset: 4Dh
Default Value: 40h
Attribute: Read/Write
Size: 8 bits

This register selects the integer value used to divide the PCI clock (PCICLK) to generate the ISA clock (SYSCLK). In addition, this register provides an ISA Reset bit to software control RSTDRV, a bit to enable/disable the MOUSE function, a bit to enable/disable the coprocessor error support, and a bit to disable the positive decode for the upper 64 KBytes of BIOS at the top of 1 MByte (F0000h - FFFFFh) and aliased regions. The ICD fields are shown in Table 4-22.

Table 4-22. ISA Clock Divisor Register

Bit	Description
7	**Reserved.**
6	**Positive Decode of Upper 64 KByte BIOS Enable.** This bit enables (bit 6 = 1) and disables (bit 6 = 0) the positive decode of the upper 64 KBytes of BIOS area at the top of 1 MByte (F0000h-FFFFFh) and the aliased regions at the top of 4 GBytes (FFFF0000h-FFFFFFFFh) and 4 GByte-1 MByte (FFEF0000-FFEFFFFFh). When bit 6 = 1, these address regions are positively decoded, unless bit 4 in the MEMCS# Control Register is set to a 1 in which case these regions are subtractively decoded. When bit 6 = 0, these address regions are subtractively decoded. The encoded chip selects for BIOSCS# and the UBUSOE# signal will always be generated when these locations are accessed, regardless of the state of this bit. A reset, sets this bit to a 1 (positive decode enabled).
5	**Coprocessor Error Enable.** This bit is used to enable and disable the Coprocessor error support. When enabled (bit 5 = 1), the FERR# input, when driven active, triggers an IRQ13 to the SIO's interrupt controller. FERR# is also used to gate the IGNNE# output. When disabled (bit 5 = 0), the FERR# signal can be used as IRQ13 and the coprocessor support is disabled. A reset sets this bit to 0 (coprocessor support disabled).
4	**IRQ12/M Mouse Function Enable.** When this bit is set to 1, IRQ12/M provides the mouse function. When this bit is set to 0, IRQ12/M provides the standard IRQ12 interrupt function. A hard reset sets this bit to 0.
3	**RSTDRV Enable.** This bit is used to enable RSTDRV on the ISA Bus. When this bit is set to 1, RSTDRV is asserted and remains asserted until this bit is set to a 0. When set to 0, normal operation of RSTDRV is provided. This bit should be used during configuration to reset the ISA Bus when changing the clock divisor. For a reset, this bit defaults to 0. Note that the software must ensure that RSTDRV is asserted for a minimum of 1 μs.
2:0	**PCICLK-to-ISA SYSCLK Divisor.** These bits are used to select the integer that is used to divide the PCICLK down to generate the ISA SYSCLK. Upon reset, these bits are set to 000 (divisor of 4 selected). For PCI frequencies less than 33 Mhz (not including 25 Mhz), a clock divisor value must be selected that ensures that the ISA Bus frequency does not violate the 6 Mhz to 8.33 Mhz SYSCLK specification. **Bit 2 1 0 Divisor SYSCLK** 0 0 0 4 (33 Mhz) 8.33 Mhz 0 0 1 3 (25 Mhz) 8.33 Mhz 0 1 0 Reserved 0 1 1 Reserved 1 0 0 Reserved 1 0 1 Reserved 1 1 0 Reserved 1 1 1 Reserved

4.1.19 UBCSA - UTILITY BUS CHIP SELECT A REGISTER

Register Name: Utility Bus Chip Select Enable A
Address Offset: 4Eh
Default Value: 07h
Attribute: Read/Write
Size: 8 bits

This register enables/disables accesses to the RTC, keyboard controller, Floppy Disk controller, IDE, and BIOS locations E0000h-EFFFFh and FFF80000h-FFFDFFFFh. Disabling any of these bits prevents the encoded chip select bits (ECSADDR[2:0]) and utility bus transceiver control signal (UBUSOE#) for that device from being generated.

This register is also used to select which address range (primary or secondary) will be decoded for the resident floppy controller and IDE. This ensures that there is no contention with the Utility bus transceiver driving the system data bus during read accesses to these devices. The UBCSA fields are shown in Table 4-23.

Table 4-23. UBCSA Register

Bit	Description
7	**Extended BIOS Enable.** When bit 7 = 1 (enabled), PCI accesses to locations FFF80000h-FFFDFFFFh result in the generation of the encoded signals (ECSADDR[2:0]) for BIOS. When enabled, PCI master accesses to this area are positively decoded and UBUSOE# is generated. When this bit is disabled (bit 7 = 0), the SIO does not generate the encoded (ECSADDR[2:0]) signals or UBUSOE#.
6	**Lower BIOS Enable.** When bit 6 = 1 (enabled), PCI or ISA accesses to the lower 64 KByte BIOS block (E0000h - EFFFFh) at the top of 1 MByte, or the aliases at the top of 4 GByte and 4 GByte - 1 MByte results in the generation of the encoded (ECSADDR[2:0]) signals for BIOS. When enabled, PCI master accesses to this area are positively decoded to the ISA Bus, unless bit 4 in the MEMCS# Control Register is set to a 1 in which case these regions are subtractively decoded.. Also, when enabled, ISA master or DMA master accesses to this region are not forwarded to the PCI Bus. When this bit is disabled (bit 6 = 0), the SIO does not generate the encoded (ECSADDR[2:0]) signals. Also, when this bit is disabled, ISA master or DMA accesses to this region are forwarded to PCI, if bit 3 in the IADCON Register is set to 1.
4	**IDE Decode Enable.** Bit 4 enables/disables IDE locations 1F0h-1F7h (primary) or 170h-177h (secondary) and 3F6h, 3F7h (primary) or 376h, 377h (secondary). When bit 4 = 1, the IDE encoded chip select signals and the Utility Bus transceiver signal (UBUSOE#) are generated for these addresses. When bit 4 = 0, the IDE encoded chip select signals and the Utility Bus transceiver signal (UBUSOE#) are not generated for these addresses.

Table 4-23. UBCSA Register (Continued)

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>5,3:2</td><td>Floppy Disk Address Locations Enable. Bits 2 and 3 are used to enable or disable the floppy locations as indicated below. A PCIRST# sets bit 2 to 1 and bit 3 to 0. Bit 5 is used to select between the primary and secondary address range used by the Floppy Controller and the IDE. Only primary or only secondary can be programmed at any one time. A PCIRST# sets this bit to 0 (primary).
The following table shows how these bits are used to select the floppy controller:
<table>
<tr><th>Address</th><th>Bit 5</th><th>Bit 3</th><th>Bit 2</th><th>DSKCHG</th><th>ECSADDR[2:0]</th><th>FLOPPYCS#</th></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>0</td><td>1 1 1</td><td>1</td></tr>
<tr><td>3F0h, 3F1h</td><td>0</td><td>1</td><td>X</td><td>1</td><td>1 0 0</td><td>0</td></tr>
<tr><td>3F2h - 3F7h</td><td>0</td><td>X</td><td>1</td><td>1</td><td>1 0 0</td><td>0 (note)</td></tr>
<tr><td>370h, 371h</td><td>1</td><td>1</td><td>X</td><td>1</td><td>1 0 0</td><td>0</td></tr>
<tr><td>372h - 37Fh</td><td>1</td><td>X</td><td>1</td><td>1</td><td>1 0 0</td><td>0 (note)</td></tr>
</table>
Note: If IDE decode is enabled (Bit 4 = 1), all accesses to locations 03F6h and 03F7h (primary) or 0376h and 0377h (secondary) result in the ECSADDR[2:0] signals generating a decode for IDECS1# (FLOPPYCS# is not generated). An external AND gate can be used to tie IDECS1# and FLOPPYCS# together to insure that the floppy is enabled for these accesses. If IDE decode is disabled (Bit 4 = 0), and the decode for the floppy is enabled, then the encoded chip selects for the floppy locations are generated.</td></tr>
<tr><td>1</td><td>Keyboard Controller Address Location Enable. Enables (1) or disables (0) the Keyboard controller address locations 60h, 62h, 64h, and 66h. When this bit is set to 0, the Keyboard Controller encoded chip select signals (ECSADDR[2:0]) and the Utility Bus transceiver signal (UBUSOE#) are not generated for these locations.</td></tr>
<tr><td>0</td><td>RTC Address Location Enable. Enables (1) or disables (0) the RTC address locations 70h - 77h. When this bit is set to 0, the RTC encoded chip select signals (ECSADDR[2:0]), RTCALE#, RTCCS#, and UBUSOE# signals are not generated for these addresses.</td></tr>
</table>

4.1.20 UBCSB - UTILITY BUS CHIP SELECT B REGISTER

Register Name: Utility Bus Chip Select B
Address Offset: 4Fh
Default Value: 4Fh
Attribute: Read/Write
Size: 8

This register is used to enable/disable accesses to the serial ports and parallel port locations supported by the SIO. When disabled, the ECSADDR(2:0) encoded chip select bits and Utility Bus Transceiver control signal (UBUSOE#), for that device, are not generated. This register is also used to disable accesses to port 92 and enable or disable configuration RAM decode. The UBCSB fields are shown in Table 4-24.

Table 4-24. Utility Bus Chip Select B Register

Bit	Description
7	**Configuration RAM Decode Enable.** This bit is used to enable (bit 7 = 1) or disable (bit 7 = 0) I/O write accesses to location 0C00h and I/O read/write accesses to locations 0800h-08FFh. When enabled, the encoded chip select signals for generating an external configuration page chip select (CPAGECS#) are generated for accesses to 0C00h. The encoded chip select signals for generating an external configuration memory chip select (CFIGMEMCS#) are generated for accesses to 0800h-08FFh. When bit 7 = 0, configuration RAM decode is disabled and the CPAGECS# and CFIGMEMCS# are not generated for the corresponding accesses.
6	**Port 92 Enable.** This bit is used to enable/disable access to Port 92. When bit 6 = 1, Port 92 is enabled. When bit 6 = 0, Port 92 is disabled. When a PCIRST# occurs, this bit is set to 1 (enable).
5:4	**Parallel Port Enable.** These bits are used to select the parallel port address range: (LPT1, LPT2, LPT3, or disable). When a PCIRST# occurs, this field is set to 00 (LPT1). **Bit 5 4 Function** 0 0 3BCh - 3BFh (LPT1) 0 1 378h - 37Fh (LPT2) 1 0 278h - 27Fh (LPT3) 1 1 Disabled
3:2	**Serial Port B Enable.** These bits are used to assign serial port B address range: (COM1, COM2, or disable). If either COM1 or COM2 address ranges are selected, the encoded chip select signals [ECSADDR(2:0)] for Port B will be generated. A PCIRST# sets bit[3:2] to 11 (Port B disabled). **Bit 3 2 Function** 0 0 3F8h - 3FFh (COM1) 0 1 2F8h - 2FFh (COM2) 1 0 Reserved 1 1 Port B Disabled **Note:** If Serial port A and B are programmed for the same I/O address, the encoded chip select signals, ECSADDR(2:0), for port B are disabled.
1:0	**Serial Port A Enable.** These bits are used to assign serial port A address range: (COM1, COM2, or disable). If either COM1 or COM2 address ranges are selected, the encoded chip select signals [ECSADDR(2:0)] for Port A will be generated. A PCIRST# sets bit[1:0] to 11 (Port A disabled). **Bit 1 0 Function** 0 0 3F8h - 3FFh (COM1) 0 1 2F8h - 2FFh (COM2) 1 0 Reserved 1 1 Port A disabled **Note:** If Serial port A and B are programmed for the same I/O address, the encoded chip select signals, ECSADDR(2:0), for port B are disabled.

4.1.21 MAR1 - MEMCS# ATTRIBUTE REGISTER #1

Register Name: 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 SIO generates MEMCS# for PCI master, DMA, or ISA master memory read accesses to the corresponding segment. When the RE bit is set to a 0, the SIO does not generate MEMCS# for PCI 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 ISA master can not access the corresponding segment.

WE - Write Enable. When the WE bit (bit 7, 5, 3, 1) is set to a 1, the SIO generates MEMCS# for PCI master, DMA, or ISA master memory write accesses to the corresponding segment. When this bit is set to a 0, the SIO does not generate MEMCS# for PCI 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 ISA master can not access the corresponding segment.

Table 4-25. MEMCS# Attribute Register #1

Bit	Description
7	0CC000h - 0CFFFFh Exp. ROM : WE
6	0CC000h - 0CFFFFh Exp. ROM : RE
5	0C8000h - 0CBFFFh Exp. ROM : WE
4	0C8000h - 0CBFFFh Exp. ROM : RE
3	0C4000h - 0C7FFFh Exp. ROM : WE
2	0C4000h - 0C7FFFh Exp. ROM : RE
1	0C0000h - 0C3FFFh Exp. ROM : WE
0	0C0000h - 0C3FFFh Exp. ROM : RE

4.1.22 MAR2 - MEMCS# ATTRIBUTE REGISTER #2

Register Name: 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 SIO generates MEMCS# for PCI master, DMA, or ISA master memory read accesses to the corresponding segment. When this bit is set to a 0, the SIO does not generate MEMCS# for PCI 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 ISA master can not access the corresponding segment.

WE - Write Enable. When the WE bit (bit 7, 5, 3, 1) is set to a 1, the SIO generates MEMCS# for PCI master , DMA, or ISA master memory write accesses to the corresponding segment. When this bit is set to a 0, the SIO does not generate MEMCS# for PCI 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 ISA master can not access the corresponding segment.

Table 4-26. MEMCS# Attribute Register #2

Bit	Description
7	0DC000h - 0DFFFFh Exp. ROM : WE
6	0DC000h - 0DFFFFh Exp. ROM : RE
5	0D8000h - 0DBFFFh Exp. ROM : WE
4	0D8000h - 0DBFFFh Exp. ROM : RE
3	0D4000h - 0D7FFFh Exp. ROM : WE
2	0D4000h - 0D7FFFh Exp. ROM : RE
1	0D0000h - 0D3FFFh Exp. ROM : WE
0	0D0000h - 0D3FFFh Exp. ROM : RE

4.1.23 MAR3 - MEMCS# ATTRIBUTE REGISTER #3

Register Name: 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 SIO generates MEMCS# for PCI master, DMA, ISA master memory read accesses to the corresponding segment. When this bit is set to a 0, the SIO does not generate MEMCS# for PCI 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.

WE - Write Enable. When the WE bit (bit 7, 5, 3, 1) is set to a 1, the SIO generates MEMCS# for PCI master , DMA, ISA master memory write accesses to the corresponding segment. When this bit is set to a 0, the SIO does not generate MEMCS# for PCI 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.

Table 4-27. MEMCS# Attribute Register #3

Bit	Description
7	0EC000h - 0EFFFFh Lower 64 KByte BIOS: WE
6	0EC000h - 0EFFFFh Lower 64 KByte BIOS: RE
5	0E8000h - 0EBFFFh Lower 64 KByte BIOS: WE
4	0E8000h - 0EBFFFh Lower 64 KByte BIOS: RE
3	0E4000h - 0E7FFFh Lower 64 KByte BIOS: WE
2	0E4000h - 0E7FFFh Lower 64 KByte BIOS: RE
1	0E0000h - 0E3FFFh Lower 64 KByte BIOS: WE
0	0E0000h - 0E3FFFh Lower 64 KByte BIOS: RE

4.1.24 DMA SCATTER/GATHER RELOCATION BASE ADDRESS REGISTER

Register Name: DMA Scatter/Gather Relocation Base Address
Address Offset: 57h
Default Value: 04h
Attribute: Read/Write
Size: 8 bits

The value programmed into this register determines the high order I/O address of the Scatter/Gather Command Registers, Scatter/Gather Status Registers, and the Scatter/Gather Descriptor Table Registers. The default value is 04h so the first S/G register default address is at 0410h.

Table 4-28. Scatter/Gather Relocation Base Address Register

Bit	Description
7	A15
6	A14
5	A13
4	A12
3	A11
2	A10
1	A9
0	A8

4.1.25 BIOS TIMER BASE ADDRESS REGISTER

Register Name: BIOS Timer Base Address
Address Offset: 80h - 81h
Default value: 0078h
Attribute: Read/Write
Size: 16 bits

This register determines the base address for the BIOS Timer Register located in the I/O space. The base address can be set at double-word boundary anywhere in the 64 KByte I/O space. This register also provides the BIOS Timer access enable/disable control bit.

Table 4-29. BIOS Timer Base Address Register

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

4.2 DMA Register Description

The SIO contains DMA circuitry that incorporates the functionality of two 82C37 DMA controllers (DMA1 and DMA2). The DMA registers control the operation of the DMA controllers and are all accessible from the PCI Bus via PCI I/O space. In addition, some of the registers are accessed from the ISA Bus via ISA I/O space. Table 4-2, at the beginning of Section 4.0. lists the bus access for each register.

This section describes the DMA registers. Unless otherwise stated, a PCIRST# sets each register to its default value. The operation of the DMA is further described in Section 5.4, DMA Controller.

4.2.1 DCOM - DMA Command Register

Register Name: DMA Command

Register Location: Channels 0-3 - 08h
Channels 4-7 - 0D0h

Default Value: 00h

Attribute: Write Only

Size: 8 bits

This 8-bit register controls the configuration of the DMA. It is programmed by the microprocessor in the Program Condition and is cleared by PCIRST# or a Master Clear instruction. Note that disabling channels 4-7 also disables channels 0–3, since channels 0–3 are cascaded onto channel 4. The DREQ and DACK# channel assertion sensitivity is assigned by channel group, not per individual channel. For priority resolution, the DMA consists of two logical channel groups — channels 0–3 (Controller 1 - DMA1) and channels 4–7 (Controller 2 - DMA2). Each group can be assigned fixed or rotating priority. Both groups can be assigned fixed priority, one group can be assigned fixed priority and the second rotating priority, or both groups can be assigned rotating priority. Following a PCIRST# or DMA Master Clear, both DMA1 and DMA2 are enabled in fixed priority, the DREQ sense level is active high, and the DACK# assertion level is active low.

Table 4-30. DMA Command Register

Bit	Description
7	**DACK# Assert Level (DACK# [3:0, (7:5)]).** Bit 7 controls the DMA channel request acknowledge (DACK#) assertion level. Following PCIRST#, the DACK# assertion level is active low. The low level indicates recognition and acknowledgment of the DMA request to the DMA slave requesting service. Writing a 0 to bit 7 assigns active low as the assertion level. When a 1 is written to this bit, a high level on the DACK# line indicates acknowledgment of the request for DMA service to the DMA slave.
6	**DREQ Sense Assert Level (DREQ[3:0, (7:5)]).** Bit 6 controls the DMA channel request (DREQ) assertion detect level. Following PCIRST#, the DREQ sense assert level is active high. In this condition, an active high level sampled on DREQ is decoded as an active DMA channel request. Writing a 0 to bit 6 assigns active high as the sense assert level. When a 1 is written to this bit, a low level on the DREQ line is decoded as an active DMA channel request.
5	**Reserved.** Must be 0.
4	**DMA Group Arbitration Priority.** Each channel group is individually assigned either fixed or rotating arbitration priority. At PCIRST#, each group is initialized in fixed priority. Writing a 0 to bit 4 assigns fixed priority to the channel group, while writing a 1 assigns rotating priority to the group.
3	**Reserved.** Must be 0
2	**DMA Channel Group Enable.** Writing a 1 to this bit disables the DMA channel group, while writing a 0 to this bit enables the DMA channel group. Both channel groups are enabled following PCIRST#. Disabling channel group 4-7 also disables channel group 0-3, which is cascaded through channel 4.
1:0	**Reserved.** Must be 0.

4.2.2 DCM - DMA CHANNEL MODE REGISTER

Register Name: DMA Channel Mode

Register Location: Channels 0-3 - 0Bh
Channels 4-7 - 0D6h

Default Value: Bits[7:2] = 0,
Bits[1:0] = undefined

Attribute: Write Only

Size: 6 bits

Each channel has a 6-bit DMA Channel Mode Register. The Channel Mode Registers provide control over DMA Transfer type, transfer mode, address increment/decrement, and autoinitialization. Bits [1:0] select the appropriate Channel Mode Register and are not stored. Only bits [7:2] are stored in the register. This register is set to its default value upon PCIRST# or Master Clear. Its default value is Verify transfer, Autoinitialize disable, Address increment, and Demand mode. Channel 4 defaults to cascade mode and cannot be programmed for any mode other than cascade mode.

Table 4-31. DMA Channel Mode Register

Bit	Description
7:6	**DMA Transfer Mode.** Each DMA channel can be programmed in one of four different modes: single transfer, block transfer, demand transfer and cascade. **Bits 7 6 Transfer Mode** 0 0 Demand mode 0 1 Single mode 1 0 Block mode 1 1 Cascade mode
5	**Address Increment/Decrement Select.** Bit 5 controls address increment/decrement during multi-byte DMA transfers. When bit 5 = 0, address increment is selected. When bit 5 = 1, address decrement is selected. Address increment is the default after a PCIRST# cycle or Master Clear command.
4	**Autoinitialize Enable.** When bit 4 = 1, the DMA restores the Base Page, Address, and Word count information to their respective current registers following a terminal count (TC). When bit 4 = 0, the autoinitialize feature is disabled and the DMA does not restore the above mentioned registers. A PCIRST# or Master Clear disables autoinitialization (sets bit 4 to 0).
3:2	**DMA Transfer Type.** Verify, write and read transfer types are available. Verify transfer is the default transfer type upon PCIRST# or Master Clear. Write transfers move data from an I/O device to memory. Read transfers move data from memory to an I/O device. Verify transfers are pseudo transfers; addresses are generated as in a normal read or write transfer and the device responds to EOP etc.. However, with Verify transfers, the ISA memory and I/O cycle lines are not driven. Bit combination 11 is illegal. When the channel is programmed for cascade ([7:6] = 11) the transfer type bits are irrelevant. **Bits 3 2 Transfer Type** 0 0 Verify transfer 0 1 Write transfer 1 0 Read Transfer 1 1 Illegal
1:0	**DMA Channel Select.** Bits [1:0] select the DMA Channel Mode Register that will be written by bits [7:2]. **Bits 1 0 Channel** 0 0 Channel 0 (4) 0 1 Channel 1 (5) 1 0 Channel 2 (6) 1 1 Channel 3 (7)

4.2.3 DCEM - DMA CHANNEL EXTEND MODE REGISTER

Register Name: DMA Channel Extended Mode Register

Register Location: Channels 0-3 - 040Bh
Channels 4-7 - 04D6h

Default Value: Bits[1:0] = undefined,
Bits[3:2] = 00 for DMA1,
Bits[3:2] = 01 for DMA2,
Bits[7:4] = 0

Attribute: Write Only

Size: 6 bits

Each channel has a 6-bit Extended Mode Register. The register is used to program the DMA device data size, timing mode, EOP input/output selection, and Stop Register selection. Bits [1:0] select the appropriate Channel Extend Mode Register and are not stored. Only bits [7:2] are stored in the register.

Four timing modes are available: ISA-compatible, A, B, and F. Timings A, B, and F are extended timing modes and can only be run to main memory. DMA cycles to ISA expansion bus memory defaults to compatible timing if the channel is programmed in an extended timing mode.

The default bit values for each DMA group are selected upon PCIRST#. A Master Clear or any other programming sequence will not set the default register settings. The default programmed values for DMA1 channels 0-3 are 8-bit I/O count by bytes, compatible timing, and EOP output. The default values for DMA2 channels 4-7 are 16-bit I/O count by words with shifted address, compatible timing, and EOP output.

Table 4-32. DMA Channel Extend Mode Register

Bit	Description
7	**Reserved.** Must be 0.
6	**EOP Input/Output Selection.** Bit 6 selects whether the EOP signal is to be used as an output during DMA transfers on this channel or an input. EOP is typically used as an output, as was available on the PC-AT. The input function was added to support data communication and other devices that would like to trigger an autoinitialize when a collision or some other event occurs. The direction of EOP is switched when DACK is changed (when a different channel is granted the bus). There may be some overlap of the SIO driving the EOP signal along with the DMA slave. However, during this overlap, both devices drive the signal to a low level (inactive). For example, assume channel 2 is about to go inactive (DACK negating) and channel 1 is about to go active. In addition, assume that channel 2 is programmed for "EOP OUT" and channel 1 is programmed for "EOP IN". When channel 2's DACK is negated and channel 1's DACK is asserted, the SIO may be driving EOP to a low value on behalf of channel 2. At the same time the device connected to channel 1 is driving EOP in to the SIO, also at an inactive level. This overlap only lasts until the SIO EOP output buffer is tri-stated, and does not effect the DMA operation. Upon PCIRST#, bit 6 is set to 0 - EOP output selected.
5:4	**DMA Cycle Timing Mode.** The SIO supports four DMA transfer timings: ISA-compatible, Type A, Type B, and Type F. Each timing and its corresponding code are described below. Upon PCIRST#, compatible timing is selected and the value of these bits is "00". The cycle timings noted below are for a SYSCLK (8.33 Mhz) (maximum SYSCLK frequency). DMA cycles to ISA expansion bus memory defaults to compatible timing if the channel is programmed in one of the performance timing modes (Type A, B, or F). **Bits[5:4] = 00: Compatible Timing** Compatible timing is provided for DMA slave devices, that, due to some design limitation, cannot support one of the faster timings. Compatible timing runs at 9 SYSCLKs (1080 nsec/single cycle) and 8 SYSCLKs (960 nsec/cycle) during the repeated portion of a BLOCK or DEMAND mode transfer. **Bits[5:4] = 01: Type "A" Timing** Type "A" timing is provided to allow shorter cycles to main memory (via the PCI Bus). Type "A" timing runs at 6 SYSCLKs (720 nsec/cycle) during the repeated portion of a BLOCK or DEMAND mode transfer. Type "A" timing varies from compatible timing primarily in shortening the memory operation to the minimum allowed main memory. The I/O portion of the cycle (data setup on write, I/O read access time) is the same as with compatible cycles. The actual active command time is shorter. However, it is expected that the DMA devices that provide the data access time or write data setup time should not require excess IOR# or IOW# command active time. Because of this, most ISA DMA devices should be able to use type "A" timing.

Table 4-32. DMA Channel Extend Mode Register (Continued)

Bit	Description
5:4	**Bits[5:4] = 10 Type "B" Timing** Type "B" timing is provided for 8/16-bit ISA DMA devices that can accept faster I/O timing. Type "B" only works with fast main memory. Type "B" timing runs at 5 SYSCLKs (600 nsec/cycle) during the repeated portion of a BLOCK or DEMAND mode transfer. Type "B" timing requires faster DMA slave devices than compatible timing. In Type "B" timing the cycles are shortened so that the data setup time on I/O write cycles is shortened and the I/O read access time is required to be faster. **Bits[5:4] = 11 Type "F" Timing** Type "F" timing provides high performance DMA transfer capability. Type "F" timing runs at 3 SYSCLKs (360 nsec/single cycle) during the repeated portion of a BLOCK or DEMAND mode transfer, resulting in a maximum data transfer rate of 8.33 MBytes/second.
3:2	**Addressing Mode.** The SIO supports both 8 and 16 bit DMA device data sizes. Three data size options are programmable with bits [3:2]. Both the 8-bit I/O, "count by bytes" mode and the 16-bit I/O, "count by words" (Address Shifted) mode are ISA compatible. The 16-bit I/O, "count by bytes" mode is offered as an extension of the ISA compatible modes. Bits [3:2] = 10 is reserved. Byte assembly/disassembly is performed by the ISA control unit. Each of the data transfer size modes is discussed below. **Bits[3:2] = 00: 8-bit I/O, "Count By Bytes" Mode** In 8-bit I/O, "count by bytes" mode, the Current Address Register can be programmed to any address. The Current Byte/Word Count Register is programmed with the "number of bytes minus 1" to transfer. **Bits[3:2] = 01: 16-bit I/O, "Count By Words" (Address Shifted) Mode** In "count by words" mode (address shifted), the Current Address Register can be programmed to any even address, but must be programmed with the address value shifted right by one bit. The Low Page and High Page Registers are not shifted during DMA transfers. Thus, the least significant bit of the Low Page register is ignored when the address is driven out onto the bus. The Current Byte/Word Count Register is programmed with the number of words minus 1 to be transferred. **Bits[3:2] = 10: Reserved** **Bits[3:2] = 11: 16-Bit I/O, "Count By Bytes" Mode** In 16-bit "count by bytes" mode, the Current Address Register can be programmed to any byte address. For most DMA devices, however, it should be programmed only to even addresses. If the address is programmed to an odd address, the DMA controller does a partial word transfer during the first and last transfer, if necessary. The bus controller does the byte/word assembly necessary to write any size memory device. In this mode, the Current Address Register is incremented or decremented by two and the byte count is decremented by the number of bytes transferred during each bus cycle. The Current Byte/Word Count Register is programmed with the "number of bytes minus 1" to be transferred. This mode is offered as an extension of the two ISA compatible modes discussed above. This mode should only be programmed for 16-bit ISA DMA slaves.
1:0	**DMA Channel Select.** Bits [1:0] selects the particular channel that will have its DMA Channel Extend Mode Register programmed with bits [7:2]. **Bits 1 0 Channel** 0 0 Channel 0 (4) 0 1 Channel 1 (5) 1 0 Channel 2 (6) 1 1 Channel 3 (7)

4.2.4 DR - DMA REQUEST REGISTER

Register Name: DMA Request Register

Register Location: Channels 0-3 - 09h
Channels 4-7 - 0D2h

Default Value: Bits[1:0] = undefined,
Bits[7:2] = 0

Attribute: Write Only

Size: 4 bits

Each channel has a request bit in one of the two 4-bit DMA Request Registers. The Request Register is used by software to initiate a DMA request. The DMA responds to the software request as though DREQ[x] is asserted. These requests are non-maskable and subject to prioritization by the priority encoder network. Each register bit is set to 1 or 0 separately under software control or is set to 0 upon generation of a TC. The entire register is set to 0 upon PCIRST# or a Master Clear. It is not affected upon a RSTDRV output. To program a bit, the software loads the proper form of the data word. Bits [1:0] determine which channel Request Register will be written. In order to make a software request, the channel must be in Block Mode. The Request Register status for DMA1 and DMA2 is output on bits [7:4] of a Status Register read to the appropriate port.

Table 4-33. DMA Request Register

Bit	Description
7:3	**Reserved.** Must be 0
2	**DMA Channel Service Request.** Writing a 0 to bit 2 resets the individual software DMA channel request bit. Writing a 1 to bit 2 sets the request bit. The request bit for each DMA channel is reset to 0 upon a PCIRST# or a Master Clear.
1:0	**DMA Channel Select.** Bits [1:0] select the DMA channel mode register to program with bit 2. **Bits 1 0 Channel** 0 0 Channel 0 0 1 Channel 1 (5) 1 0 Channel 2 (6) 1 1 Channel 3 (7)

4.2.5 MASK REGISTER - WRITE SINGLE MASK BIT

Register Name: Mask Register - Write Single Mask Bit

Register Location: Channels 0-3 - 0Ah
Channels 4-7 - 0D4h

Default Value: Bits[1:0] = undefined, Bit 2 = 1, Bits[7:3] = 0

Attribute: Write Only

Size: 1 bit/channel

Each DMA channel has a mask bit that enables/disables an incoming DMA channel service request DREQ[x]. Two 4-bit registers store the current mask status for DMA1 and DMA2. Setting the mask bit disables the incoming DREQ[x] for that channel. Clearing the mask bit enables the incoming DREQ[x]. A channel's mask bit is automatically set when the Current Byte/Word Count register reaches terminal count (unless the channel is programmed for autoinitialization). Each mask bit may also be set or cleared under software control. The entire register is also set by a PCIRST# or a Master Clear. Setting the entire register disables all DMA requests until a clear mask register instruction allows them to occur. This instruction format is similar to the format used with the DMA Request Register.

Individually masking DMA channel 4 (DMA controller 2, channel 0) will automatically mask DMA channels [3:0], as this channel group is logically cascaded onto channel 4. Setting this mask bit disables the incoming DREQ's for channels [3:0].

Table 4-34. Write Single Mask Bit Register

Bit	Description
7:3	**Reserved.** Must be 0.
2	**Channel Mask Select.** When bit 2 is set to a 1, DREQ is disabled for the selected channel. When bit 2 is set to a 0, DREQ is enabled for the selected channel.
1:0	**DMA Channel Select.** Bits [1:0] select the DMA Channel Mode Register to program with bit 2. **Bits 1 0 Channel** 0 0 Channel 0 (4) 0 1 Channel 1 (5) 1 0 Channel 2 (6) 1 1 Channel 3 (7)

4.2.6 MASK REGISTER - WRITE ALL MASK BITS

Register Name: Mask Register - Write All Mask Bits

Register Location: Channels 0-3 - 0Fh
Channels 4-7 - 0DEh

Default Value: Bit[3:0] = 1, Bit[7:4] = 0

Attribute: Read/Write

Size: 4 bits

Writing to this register enables/disables incoming DREQ assertions. There are four mask bits per register, one for each channel. This permits all four channels to be simultaneously enabled/disabled instead of enabling/disabling each channel individually, as is the case with the Mask Register - Write Single Mask Bit.

Two 4-bit registers store the current mask status for DMA1 and DMA2. Unlike the Mask Register - Write Single Mask Bit, this register and includes a status read to check the current mask status of the selected DMA channel group. A channel's mask bit is automatically set to 1 when the Current Byte/Word Count Register reaches terminal count (unless the channel is programmed for autoinitialization). Bits [3:0] are set to 1 by a PCIRST# or a Master Clear. Setting bits [3:0] to 1 disables all DMA requests until a clear mask register instruction enables the requests.

Two important points should be taken into consideration when programming the mask registers. First, individually masking DMA channel 4 (DMA controller 2, channel 0) will automatically mask DMA channels [3:0], as this channel group is logically cascaded onto channel 4. Second, masking DMA controller 2 with a write to port 0DEh will also mask DREQ assertions from DMA controller 1 for the same reason. When DMA channel 4 is masked, so are DMA channels 0-3.

Table 4-35. Write All Mask Bits Register

Bit	Description
7:4	**Reserved.** Must be 0.
3:0	**Channel Mask Bits.** Setting the bit(s) to a 1 disables the corresponding DREQ(s). Setting the bit(s) to a 0 enables the corresponding DREQ(s). Bits [3:0] are set to 1 upon PCIRST# or Master Clear. When read, bits [3:0] indicate the DMA channel [3:0] ([7:4]) mask status. **Bit Channel** 0 0 (4) 1 1 (5) 2 2 (6) 3 3 (7) Note: Disabling channel 4 also disables channels 0-3 due to the cascade of DMA1 through channel 4 of DMA2.

4.2.7 DS - DMA STATUS REGISTER

Register Name: DMA Status

Register Location: Channels 0-3 - 08h
Channels 4-7 - 0D0h

Default Value: 00h

Attribute: Read Only

Size: 8 bits

Each DMA controller has a read-only DMA Status Register. This register indicates which channels have reached terminal count and which channels have a pending DMA request. Bits [3:0] are set every time the corresponding TC is reached by that channel. Bits [3:0] are set to 0 upon PCIRST# and on each status read. Bits [7:4] are set whenever their corresponding channel is requesting service.

Table 4-36. DMA Status Register

Bit	Description
7:4	**Channel Request Status.** When a valid DMA request is pending for a channel (on its DREQ signal line), the corresponding bit is set to 1. When a DMA request is not pending for a particular channel, the corresponding bit is set to 0. The source of the DREQ may be hardware, a timed-out block transfer, or a software request. Note that channel 4 does not have DREQ or DACK lines, so the response for a read of DMA2 status for channel 4 is irrelevant. **Bit Channel** 4 0 5 1 (5) 6 2 (6) 7 3 (7)
3:0	**Channel Terminal Count Status.** When a channel reaches terminal count (TC), its status bit is set to 1. If TC has not been reached, the status bit is set to 0. Note that channel 4 is programmed for cascade, and is not used for a DMA transfer. Therefore, the TC bit response for a status read on DMA2 for channel 4 is irrelevant. **Bit Channel** 0 0 1 1 (5) 2 2 (6) 3 3 (7)

4.2.8 DMA BASE AND CURRENT ADDRESS REGISTERS (8237 COMPATIBLE SEGMENT)

Register Name: DMA Base and Current Address Register (8237 compatible segment)

Register Location: DMA Channel 0 - 000h
DMA Channel 1 - 002h
DMA Channel 2 - 004h
DMA Channel 3 - 006h
DMA Channel 4 - 0C0h
DMA Channel 5 - 0C4h
DMA Channel 6 - 0C8h
DMA Channel 7 - 0CCh

Default Value: All bits undefined

Attribute: Read/Write

Size: 16 bits per channel

Each channel has a 16-bit Current Address Register. This register contains the value of the 16 least significant bits of the full 32-bit address used during DMA transfers. The address is automatically incremented or decremented after each transfer and the intermediate values of the address are stored in the Current Address Register during the transfer. This register is written to or read from by the PCI Bus or ISA Bus master in successive 8-bit bytes. The programmer must issue the "Clear Byte Pointer Flip-Flop" command to reset the internal byte pointer and correctly align the write prior to programming the Current Address Register. After clearing the Byte Pointer Flip-Flop, the first write to the Current Address Register programs the low byte, bits [7:0], and the second write programs the high byte, bits [15:8]. This procedure also applies to read cycles. It may also be reinitialized by an Autoinitialize back to its original value. Autoinitialize takes place only after a TC or EOP.

Each channel has a Base Address Register located at the same port address as the corresponding Current Address Register. These registers store the original value of their associated Current Address Registers. During autoinitialize these values are used to restore the Current Address Registers to their original values. The Base Registers are written simultaneously with their corresponding Current Address Register in successive 8-bit bytes. The Base Registers are write-only.

Table 4-37. DMA Base and Current Address Register

Bit	Description
15:0	**Base and Current Address [15:0].** These bits represent the 16 least significant address bits used during DMA transfers. Together with the DMA Low Page Register, they form the ISA-compatible 24-bit DMA address. As an extension of the ISA compatible functionality, the DMA High Page Register completes the 32-bit address needed when implementing SIO extensions such as DMA to the PCI Bus slaves that can take advantage of full 32-bit addressability. Upon PCIRST# or Master Clear, the value of these bits is 0000h.

4.2.9 DMA BASE AND CURRENT BYTE/WORD COUNT REGISTERS (8237 COMPATIBLE SEGMENT)

Register Name: DMA Base and Current Byte/Word Count Register (8237 compatible segment)

Register Location:
DMA Channel 0 - 001h
DMA Channel 1 - 003h
DMA Channel 2 - 005h
DMA Channel 3 - 007h
DMA Channel 4 - 0C2h
DMA Channel 5 - 0C6h
DMA Channel 6 - 0CAh
DMA Channel 7 - 0CEh

Default Value: All bits undefined

Attribute: Read/Write

Size: 16 bits per channel

Each channel has a 16-bit Current Byte/Word Count Register. This register determines the number of transfers to be performed. The actual number of transfers is one more than the number programmed in the Current Byte/Word Count Register (i.e., programming a count of 100 results in 101 transfers). The byte/word count is decremented after each transfer. The intermediate value of the byte/word count is stored in the register during the transfer. When the value in the register goes from zero to 0FFFFh, a TC is generated.

Following the end of a DMA service the register may also be re-initialized by an autoinitialization back to its original value. Autoinitialize can only occur when a TC occurs. If it is not autoinitialized, this register has a count of FFFFh after TC.

When the Extended Mode Register is programmed for, or defaulted to, transfers to/from an 8-bit I/O, the Byte/Word count indicates the number of bytes to be transferred.

When the Extended Mode Register is programmed for, or defaulted to, transfers to/from a 16-bit I/O, with shifted address, the Byte/Word count indicates the number of 16-bit words to be transferred.

When the Extended Mode Register is programmed for transfers to/from a 16-bit I/O, the Byte/Word Count indicates the number of bytes to be transferred. The number of bytes does not need to be a multiple of two or four in this case.

Each channel has a Base Byte/Word Count Register located at the same port address as the corresponding Current Byte/Word Count Register. These registers store the original value of their associated Current Byte/Word Count Registers. During Autoinitialize these values are used to restore the Current registers to their original values. The Base registers are written simultaneously with their corresponding Current register in successive 8-bit bytes. The Base registers cannot be read by any external agents.

Table 4-38. DMA Base and Current Byte/Word Count Register

Bit	Description
15:0	**Base and Current Byte/ Word Count.** These bits represent the 16 byte/word count bits used when counting down a DMA transfer. Upon PCIRST# or Master Clear, the value of these bits is 0000h.

4.2.10 DMA MEMORY BASE LOW PAGE AND CURRENT LOW PAGE REGISTERS

Register Name: DMA Memory Low Page Register (Read/Write)
DMA Memory Base Low Page Register (Write Only)

Register Location: DMA Channel 0 - 087h
DMA Channel 1 - 083h
DMA Channel 2 - 081h
DMA Channel 3 - 082h
DMA Channel 5 - 08Bh
DMA Channel 6 - 089h
DMA Channel 7 - 08Ah

Default Value: All bits undefined

Size: 8 bits per channel

Each channel has an 8-bit Low Page Register. The DMA memory Low Page Register contains the eight second most-significant bits of the 32-bit address. The register works in conjunction with the DMA controller's High Page Register and Current Address Register to define the complete (32-bit) address for the DMA channel. This 8-bit register is read or written directly. It may also be re-initialized by an autoinitialize back to its original value. Autoinitialize takes place only after a TC or EOP.

Each channel has a Base Low Page Address Register located at the same port address as the corresponding Current Low Page Register. These registers store the original value of their associated Current Low Page Registers. During autoinitialization, these values are used to restore the Current Low Page Registers to their original values. The 8-bit Base Low Page Registers are written simultaneously with their corresponding Current Low Page Register by the microprocessor. The Base Low Page registers are write only.

Table 4-39. DMA Memory Low Page and Base Low Page Register

Bit	Description
7:0	**DMA Low Page and Base Low Page [23:16].** These bits represent the eight second most significant address bits when forming the full 32-bit address for a DMA transfer. Upon PCIRST# or Master Clear, the value of these bits is 00h.

4.2.11 DMA MEMORY BASE HIGH PAGE AND CURRENT HIGH PAGE REGISTERS

Register Name: DMA Memory Current High Page Register (Read/Write)
DMA Memory Base High Page Register (Write Only)

Register Location: DMA Channel 0 - 0487h
DMA Channel 1 - 0483h
DMA Channel 2 - 0481h
DMA Channel 3 - 0482h
DMA Channel 5 - 048Bh
DMA Channel 6 - 0489h
DMA Channel 7 - 048Ah

Default Value: All bits undefined

Size: 8 bits per channel

Each channel has an 8-bit Current High Page Register. The DMA memory Current High Page Register contains the eight most significant bits of the 32-bit address. The register works in conjunction with the DMA controller's Current Low Page Register and

Current Address Register to define the complete (32-bit) address for the DMA channels and corresponds to the Current Address Register for each channel. This 8-bit register is read or written directly. It may also be autoinitialized back to its original value. Autoinitialize takes place only after a TC or EOP.

This register is set to 0 during the programming of both the Current Low Page Register and the Current Address Register. Thus, if this register is not programmed after the other address and Low Page Registers are programmed, then its value is 00h. In this case, the DMA channel operates the same as an 82C37 (from an addressing standpoint). This is the address compatibility mode.

If the high 8 bits of the address are programmed after the other addresses, then the channel modifies its operation to increment (or decrement) the entire 32-bit address. This is unlike the 82C37 "Page" register in the original PCs which could only increment to a 64 KByte boundary for 8-bit channels or 128 KByte boundary for 16-bit channels. This is extended address mode. In this mode, the ISA Bus controller generates the signals MEMR# and MEMW# only for addresses below 16 MBytes.

Each channel has a Base High Page Register located at the same port address as the corresponding Current High Page Register. These registers store the original value of their associated Current High Page Registers. During autoinitialize, these values are used to restore the Current High Page Registers to their original values. The 8-bit Base High Page Registers are written simultaneously with their corresponding Current High Page Register. The Base High Page Registers are write only.

Table 4-40. DMA Base High Page and Base High Page Registers

Bit	Description
7:0	**DMA High Page and Base High Page [31:24].** These bits represent the eight most-significant address bits when forming the full 32-bit address for a DMA transfer. Upon PCIRST# or Master Clear, the value of these bits is 00h.

4.2.12 DMA CLEAR BYTE POINTER REGISTER

Register Name: DMA Clear Byte Pointer Flip-Flop

Register Location: Channels 0-3 - 00Ch
Channels 4-7 - 0D8h

Default Value: All bits undefined

Attribute: Write Only

Size: 8 bits

Writing to this register executes the clear byte pointer command. This command is executed prior to writing or reading new address or word count information to the DMA. This command initializes the byte pointer flip-flop to a known state so that subsequent accesses to register contents will address upper and lower bytes in the correct sequence.

The clear byte pointer command clears the internal latch used to address the upper or lower byte of the 16-bit Address and Word Count Registers. The latch is also cleared at power on by PCIRST# and by the Master Clear command. The Host CPU may read or write a 16-bit DMA controller register by performing two consecutive accesses to the I/O port. The Clear Byte Pointer command precedes the first access. The first I/O write to a register port loads the least significant byte, and the second access automatically accesses the most significant byte.

When DMA registers are being read or written, two Byte Pointer flip-flops are used. One flip-flop is for channels 0-3 and one for channels 4-7. Both of these act independently. There are separate software commands for clearing each of them (0Ch for Channels 0-3, 0D8h for Channels 4-7).

Table 4-41. Clear Byte Pointer Register

Bit	Description
7:0	**Clear Byte Pointer.** No specific pattern. Command enabled with a write to the I/O port address.

4.2.13 DMC - DMA MASTER CLEAR REGISTER

Register Name: DMA Master Clear
Register Location: Channel 0-3 - 00Dh
Channel 4-7 - 0DAh
Default Value: All bits undefined
Attribute: Write Only
Size: 8 bit

This software instruction has the same effect as the hardware Reset. The Command, Status, Request, and Internal First/Last Flip-Flop registers are cleared and the Mask Register is set. The DMA controller enters the idle cycle. There are two independent Master Clear Commands; 0Dh acts on channels 0-3, and 0DAh acts on channels 4-7.

Table 4-42. Master Clear Register

Bit	Description
7:0	**Master Clear.** No specific pattern. Command enabled with a write to the I/O port address.

4.2.14 DCM - DMA CLEAR MASK REGISTER

Register Name: DMA Clear Mask
Register Location: Channel 0-3 - 00Eh
Channel 4-7 - 0DCh
Default Value: All bits undefined
Attribute: Write Only
Size: 8 bit

This command clears the mask bits of all four channels, enabling them to accept DMA requests. I/O port 0Eh is used for Channels 0-3 and I/O port 0DCh is used for Channels 4-7.

Table 4.43. DMA Clear Mask Register

Bit	Description
7:0	**Clear Mask Register.** No specific pattern. Command enabled with a write to the I/O port address.

4.2.15 SCATTER/GATHER COMMAND REGISTER

Register Name: DMA Scatter Gather Command
Register Location:
Channels 0 default address - 0410h
Channels 1 default address - 0411h
Channels 2 default address - 0412h
Channels 3 default address - 0413h
Channels 5 default address - 0415h
Channels 6 default address - 0416h
Channels 7 default address - 0417h
Default Value: 00h
Attribute: Write Only, Relocatable
Size: 8 bits

NOTE:
The Base Registers and Current Registers described in the Scatter/Gather sections are hidden registers that are not accessible by software.

The Scatter/Gather Command Register controls operation of the descriptor table aspect of scatter/gather transfers. This register can be used to start and stop a scatter/gather transfer. The register can also be used to select between IRQ13 and EOP to be asserted following a terminal count. The current scatter/gather transfer status can be read in the scatter/gather channel's corresponding Scatter/Gather Status Register. After a PCIRST# or Master Clear, IRQ13 is disabled and EOP is enabled.

Table 4-44. Scatter/Gather Register

Bit	Description Number
7	**IRQ13/EOP Select.** Bit 7, if enabled via bit 6 of this register, selects whether EOP or IRQ13 is asserted at termination caused by a last buffer expiring. The last buffer can be either the last buffer in the list or the last buffer loaded in the DMA while it is suspended. If bit 7 = 1 (and bit 6 = 1), EOP is asserted when the last buffer is completed. If bit 7 = 0 (and bit 6 = 1), IRQ13 is asserted when the last buffer is completed. EOP can be used to alert an expansion bus I/O device that a scatter-gather termination condition was reached. The I/O device, in turn, can assert its own interrupt request line to invoke a dedicated interrupt handling routine. IRQ13 should be used when the CPU needs to be notified directly. Following PCIRST#, or Master Clear, the value stored for this bit is "1", and EOP is selected. Bit-6 must be set to a "1" to enable this bit during a S/G Command register write. When bit-6 is a "0" during the write, bit-7 will not have any effect on the current EOP/IRQ13 selection.
6	**IRQ13/EOP Programming Enable.** Enabling IRQ13/EOP programming allows initialization or modification of the S/G termination handling bits. When bit 6 = 0, bit 7 does not affect the state of IRQ13 or EOP assertion. When bit 6 = 1, bit 7 determines the termination handling following a terminal count.
5:2	**Reserved.** Must be 0.
1:0	**Scatter/Gather Commands.** This 2-bit field is used to start and stop scatter/gather. **Bits[1:0] = 00: No S/G operation** No S/G command operation is performed. Bits[7:6] may still be used to program IRQ13/EOP selection. **Bits[1:0] = 01: Start S/G Command** The Start command initiates the scatter-gather process. Immediately after the start command is issued (setting bits[1:0] to 01), a request is issued to fetch the initial buffer from the descriptor table to fill the Base Register set in preparation for performing a transfer. The buffer prefetch request has the same priority with respect to other channels as the DREQ it is associated with. Within the channel, DREQ is higher in priority than a prefetch request. The Start command assumes the Base and Current registers are both empty and will request a prefetch automatically. Note that this command also sets the Scatter/Gather Status Register to S/G Active, Base Empty, Current Empty, not Terminated, and Next Null Indicator to 0. The EOP/IRQ13 bit will still reflect the last value programmed **Bits[1:0] = 10: Stop S/G Command** The Stop command halts a Scatter-Gather transfer immediately. When a Stop command is given, the Terminate bit in the S/G Status register and the DMA channel mask bit are both set. **Bits[1:0] = 11: Reserved**

4.2.16 SCATTER/GATHER STATUS REGISTER

Register Name: Scatter/Gather Status

Register Location:

Channels 0 default address - 0418h
Channels 1 default address - 0419h
Channels 2 default address - 041Ah
Channels 3 default address - 041Bh
Channels 5 default address - 041Dh
Channels 6 default address - 041Eh
Channels 7 default address - 041Fh

Default Value: 00h

Attribute: Read Only, Relocatable

Size: 8-bits

NOTE:
The Base Registers and Current Registers described in the Scatter/Gather sections are hidden registers that are not accessible by software.

The Scatter/Gather Status Register contains information on the scatter/gather transfer status. This register provides dynamic status information on S/G transfer activity, the current and base buffer state, S/G transfer termination, and the End of the List indicator.

An Active bit is set to "1" after the S/G Start command is issued. The Active bit will be "0" before the initial Start command, following a terminal count, and after a S/G Stop command is issued. The Current Register and Base Register Status bits indicate whether the corresponding register has a buffer loaded. It is possible for the Base Register Status to be set while the Current Register Status is cleared. When the Current Register transfer is complete, the Base Register will not be moved into the Current Register until the start of the next data transfer. Thus, the Current Register State is empty (cleared), while the Base Register State is full (set). The Terminate bit is set active after a Stop command, after TC for the last buffer in the list, and both Base and Current Registers have expired. The EOP and IRQ13 bits indicate which end of process indicator will be used to alert the system of an S/G process termination.. The EOL status bit is set if the DMA controller has loaded the last buffer of the Link List. Following PCIRST#, or Master Clear, each bit in this register is reset to "0".

Table 4-45. Scatter/Gather Status Register

Bit	Description
7	**Next Link Null Indicator**. If the next scatter/gather descriptor fetched from memory during a fetch operation has the EOL value set to 1, the current value of the Next Link Register is not overwritten. Instead, bit 7 of the channel's Scatter/Gather Status Register is set to a 1. If the fetch returns a EOL value set to 0, this bit is set to 0. This status bit is written after every fetch operation. Following PCIRST#, or Master Clear, this bit is set to 0. This bit is also cleared by an S/G Start Command write to the Scatter/Gather Command Register.
6	**Reserved**.
5	**Issue IRQ13/EOP on Last Buffer**. When bit 5 = 0, EOP was either defaulted to at reset or selected through the Scatter/Gather Command Register as the S/G process termination indicator. EOP is issued when a terminal count occurs or following the Stop Command. When bit 5 = 1, an IRQ13 is issued to alert the CPU of this same status.
4	**Reserved**.
3	**Scatter/Gather Base Register Status**. When bit 3 = 0, the Base Register is empty. When bit 3 = 1, the Base Register has a buffer link loaded. Note that the Base Register State may be set while the Current Register state is cleared. This condition occurs when the Current Register expires following a transfer. The Base Register will not be moved into the Current Register until the start of the next DMA transfer.
2	**Scatter/Gather Current Register Status**. When bit 2 = 0, the Current Register is empty. When bit 2 = 1, the Current Register has a buffer link loaded and is considered full. Following PCIRST#, bit 2 is set to 0.
1	**Reserved**.
0	**Scatter/Gather Active**. The Scatter-Gather Active bit indicates the current S/G transfer status. Bit 0 is set to a 1 after an S/G Start Command is issued. Bit 0 is set to 0 before the Start Command is issued. Bit 0 is 0 after terminal count on the last buffer on the channel is reached. Bit 0 is also 0 after an S/G Stop Command has been issued. Following a PCIRST# or Master Clear, this bit is 0.

4.2.17 SCATTER/GATHER DESCRIPTOR TABLE POINTER REGISTER

Register Name: Scatter/Gather Descriptor Table Pointer

Register Location:
Channel 0 default address - 0420h-0423h
Channel 1 default address - 0424h-0427h
Channel 2 default address - 0428h-042Bh
Channel 3 default address - 042Ch-042Fh
Channel 5 default address - 0434h-0437h
Channel 6 default address - 0438h-043Bh
Channel 7 default address - 043Ch-043Fh

Default Value: All bits undefined

Attribute: Read/Write, Relocatable

Size: 32 bits

NOTE:
The Base Registers and Current Registers described in the Scatter/Gather sections are hidden registers that are not accessible by software.

The Scatter/Gather Descriptor Table Pointer Register contains the 32-bit pointer address to the first scatter/gather descriptor entry in the descriptor table in memory. Before the start of a S/G transfer, this register should be programmed to point to the first descriptor in the Scatter/Gather Descriptor Table. Following a S/G Start command, the SIO reads the first SGD entry. Subsequently, at the end of the each buffer block transfer, the contents of the SGD Table pointer registers are incremented by 8 until the end of the SGD Table is reached.

The Scatter/Gather Descriptor Table Pointer Registers can be programmed with a single 32-bit PCI write.

Following a prefetch to the address pointed to by the channel's Scatter/Gather Descriptor Table Pointer Register, the new memory address is loaded into the Base Address Register, the new Byte Count is loaded into the Base Byte Count Register, and the newly fetched next scatter/gather descriptor replaces the current next scatter/gather value.

The end of the Scatter/Gather Descriptor Table is indicated by an End of Table field having a MSB equal to 1. When this value is read during a scatter/gather descriptor fetch, the current scatter/gather descriptor value is not replaced. Instead, bit 7 of the channel's Status Register is set to a 1, when the EOL is read from memory.

Table 4-46. Scatter/Gather Descriptor Table Pointer Register

Bit	Description
31:0	The Scatter/Gather Descriptor Table Pointer Register contains a 32-bit pointer address to the main memory location where the software maintains the Scatter Gather Descriptors for the linked-list buffers. Bits [31:0] correspond to A[31:0] on the PCI.

4.2.18 Scatter/Gather Interrupt Status Register

Register Name: Scatter Gather Interrupt Status Register
Register Location: 040Ah
Default Value: 00h
Attribute: Read Only, Relocatable
Size: 8 bits

NOTE:
The Base Registers and Current Registers described in the Scatter/Gather sections are hidden registers that are not accessible by software.

The Scatter/Gather Interrupt Status Register is a read only register and is used to indicate the source (channel) of a DMA Scatter/Gather interrupt on IRQ13. The DMA controller drives IRQ13 active after reaching terminal count during a Scatter/Gather transfer. It does not drive IRQ13 active during the initial programming sequence that loads the Base registers.

4.3 Timer Register Description

The SIO contains three counters that are equivalent to those found in the 82C54 Programmable Interval Timer. The Timer registers control these counters and can be accessed from either the ISA Bus via ISA I/O space or the PCI Bus via PCI I/O space.

This section describes the counter/timer registers on the SIO. The counter/timer operations are further described in Section 5.7, Timer Unit.

4.3.1 TCW - TIMER CONTROL WORD REGISTER

Register Name: Timer Control Word
Register Location: 043h
Default Value: All bits undefined
Attribute: Write Only
Size: 8 bits

The Timer Control Word Register specifies the counter selection, the operating mode, the counter byte programming order and size of the count value, and whether the counter counts down in a 16-bit or binary-coded decimal (BCD) format. After writing the control word, a new count can be written at any time. The new value will take effect according to the programmed mode.

There are six programmable counting modes. Typically, the SIO Timer Counters 0 and 2 are programmed for Mode 3, the Square Wave Mode, while Counter 1 is programmed in Mode 2, the Rate Generator Mode.

Two special commands are selected through the Timer Control Word Register. The Read Back Command (see Section 4.3.1.1) is selected when bits[7:6] are both 1 and the Counter Latch Command (see Section 4.3.1.2) is selected when bits[5:4] are both 0. When either of these two commands are selected, the meaning of the other bits in the register changes.

Table 4-47. Scatter/Gather Interrupt Status Register

Bit	Description Number
7	**Channel 7 Interrupt Status:** When this bit is set to a 1, Channel 7 has an interrupt due to a Scatter/Gather Transfer; otherwsie this bit is set to a 0.
6	**Channel 6 Interrupt Status:** When this bit is set to a 1, Channel 6 has an interrupt due to a Scatter/Gather Transfer; otherwsie this bit is set to a 0.
5	**Channel 5 Interrupt Status:** When this bit is set to a 1, Channel 5 has an interrupt due to a Scatter/Gather Transfer; otherwsie this bit is set to a 0.
4	**Reserved:** Read as 0.
3	**Channel 3 Interrupt Status:** When this bit is set to a 1, Channel 3 has an interrupt due to a Scatter/Gather Transfer; otherwsie this bit is set to a 0.
2	**Channel 2 Interrupt Status:** When this bit is set to a 1, Channel 2 has an interrupt due to a Scatter/Gather Transfer; otherwsie this bit is set to a 0.
1	**Channel 1 Interrupt Status:** When this bit is set to a 1, Channel 1 has an interrupt due to a Scatter/Gather Transfer; otherwsie this bit is set to a 0.
0	**Channel 0 Interrupt Status:** When this bit is set to a 1, Channel 0 has an interrupt due to a Scatter/Gather Transfer; otherwsie this bit is set to a 0.

Bits 4 and 5 are also used to select the count register programming mode. The read/write selection chosen with the control word indicates the programming sequence that must follow when initializing the specified counter. If a counter is programmed to read/write two byte counts, note that a program must not transfer control between writing the first and second byte to another routine that also writes into that same counter. Otherwise, the counter will be loaded with an incorrect count. The count must always be completely loaded with both bytes.

Bits 6 and 7 are also used to select the counter for the control word being written.

Following PCIRST#, the control words for each register are undefined. Each timer must be programmed to bring it into a known state. However, each counter OUT signal is set to 0 following PCIRST#. The SPKR output, interrupt controller input IRQ0 (internal), bit 5 of port 061h, and the internally generated refresh request are each set to 0 following PCIRST#.

Table 4-48. Timer Control Word Register

Bit	Description
7:6	**Counter Select.** The Counter Selection bits select the counter the control word acts upon as shown below. The Read Back Command is selected when bits[7:6] are both 1. **Bit 7 6 Function** 0 0 Counter 0 select 0 1 Counter 1 select 1 0 Counter 2 select 1 1 Read Back Command (see Section 4.3.1.1)
5:4	**Read/Write Select.** Bits [5:4] are the read/write control bits. The Counter Latch Command is selected when bits[5:4] are both 0. The read/write options include R/W least significant byte, R/W most significant byte, or R/W the LSB and then the MSB. The actual counter programming is done through the counter I/O port (040h, 041h, and 042h for counters 0, 1, and 2, respectively). **Bit 5 4 Function** 0 0 Counter Latch Command (see Section 4.3.1.2) 0 1 R/W Least Significant Byte (LSB) 1 0 R/W Most Significant Byte (MSB) 1 1 R/W LSB then MSB
3:1	**Counter Mode Selection.** Bits [3:1] select one of six possible modes of operation for the counter as shown below. **Bit 3 2 1 Mode Function** 0 0 0 0 Out signal on end of count (=0) 0 0 1 1 Hardware retriggerable one-shot X 1 0 2 Rate generator (divide by n counter) X 1 1 3 Square wave output 1 0 0 4 Software triggered strobe 1 0 1 5 Hardware triggered strobe
0	**Binary/BCD Countdown Select.** When bit 0 = 0, a binary countdown is used. The largest possible binary count is 2^{16}. When bit 0 = 1, a binary coded decimal (BCD) count is used. The largest BCD count allowed is 10^4.

4.3.1.1 Read Back Command

The Read Back Command is used to determine the count value, programmed mode, and current states of the OUT pin and Null count flag of the selected counter or counters. The Read Back Command is written to the Timer Control Word Register which latches the current states of the above mentioned variables. The value of the counter and its status may then be read by I/O access to the counter address.

Status and/or count may be latched on one, two, or all three of the counters by selecting the counter during the register write. The count latched remains latched until read, regardless of further latch commands. The count must be read before newer latch commands latch a new count. The status latched by the Read Back Command also remains latched until after a read to the counter's I/O port by reading the Counter Access Ports Register. Thus, the status and count are unlatched only after a counter read of the Timer Status Byte Format Register, the Counter Access Ports Register, or the Timer Status Byte Register and Counter Access Ports Register in succession.

Both count and status of the selected counter(s) may be latched simultaneously by setting both bit 5 and bit 4 to 0. This is functionally the same as issuing two consecutive, separate Read Back Commands. As mentioned above, if multiple count and/or status Read Back Commands are issued to the same counter(s) without any intervening reads, all but the first are ignored.

If both count and status of a counter are latched, the first read operation from that counter returns the latched status, regardless of which was latched first. The next one or two reads (depending on whether the counter is programmed for one or two byte counts) returns the latched count. Subsequent reads return an unlatched count.

NOTE:
The Timer Counter Register bit definitions are different during the Read.

Table 4-49. Timer Counter Register Definition For Read Back Command

Bit	Description
7:6	**Read Back Command.** When bits[7:6] are both 1, the Read Back Command is selected during a write to the Timer Control Word Register. As noted above, the normal meanings (mode, countdown, r/w select) of the bits in the control register at I/O address 043h change when the Read Back Command is selected. Following the Read Back Command, I/O reads from the selected counter's I/O addresses produce the current latch status, the current latched count, or both if bits 4 and 5 are both 0.
5	**Latch Count of Selected Counters.** When bit 5 = 1, the current count value of the selected counters will be latched. When bit 4 = 0, the status will not be latched.
4	**Latch Status of Selected Counters.** When bit 4 = 1, the status of the selected counters will be latched. When bit 4 = 0, the status will not be latched. The status byte format is described in Section 4.3.3, Interval Timer Status Byte Format Register.
3	**Counter 2 Select.** When bit 3 = 1, Counter 2 is selected for the latch command selected with bits 4 and 5. When bit 3 = 0, status and/or count will not be latched.
2	**Counter 1 Select.** When bit 2 = 1, Counter 1 is selected for the latch command selected with bits 4 and 5. When bit 2 = 0, status and/or count will not be latched.
1	**Counter 0 Select.** When bit 1 = 1, Counter 0 is selected for the latch command selected with bits 4 and 5. When bit 1 = 0, status and/or count will not be latched.
0	**Reserved.** Must be 0.

4.3.1.2 Counter Latch Command

The Counter Latch Command latches the current count value at the time the command is received. This command is used to insure that the count read from the counter is accurate (particularly when reading a two-byte count). The count value is then read from each counter's count register (via the Counter Ports Access Ports Register). One, two or all three counters may be latched with one Counter Latch Command.

If a Counter is latched once and then, some time later, latched again before the count is read, the second Counter Latch Command is ignored. The count read will be the count at the time the first Counter Latch Command was issued.

The count must be read according to the programmed format. Specifically, if the Counter is programmed for two byte counts, two bytes must be read. The two bytes do not have to be read one right after the other (read, write, or programming operations for other counters may be inserted between the reads).

NOTE:

1. If a counter is programmed to read/write two-byte counts, a program must not transfer control between reading the first and second byte to another routine that also reads from that same counter. Otherwise, an incorrect count will be read. Finish reading the latched two-byte count before transferring control to another routine.
2. The Timer Counter Register bit definitions are different during the Counter Latch Command than for a normal Timer Counter Register write.

Table 4-50. Timer Control Word Register Definition For Counter Latch Command

Bit	Description
7:6	**Counter Selection.** Bits 6 and 7 are used to select the counter for latching. **Bit 7 6 Function** 0 0 latch counter 0 select 0 1 latch counter 1 select 1 0 latch counter 2 select 1 1 Read Back Command select
5:4	**Counter Latch Command.** When bits[5:4] are both 0, the Counter Latch Command is selected during a write to the Timer Control Word Register. As noted above, the normal meanings (mode, countdown, r/w select) of the bits in the control register at I/O address 043h change when the Counter Latch Command is selected. Following the Counter Latch Command, I/O reads from the selected counter's I/O addresses produce the current latched count.
3:0	**Reserved.** Must be 0.

4.3.2 INTERVAL TIMER STATUS BYTE FORMAT REGISTER

Register Name: Interval Timer Status Byte Format

Register Location: Counter 0 - 040h
Counter 1 - 041h
Counter 2 - 042h

Default Value: Bits[6:0] = X, Bit 7 = 0

Attribute: Read Only

Size: 8 bits per counter

Each counter's status byte can be read following an Interval Timer Read Back Command. The Read Back Command is programmed through the Timer Control Word Register. If latch status is chosen (bit 4 = 0, Read Back Command) as a read back option for a given counter, the next read from the counter's Counter Access Ports Register returns the status byte. The status byte returns the countdown type, either BCD or binary; the counter operational mode; the read/write selection status; the Null count, also referred to as the count register status; and the current state of the counter OUT pin.

Table 4-51. Interval Timer Status Byte Format

Bit	Description
7	**Counter OUT Pin State.** When this bit is a 1, the OUT pin of the counter is also a 1. When this bit is a 0, the OUT pin of the counter is also a 0.
6	**Count Register Status.** Null Count, also referred to as the Count Status Register, indicates when the last count written to the Count Register (CR) has been loaded into the counting element (CE). The exact time this happens depends on the counter mode, but until the count is loaded into the counting element (CE), it can't be read from the counter. If the count is latched or read before the load time, the count value returned will not reflect the new count written to the register. When bit 6 = 0, the count has been transferred from CR to CE and is available for reading. When bit 6 = 1, the Null count condition exists. The count has not been transferred from CR to CE and is not yet available for reading.
5:4	**Read/Write Selection Status.** Bits[5:4] reflect the read/write selection made through bits[5:4] of the control register. The binary codes returned during the status read match the codes used to program the counter read/write selection. **Bit 5 4 Function** 0 0 Counter Latch Command 0 1 R/W Least Significant Byte (LSB) 1 0 R/W Most Significant Byte (MSB) 1 1 R/W LSB then MSB
3:1	**Mode Selection Status.** Bits[3:1] return the counter mode programming. The binary code returned matches the code used to program the counter mode, as listed under the bit function above. **Bit 3 2 1 Mode Selected** 0 0 0 0 0 0 1 1 X 1 0 2 X 1 1 3 1 0 0 4 1 0 1 5
0	**Countdown Type Status.** Bit reflects the current countdown type; ether 0 for binary countdown or a 1 for binary coded decimal (BCD) countdown.

4.3.3 COUNTER ACCESS PORTS REGISTER

Register Name: Counter Access Ports

Register Location: Counter 0, System Timer - 040h
Counter 1, Refresh Request - 041h
Counter 2, Speaker Tone - 042h

Default Value: All bits undefined

Attribute: Read/Write

Size: 8 bits per counter

Each of these I/O ports is used for writing count values to the Count Registers; reading the current count value from the counter by either an I/O read, after a counter-latch command, or after a Read Back Command; and reading the status byte following a Read Back Command.

Table 4-52. Counter Access Ports register

Bit	Description
7:0	**Counter Port bit[x]**. Each counter I/O port address is used to program the 16-bit Count Register. The order of programming, either LSB only, MSB only, or LSB then MSB, is defined with the Interval Counter Control Register at I/O port address 043h. The counter I/O port is also used to read the current count from the Count Register, and return the status of the counter programming following a Read Back Command.

4.3.4 BIOS TIMER REGISTER

Register Name: BIOS Timer

Register Location: Default = 78h - 7Bh (dword aligned)

Default Value: 0000xxxxh

Attribute: Read/Write, Programmable

Size: 32 bit

A write to the BIOS Timer initiates a counting sequence. The timer can be initiated by writing either a 16-bit data portion or the entire 32-bit register (the upper 16 bits are don't cares). Bits[15:0] can be written with the initial count value to start the timer or read to check the current count value. It is the programmer's responsibility to ensure that all 16 bits are written at the same time. After data is written into BIOS timer, the timer will start decrementing until it reaches zero. It will "freeze" at zero until the new count value is written.

The BIOS Timer consists of a single 32-bit register mapped in the I/O space on the location determined by the value written into the BIOS Timer Base Address Register. Bit 0 of the BIOS Timer Base Address Register enables/disables accesses to the BIOS Timer and must be 1 to enable access to the BIOS Timer Register. When the BIOS Timer is enabled, PCI accesses to the BIOS Timer Register do not flow through to the ISA Bus. If the BIOS Timer is disabled, accesses to the addresses assigned to the BIOS Timer Register flow through to the ISA bus. Note, however, that the counter continues to count normally.

Table 4-53. BIOS Timer Register

Bit	Description
31:16	**Reserved**. Read as 0.
15:0	**Timer count value**.

4.4 Interrupt Controller Register Description

The SIO contains an ISA compatible interrupt controller that incorporates the functionality of two 82C59 interrupt controllers. The interrupt registers control the operation of the interrupt controller and can be accessed from the PCI Bus via PCI I/O space. In addition, some of the registers can be accessed from the ISA Bus via ISA I/O space. The bus access for each register is listed in Table 4-2.

4.4.1 ICW1 - INITIALIZATION COMMAND WORD 1 REGISTER

Register Name: Initialization Command Word 1

Register Location: INT CNTRL-1 - 020h
INT CNTRL-2 - 0A0h

Default Value: All bits undefined

Attribute: Write Only

Size: 8 bits per controller

A write to Initialization Command Word 1 starts the interrupt controller initialization sequence. Addresses 020h and 0A0h are referred to as the base addresses of CNTRL-1 and CNTRL-2, respectively.

An I/O write to the CNTRL-1 or CNTRL-2 base address with bit 4 equal to 1 is interpreted as ICW1. For SIO-based ISA systems, three I/O writes to "base address + 1" must follow the ICW1. The first write to "base address + 1" performs ICW2, the second write performs ICW3, and the third write performs ICW4.

ICW1 starts the initialization sequence during which the following automatically occur:

a. The edge sense circuit is reset. This means that following initialization, an interrupt request (IRQ) input must make a low-to-high transition to generate an interrupt.
b. The Interrupt Mask register is cleared.
c. IRQ7 input is assigned priority 7.
d. The slave mode address is set to 7.
e. Special Mask Mode is cleared and Status Read is set to IRR.
f. If IC4 was set to 0, then all functions selected by ICW4 are set to 0. However, ICW4 must be programmed in the SIO implementation of this interrupt controller, and IC4 must be set to a 1.

ICW1 has three significant functions within the SIO interrupt controller configuration. ICW4 is needed, so bit 0 must be programmed to a 1. There are two interrupt controllers in the system, so bit 1, SNGL, must be programmed to a 0 on both CNTRL-1 and CNTRL-2, to indicate a cascade configuration. LTIM, the interrupt controller IRQ edge/level detection control bit, defines the IRQ sensing mode for each controller. When bit 3 is a 0, each IRQ line on the selected controller is programmed for edge-triggered mode. This mode is signified by a low-to-high transition on an IRQ input line. When bit 3 is a 1, the controller is programmed in level-triggered mode, where a high level on an IRQ input indicates the presence of an interrupt request. LTIM is global for each controller. The incoming IRQs are either all edge-triggered or all level-triggered. Bit D4 must be a 1 when programming ICW1. OCW2 and OCW3 are also addressed at the same port as ICW1. This bit indicates that ICW1, and not OCW2 or OCW3, will be programmed during the write to this port.

Bit 2, ADI, and bits [7:5], A7-A5, are specific to an MSC-85 implementation. These bits are not used by the SIO interrupt controllers. Bits [7:5,2] should each be initialized to 0.

Table 4-54. Initialization Command Word 1 Register

Bit	Description
7:5	**ICW/OCW select.** A7-A5 are MCS-85 implementation specific bits. They are not needed by the SIO. These bits should be 000 when programming the SIO.
4	**ICW/OCW select.** Bit 4 must be a 1 to select ICW1. After the fixed initialization sequence to ICW1, ICW2, ICW3, and ICW4, the controller base address is used to write to OCW2 and OCW3. Bit 4 is a 0 on writes to these registers. A 1 on this bit at any time will force the interrupt controller to interpret the write as an ICW1. The controller will then expect to see ICW2, ICW3, and ICW4.
3	**LTIM (Edge/Level Bank Select).** Bit 3, LTIM selects the IRQ assertion detect mode for each controller. Edge-triggered mode, when bit 3 is a 0, signifies that a low-to-high transition on an IRQ line indicates the presence of a valid interrupt request. When bit 3 = 1, the controller is programmed for level-triggered mode. A high level on an incoming IRQ line indicates the presence of an interrupt request. LTIM is global for each controller.
2	**ADI.** Ignored for the SIO.
1	**SNGL (Single or Cascade).** SNGL must be programmed to a 0 to indicate that two interrupt controllers are operating in cascade mode on the SIO.
0	**IC4 (ICW4 Write Required).** This bit must be set to a 1. IC4 indicates that ICW4 needs to be programmed. The SIO requires that ICW4 be programmed to indicate that the controllers are operating in an 80x86 type system.

4.4.2 ICW2 - INITIALIZATION COMMAND WORD 2 REGISTER

Register Name: Initialization Command Word 2
Register Location: INT CNTRL-1 - 021h
INT CNTRL-2 - 0A1h
Default Value: All bits undefined
Attribute: Write Only
Size: 8 bits per controller

ICW2 is used to initialize the interrupt controller with the five most significant bits of the interrupt vector address. The value programmed for bits[7:3] is used by the Host CPU to define the base address in the interrupt vector table for the interrupt routines associated with each IRQ on the controller. Typical ISA ICW2 values are 04h for CNTRL-1 and 70h for CNTRL-2.

4.4.3 ICW3 - INITIALIZATION COMMAND WORD 3 REGISTER

Register Name: Initialization Command Word 3 (Controller 1 - Master Unit)
Register Location: INT CNTRL-1 - 021h
Default Value: All bits undefined
Attribute: Write Only
Size: 8 bits

The meaning of ICW3 differs between CNTRL-1 and CNTRL-2. On CNTRL-1, the master controller, ICW3 indicates which CNTRL-1 IRQ line physically connects the INT output of CNTRL-2 to CNTRL-1. ICW3 must be programmed to 04h, indicating the cascade of the CNTRL-2 INT output to the IRQ[2] input of CNTRL-1.

Table 4-55. Initialization Command Word 2 Register

Bit	Description
7:3	**Interrupt Vector Base Address.** Bits [7:3] define the base address in the interrupt vector table for the interrupt routines associated with each interrupt request level input. For CNTRL-1, a typical value is 00001, and for CNTRL-2, 10000. The interrupt controller combines a binary code representing the interrupt level to receive service with this base address to form the interrupt vector that is driven out onto the bus. For example, the complete interrupt vector for IRQ[0] (CNTRL-1), would be 0000 1000b (CNTRL-1 [7:3] = 00001b and 000b representing IRQ[0]). This vector is used by the CPU to point to the address information that defines the start of the interrupt routine.
2:0	**Interrupt Request Level.** When writing ICW2, these bits should all be 0. During an interrupt acknowledge cycle, these bits are programmed by the interrupt controller with the interrupt code representing the interrupt level to be serviced. This interrupt code is combined with bits [7:3] to form the complete interrupt vector driven onto the data bus during the second INTA# cycle. Section 5.0, Function Description, outlines each of these codes. The code is a simple three bit binary code: 000 represents IRQ0 (IRQ8), 001 IRQ1 (IRQ9), 010 IRQ2 (IRQ10), and so on until 111 IRQ7 (IRQ15).

Table 4-56. Initialization Command Word 3 Register

Bit	Description
7:3	These bits must be programmed to zero.
2	**Cascaded Interrupt Controller IRQ Connection.** Bit 2 must always be programmed to a 1. This bit indicates that CNTRL-2, the slave controller, is cascaded on interrupt request line two (IRQ[2]). When an interrupt request is asserted to CNTRL-2, the IRQ goes through the priority resolver. After the slave controller priority resolution is finished, the INT output of CNTRL-2 is asserted. However, this INT assertion does not go directly to the CPU. Instead, the INT assertion cascades into IRQ[2] on CNTRL-1. IRQ[2] must go through the priority resolution process on CNTRL-1. If it wins the priority resolution on CNTRL-1 and the CNTRL-1 INT signal is asserted to the CPU, the returning interrupt acknowledge cycle is really destined for CNTRL-2. The interrupt was originally requested at CNTRL-2, so the interrupt acknowledge is destined for CNTRL-2, and not a response for IRQ[2] on CNTRL-1. When an interrupt request from IRQ[2] wins the priority arbitration, in reality an interrupt from CNTRL-2 has won the arbitration. Because bit 2 of ICW3 on the master is set to 1, the master knows which identification code to broadcast on the internal cascade lines, alerting the slave controller that it is responsible for driving the interrupt vector during the second INTA# pulse.
1:0	These bits must be programmed to zero.

An interrupt request on IRQ2 causes CNTRL-1 to enable CNTRL-2 to present the interrupt vector address during the second interrupt acknowledge cycle.

4.4.4 ICW3 - INITIALIZATION COMMAND WORD 3 REGISTER

Register Name: Initialization Command Word 3 (Controller 2 - Slave Unit)
Register Location: INT CNTRL-2 - 0A1h
Default Value: All bits undefined
Attribute: Write Only
Size: 8 bits

On CNTRL-2 (the slave controller), ICW3 is the slave identification code broadcast by CNTRL-1 from the trailing edge of the first INTA# pulse to the trailing edge of the second INTA# pulse. CNTRL-2 compares the value programmed in ICW3 with the incoming identification code. The code is broadcast over three SIO internal cascade lines. ICW3 must be programmed to 02h for CNTRL-2. When 010b is broadcast by CNTRL-1 during the INTA# sequence, CNTRL-2 assumes responsibility for broadcasting the interrupt vector during the second interrupt acknowledge cycle.

As an illustration, consider an interrupt request on IRQ[2] of CNTRL-1. By definition, a request on IRQ[2] must have been asserted by CNTRL-2. If IRQ[2] wins the priority resolution on CNTRL-1, the interrupt acknowledge cycle returned by the CPU following the interrupt is destined for CNTRL-2, not CNTRL-1. CNTRL-1 will see the INTA# signal, and knowing that the actual destination is CNTRL-2, will broadcast a slave identification code across the internal cascade lines. CNTRL-2 will compare this incoming value with the 010b stored in ICW3. Following a positive decode of the incoming message from CNTRL-1, CNTRL-2 will drive the appropriate interrupt vector onto the data bus during the second interrupt acknowledge cycle.

Table 4-57. Initialization Command Word 3 Register

Bit	Description
7:3	**Reserved.** Must be 0.
2:0	**Slave Identification Code.** The Slave Identification code must be programmed to 010b during the initialization sequence. The code stored in ICW3 is compared to the incoming slave identification code broadcast by the master controller during interrupt acknowledge cycles.

4.4.5 ICW4 - INITIALIZATION COMMAND WORD 4 REGISTER

Register Name: Initialization Command Word 4
Register Location: INT CNTRL-1 - 021h
INT CNTRL-2 - 0A1h
Default Value: 01h
Attribute: Write Only
Size: 8 bits

Both SIO interrupt controllers must have ICW4 programmed as part of their initialization sequence. Minimally, the microprocessor mode bit, bit 0, must be set to a 1 to indicate to the controller that it is operating in an 80x86 based system. Failure to program this bit will result in improper controller operation during interrupt acknowledge cycles. Additionally, the Automatic End of Interrupt (AEOI) may be selected, as well as the Special Fully Nested Mode (SFNM) of operation.

The default programming for ICW4 is 01h, which selects 80x86 mode, normal EOI, buffered mode, and special fully nested mode disabled.

Bits 2 and 3 must be programmed to 0 for the SIO interrupt controller to function correctly.

Both bit 1, AEOI, and bit 4, SFNM, can be programmed if the system developer chooses to invoke either mode.

4.4.6 OCW1 - OPERATIONAL CONTROL WORD 1 REGISTER

Register Name: Operation Control Word 1
Register Location: INT CNTRL-1 - 021h
INT CNTRL-2 - 0A1h
Default Value: 00h
Attribute: Read/Write
Size: 8 bits

OCW1 sets and clears the mask bits in the Interrupt Mask Register (IMR). Each interrupt request line may be selectively masked or unmasked any time after initialization. A single byte is written to this register. Each bit position in the byte represents the same-numbered channel: bit 0 = IRQ[0], bit 1 = IRQ[1] and so on. Setting the bit to a 1 sets the mask, and clearing the bit to a 0 clears the mask. Note that masking IRQ[2] on CNTRL-1 will also mask all of controller 2's interrupt requests (IRQ8-IRQ15). Reading OCW1 returns the controller's mask register status.

The IMR stores the bits which mask the interrupt lines to be masked. The IMR operates on the IRR. Masking of a higher priority input will not affect the interrupt request lines of lower priority.

Unlike status reads of the ISR and IRR, for reading the IMR, no OCW3 is needed. The output data bus will contain the IMR whenever I/O read is active and the I/O port address is 021h or 0A1h (OCW1).

All writes to OCW1 must occur following the ICW1-ICW4 initialization sequence, since the same I/O ports are used for OCW1, ICW2, ICW3 and ICW4.

Table 4-58. Initialization Command Word 3 Register

Bit	Description
7:5	**Reserved**. Must be 0.
4	**SFNM (Special Fully Nested Mode)**. Bit 4, SFNM, should normally be disabled by writing a 0 to this bit. If SFNM = 1, the special fully nested mode is programmed.
3	**BUF (Buffered mode)**. Bit 3, BUF, must be programmed to 0 for the SIO. This is non-buffered mode. As illustrated above under Bit functionality, different programming options are offered for bits 2 and 3. However, within the SIO interrupt unit, bits 2 and 3 must always be programmed to 00b.
2	**Master/Slave in Buffered Mode**. This bit is not used by the SIO interrupt unit. Bit 2 should always be programmed to 0.
1	**AEOI (Automatic End of Interrupt)**. This bit should normally be programmed to 0. This is the normal end of interrupt. If this bit is 1, the automatic end of interrupt mode is programmed. AEOI is discussed in Section 16.10.2.
0	**Microprocessor Mode**. The Microprocessor Mode bit must be programmed to 1 to indicate that the interrupt controller is operating in an 80x86-based system. Never program this bit to 0.

Table 4-59. Operation Control Word 1 Register

Bit	Description
7:0	**Interrupt Request Mask (Mask [7:0])**. When a 1 is written to any bit in this register, the corresponding IRQ[x] line is masked. For example, if bit 4 is set to a 1, then IRQ[4] will be masked. Interrupt requests on IRQ[4] will not set channel 4's interrupt request register (IRR) bit as long is the channel is masked. When a 0 is written to any bit in this register, the corresponding IRQ[x] mask bit is cleared, and interrupt requests will again be accepted by the controller. Note that masking IRQ[2] on CNTRL-1 will also mask the interrupt requests from CNTRL-2, which is physically cascaded to IRQ[2].

4.4.7 OCW2 - OPERATIONAL CONTROL WORD 2 REGISTER

Register Name: Operation Control Word 2

Register Location: INT CNTRL-1 - 020h
INT CNTRL-2 - 0A0h

Default Value: Bit[4:0] = undefined,
Bit[7:5] = 001

Attribute: Write Only

Size: 8 bits

OCW2 controls both the Rotate Mode and the End of Interrupt Mode, and combinations of the two. The three high order bits in an OCW2 write represent the encoded command. The three low order bits are used to select individual interrupt channels during three of the seven commands. The three low order bits (labeled L2, L1 and L0) are used when bit 6 is set to a 1 during the command.

Following a PCIRST# and ICW initialization, the controller enters the fully nested mode of operation. Non-specific EOI without rotation is the default. Both rotation mode and specific EOI mode are disabled following initialization.

Table 4-60. Operation Control Word 2

Bit	Description
7:5	**Rotate and EOI Codes.** R, SL, EOI - These three bits control the Rotate and End of Interrupt modes and combinations of the two. A chart of these combinations is listed above under the bit definition. **Bits 7 6 5 Function** 0 0 1 Non-specific EOI command 0 1 1 Specific EOI Command 1 0 1 Rotate on Non-Specific EOI Command 1 0 0 Rotate in Auto EOI Mode (Set) 0 0 0 Rotate in Auto EOI Mode (Clear) 1 1 1 *Rotate on Specific EOI Command 1 1 0 *Set Priority Command 0 1 0 No Operation * L0 - L2 Are Used Section 16.10 and 16.11 provide detailed information on both the EOI and Rotate mode functionality.
4:3	**OCW2 Select.** When selecting OCW2, bits 3 and 4 must both be 0. If bit 4 is a 1, the interrupt controller interprets the write to this port as an ICW1. Therefore, always ensure that these bits are both 0 when writing an OCW2.
2:0	**Interrupt Level Select (L2, L1, L0).** L2, L1, and L0 determine the interrupt level acted upon when the SL bit is active. A simple binary code, outlined above, selects the channel for the command to act upon. When the SL bit is inactive, these bits do not have a defined function; programming L2, L1 and L0 to 0 is sufficient in this case. **Bit 2 1 0 Interrupt Level** 0 0 0 IRQ 0(8) 0 0 1 IRQ 1(9) 0 1 0 IRQ 2(10) 0 1 1 IRQ 3(11) 1 0 0 IRQ 4(12) 1 0 1 IRQ 5(13) 1 1 0 IRQ 6(14) 1 1 1 IRQ 7(15)

4.4.8 OCW3 - OPERATIONAL CONTROL WORD 3 REGISTER

Register Name: Operation Control Word 3

Register Location: INT CNTRL-1 - 020h
INT CNTRL-2 - 0A0h

Default Value: Bit[6,0] = 0,
Bit[7,4:2] = undefined,
Bit[5,1] = 1

Attribute: Read/Write

Size: 8 bits

OCW3 serves three important functions: Enable Special Mask Mode, Poll Mode control, and IRR/ISR register read control.

First, OCW3 is used to set or reset the Special Mask Mode (SMM). The Special Mask Mode can be used by an interrupt service routine to dynamically alter the system priority structure while the routine is executing, through selective enabling/disabling of the other channel's mask bits.

Second, the Poll Mode is enabled when a write to OCW3 is issued with bit 2 equal to 1. The next I/O read to the interrupt controller is treated like an interrupt acknowledge; a binary code representing the highest priority level interrupt request is released onto the bus.

Third, OCW3 provides control for reading the In-Service Register (ISR) and the Interrupt Request Register (IRR). Either the ISR or IRR is selected for reading with a write to OCW3. Bits 0 and 1 carry the encoded command to select either register. The next I/O read to the OCW3 port address will return the register status specified during the previous write. The register specified for a status read is retained by the interrupt controller. Therefore, a write to OCW3 prior to every status read command is unnecessary, provided the status read desired is from the register selected with the last OCW3 write.

Table 4-61. Operation Control Word 3 Register

Bit	Description
7	**Reserved**. Must be 0.
6	**SMM (Special Mask Mode)**. If ESMM = 1 and SMM = 1 the interrupt controller enters Special Mask Mode. If ESMM = 1 and SMM = 0, the interrupt controller is in normal mask mode. When ESMM = 0, SMM has no effect.
5	**ESMM (Enable Special Mask Mode)**. When ESMM = 1, the SMM bit is enabled to set or reset the Special Mask Mode. When ESMM = 0, the SMM bit becomes a "don't care".
4:3	**OCW3 Select**. When selecting OCW3, bit 3 must be a 1 and bit 4 must be 0. If bit 4 = 1, the interrupt controller interprets the write to this port as an ICW1. Therefore, always ensure that bits[4:3] = 01 when writing an OCW3.
2	**Poll Mode Command**. When bit 2 = 0, the Poll command is not issued. When bit 2 = 1, the next I/O read to the interrupt controller is treated as an interrupt acknowledge cycle. An encoded byte is driven onto the data bus, representing the highest priority level requesting service.
1:0	**Register Read Command**. Bits [1:0] provide control for reading the In-Service Register (ISR) and the Interrupt Request Register (IRR). When bit 1 = 0, bit 0 will not affect the register read selection. When bit 1 = 1, bit 0 selects the register status returned following an OCW3 read. If bit 0 = 0, the IRR will be read. If bit 0 = 1, the ISR will be read. Following ICW initialization, the default OCW3 port address read will be "read IRR". To retain the current selection (read ISR or read IRR), always write a 0 to bit 1 when programming this register. The selected register can be read repeatedly without reprogramming OCW3. To select a new status register, OCW3 must be reprogrammed prior to attempting the read. **Bit 1 0 Function** 0 0 No Action 0 1 No Action 1 0 Read IRQ Register 1 1 Read IS Register

4.5 Control Registers

This section contains NMI registers, a real-time clock register, Port 92 Register, and the Digital Output Register.

4.5.1 NMISC - NMI STATUS AND CONTROL REGISTER

Register Name: NMI Status and Control
Register Location: 061h
Default Value: 00h
Attribute: Read/Write
Size: 8 bits

This register is used to check the status of different system components, control the output of the speaker counter (Counter 2), and gate the counter output that drives the SPKR signal.

Bits 4, 5, 6, and 7 are read-only. When writing to this port, these bits must be written as 0's. Bit 6 returns the IOCHK# NMI status. This input signal comes from the ISA Bus. It is used for parity errors on memory cards plugged into the bus, and for other high priority interrupts. The current status of bit 3 enables or disables this NMI source. Bit 5 is the current state of the OUT pin of interval Timer 1, Counter 2. Bit 4 toggles from 1-0 or from 0-1 after every Refresh cycle. Following PCIRST#, bits 4 and 6 are both 0. Bit 5 is undetermined until Counter 2 is properly programmed. Bit 7 returns the PCI System Error status (SERR#). If 0, bit 7 indicates that SERR# was not pulsed active by a PCI agent. If 1, bit 7 indicates that SERR# was pulsed active by a PCI agent and that an NMI will be issued to the Host CPU. This NMI can be disabled with bit 2 of this register.

Bits 0-3 are both read and write. Bit 0 is the GATE input signal for Timer 1, Counter 2. The GATE input is used to disable counting in Counter 2. The Counter 2 output is ANDed with bit 1 to form the SPKR output signal. Bit 1 gates the Counter 2 OUT value. When bit 1 is disabled, the SPKR signal is disabled; when bit 1 is enabled, the SPKR output follows the value at the OUT pin of Counter 2. The Counter 2 OUT pin status can be checked by reading port 061h and checking bit 5. Bit 2 is used to enable the System Error (SERR#) signal. Bit 3 enables or disables the incoming IOCHK# NMI signal from the expansion bus. Each of these bits is reset to 0 following PCIRST#.

Table 4-62. NMI Status and Control Register

Bit	Description
7	**SERR# Status.** Bit 7 is set if a system board agent (PCI devices or main memory) detects a system board error and pulses the PCI SERR# line. This interrupt is enabled by setting bit 2 to 0. To reset the interrupt, set bit 2 to 0 and then set it to 1. This bit is read-only. When writing to port 061h, bit 6 must be a 0.
6	**IOCHK# NMI Source Status.** Bit 6 is set if an expansion board asserts IOCHK# on the ISA/SIO bus. This interrupt is enabled by setting bit 3 to 0. To reset the interrupt, set bit 3 to 0 and then set it to 1. This bit is read-only. When writing to port 061h, bit 6 must be a 0.
5	**Timer Counter 2 OUT Status.** The Counter 2 OUT signal state is reflected in bit 5. The value on this bit following a read is the current state of the Counter 2 OUT signal. Counter 2 must be programmed following a PCIRST# for this bit to have a determinate value. Bit 5 is read-only. When writing to port 061h, bit 5 must be a 0.
4	**Refresh Cycle Toggle.** The Refresh Cycle Toggle signal toggles from either 0 to 1 or 1 to 0 following every refresh cycle. This read-only bit is a 0 following PCIRST#. When writing to port 061h, bit 4 must be a 0.
3	**IOCHK# NMI Enable.** When bit 3 = 1, IOCHK# NMI's are disabled and cleared. When bit 3 = 0, IOCHK# NMI's are enabled. Following PCIRST#, bit 3 is reset to 0.
2	**PCI SERR# Enable.** When bit 2 = 1, the PCI System Error (SERR#) is disabled and cleared. When bit 2 = 0, SERR# is enabled. Following PCIRST#, bit 2 is a 0.
1	**Speaker Data Enable.** Speaker Data Enable is ANDed with the Counter 2 OUT signal to drive the SPKR output signal. When bit 1 = 0, the result of the AND is always 0 and the SPKR output is always 0. When bit 1 = 1, the SPKR output is equivalent to the Counter 2 OUT signal value. Following PCIRST#, bit 1 is a 0.
0	**Timer Counter 2 Enable.** When bit 0 = 0, Counter 2 counting is disabled. Counting is enabled when bit 0 = 1. This bit controls the GATE input to Counter 2. Following PCIRST#, the value of this bit is 0.

4.5.2 NMI ENABLE AND REAL-TIME CLOCK ADDRESS REGISTER

Register Name: NMI Enable and Real-Time Clock Address
Register Location: 070h
Default Value: Bit[6:0] = undefined, Bit 7 = 1
Attribute: Write Only
Size: 8 bits

The Mask register for the NMI interrupt is at I/O address 070h shown below. The most-significant bit enables or disables all NMI sources including IOCHK# and the NMI Port. Write an 80h to port 70h to mask the NMI signal. This port is shared with the real-time clock. The real-time-clock uses the lower six bits of this port to address memory locations. Writing to port 70h sets both the enable/disable bit and the memory address pointer. Do not modify the contents of this register without considering the effects on the state of the other bits. Reads and writes to this register address flow through to the ISA Bus.

Table 4-63. NMI Enable and Real-Time Clock Address Register

Bit	Description
7	**NMI Enable.** Setting bit 7 to a 1 disables all NMI sources. Setting the bit to a 0 enables the NMI interrupt. Following PCIRST#, this bit is a 1.
6:0	**Real Time Clock Address.** Used by the Real Time Clock on the Base I/O component to address memory locations. Not used for NMI enabling/disabling.

4.5.3 PORT 92 REGISTER

Register Name: Port 92
Register Location: 92h
Default Value: 24h
Attribute: Read/Write
Size: 8 bits

This register is used to support the alternate reset (ALT_RST#) and alternate A20 (ALT_A20) functions. This register is only accessible if bit 6 in the Utility Bus Chip Select B Register is set to a 1. Reads and writes to this register location flow through to the ISA Bus.

Table 4-64. Port 92 Register

Bit	Description
7:6	**Reserved.** Returns 00 when read.
5	**Reserved.** Returns a 1 when read.
4	**Reserved.** Returns a 0 when read.
3	**Reserved.** Returns a 0 when read.
2	**Reserved.** Return a 1 when read.
1	**ALT_A20 Signal Control.** Writing a 0 to this bit causes the ALT_A20 signal to be driven low. Writing a 1 to this bit causes the ALT_A20 signal to be driven high.
0	**Alternate System Reset.** This read/write bit provides an alternate system reset function. This function provides an alternate means to reset the system CPU to effect a mode switch from Protected Virtual Address Mode to the Real Address Mode. This provides a faster means of reset than is provided by the Keyboard controller. This bit is set to a 0 by a system reset. Writing a 1 to this bit will cause the ALT_RST# signal to pulse active (low) for approximately 4 SYSCLK's. Before another ALT_RST# pulse can be generated, this bit must be written back to a 0.

4.5.4 DIGITAL OUTPUT REGISTER

Register Name: Digital Output
Register Location: 03F2h (Primary), 0372h (Secondary)
Default Value: Bit[7:4,2:0] = undefined, Bit 3 = 0
Attribute: Write only
Size: 8 bits

This register is used to prevent UBUSOE# from responding to DACK2# during a DMA read access to a floppy controller on the ISA Bus. If a second floppy (residing on the ISA Bus) is using DACK2# in conjunction with a floppy on the utility bus, this prevents the floppy on the utility bus and the utility bus transceiver from responding to an access targeted for the floppy on the ISA Bus. This register is also located in the floppy controller device. Reads and writes to this register location flow through to the ISA Bus.

Table 4-65. Digital Output Register

Bit	Description
7:4	**Not Used.** These bits exist in the floppy controller.
3	**DMA Enable.** When this bit is a 1, the assertion of DACK# will result in UBUSOE# being asserted. If this bit is 0, DACK2# has no effect on UBUSOE#. This port bit also exists on the floppy controller. This bit defaults to disable (0).
2:0	**Not Used.** These bits exist in the floppy controller.

4.5.5 RESET UBUS IRQ12 REGISTER

Register Name: Reset Ubus IRQ12
Register Location: 60h
Default Value: N/A
Attribute: Read only
Size: 8 bits

This address location (60h) is used to clear the mouse interrupt function to the CPU. Reads to this address are monitored by the SIO. When the mouse interrupt function is enabled (bit 4 of the ISA Clock Divisor Register is 1), the mouse interrupt function is provided on the IRQ12/M input signal. In this mode, a mouse interrupt generates an interrupt through IRQ13 to the Host CPU. A read of 60h releases IRQ12. If bit 4 = 0 in the ISA Clock Divisor Register, a read of address 60h has no effect on IRQ12/M. Reads and writes to this register flow through to the ISA Bus. For additional information, see the IRQ12/M description in Section 3.0, Signal Description.

Table 4-66. Reset Ubus IRQ12 Register

Bit	Description
7:0	**Reset IRQ12.** No specific pattern. A read of address 60h executes the command.

4.5.6 COPROCESSOR ERROR REGISTER

Register Name: Reset Coprocessor Error
Register Location: F0h
Default Value: N/A
Attribute: Write only
Size: 8 bits

This address location (F0h) is used when the SIO is programmed for coprocessor error reporting (bit 5 of the ISA Clock Divisor Register is 1). Writes to this address are monitored by the SIO. In this mode, the SIO generates an interrupt (INT) to the CPU when it receives an error signal (FERR# asserted) from the CPU's coprocessor. Writing address F0h, when FERR# is asserted, causes the SIO to assert IGNNE# and negate IRQ13. IGNNE# remains asserted until FERR# is negated. If FERR# is not asserted, writing to address F0h does not effect IGNNE#. Reads and writes to this register flow through to the ISA Bus. For additional information, see the IGNNE# description in Section 3.0, Signal Description.

Table 4-67. Coprocessor Error Register

Bit	Description
7:0	**Reset IRQ12.** No specific pattern. A write to address F0h executes the command.

5.0 DETAILED FUNCTIONAL DESCRIPTION

5.1 PCI Interface

5.1.1 PCI COMMAND SET

Bus commands indicate to the slave the type of transaction the master is requesting. Bus Commands are encoded on the C/BE[3:0]# lines during the address phase of a PCI cycle.

5.1.2 PCI BUS TRANSFER BASICS

Details of PCI Bus operations can be found in the *Peripheral Component Interconnect (PCI) Specification*. Only details of the PCI Bus unique to the SIO are included in this data sheet.

5.1.2.1 PCI Addressing

PCI address decoding uses the AD[31:0] signals. AD[31:2] are always used for address decoding while the information contained in the two low order bits AD[1:0] varies for memory, I/O, and configuration cycles.

For I/O cycles, AD[31:0] are decoded to provide a byte address. AD[1:0] are used for generation of DEVSEL# only and indicate the least significant valid byte involved in the transfer. For example, if only BE0# is asserted, AD[1:0] are 00. If only BE3# is asserted, then AD[1:0] are 11. If BE3# and BE2# are asserted, AD[1:0] are 10. If all BEx#'s are asserted, then AD[1:0] are 00. The byte enables determine which byte lanes contain valid data. The SIO requires that PCI accesses to byte-wide internal registers must assert only one byte enable.

When the SIO is the target of any PCI transaction in which BE[3:0]# = 1111, the SIO terminates the cycle normally by asserting TRDY#. No data is written into the SIO during write cycles and the data driven by the SIO during read cycles is indeterminate.

For memory cycles, accesses are decoded as dword accesses. This means that AD[1:0] are ignored for decoding memory cycles. The byte enables determine which byte lanes contain valid data. When the SIO is a PCI master, it drives 00 on AD[1:0] for all memory cycles.

Table 5-1. PCI Commands

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

NOTES:
1) Treated as Memory Write
2) Treated as Memory Read
3) Reserved and Special Cycles are considered invalid by the SIO and are be completely ignored. All internal address decoding is ignored and DEVSEL# is never be asserted.

For configuration cycles, DEVSEL# is a function of IDSEL and AD[1:0]. DEVSEL# is selected during a configuration cycle only if IDSEL is active and both AD[1:0] = 00. The cycle is ignored by the SIO if either AD1 or AD0 is non-zero. Configuration registers are selected as dwords using AD[7:2]. The byte enables determine which byte lanes contain valid data.

5.1.2.2 DEVSEL# Generation

As a PCI slave, the SIO asserts the DEVSEL# signal to indicate it is the slave of the PCI transaction. DEVSEL# is asserted when the SIO positively or subtractively decodes the PCI transaction. The SIO asserts DEVSEL# (claim the transaction) before it issues any other slave response, i.e., TRDY#, STOP#, etc. After the SIO asserts DEVSEL# (refer to Figures 5-2 and 5-4), it does not negate DEVSEL# until the same edge that the master uses to negate the final IRDY#.

It is expected that most (perhaps all) PCI target devices will be able to complete a decode and assert DEVSEL# within 1 or 2 clocks of FRAME#. Since the SIO subtractively decodes all unclaimed PCI cycles (except configuration cycles), it provides a configuration option to pull in (by 1 or 2 clocks) the edge when the SIO samples DEVSEL#. This allows faster access to the expansion bus. Use of such an option is limited by the slowest positive decode agent on the bus. This is described in more detail in Section 5.5.1.4, Subtractively Decoded Cycles to ISA.

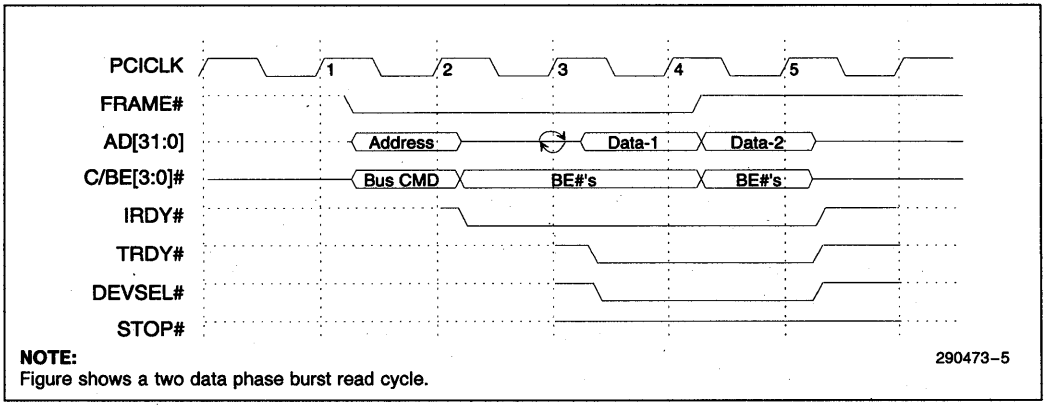
As a PCI master, the SIO waits for 5 PCICLKs after the assertion of FRAME# for a slave to assert DEVSEL#. If the SIO does not receive DEVSEL# in this time, it will master-abort the cycle. See Section 5.1.3.1, SIO as Master - Master-Initiated Termination, for further details.

5.1.2.3 Basic PCI Read Cycles (I/O and Memory)

Figure 5-1 shows the SIO as a master reading from PCI memory in zero wait states. Figure 5-2 shows the fastest the SIO, as a slave, responds to a read transaction generated by a PCI master.

As a PCI master, the SIO only performs memory read transfers (i.e. I/O read transfers are not supported). When reading data from PCI memory, the SIO requests a maximum of 8 bytes via a two data phase burst read cycle to fill its internal 8 byte line buffer. If the line buffer is programmed for single transaction mode, fewer bytes are requested (refer to section 5.6.1, DMA/ISA Master Line Buffer). Read cycles from PCI memory are generated on behalf of ISA masters and DMA devices.

The SIO asserts FRAME# on clock 1. The clock in which FRAME# is first asserted is called the address phase. During the address phase, AD[31:0] contain a valid byte address and C/BE[3:0]# contain a bus command. The clock following the address phase (clock 2) is the beginning of the data phase. During the data phase, the C/BE[3:0]# indicate which byte lanes are involved in the transaction. The SIO drives valid byte enables from the rising edge of clock 2. If the byte lanes involved in the transaction are different for data -1 and data -2, the SIO drives new values on the C/BE[3:0]# lines from the rising edge of clock 4. The C/BE[3:0]# lines remain active until the end of the burst transfer.



NOTE:
Figure shows a two data phase burst read cycle.

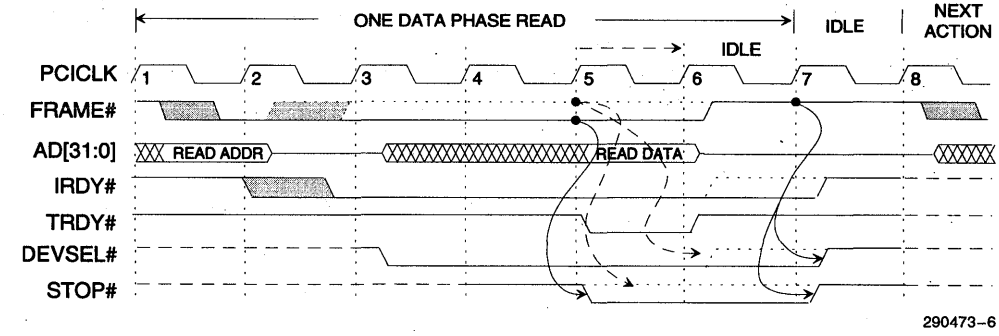
Figure 5-1. SIO Read From PCI Memory (SIO as Master)

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

A single data phase is completed when the SIO samples IRDY# and TRDY# asserted on the same PCICLK rising edge. If wait states are required by the slave, TRDY# can be negated during the rising edge of PCICLK. As a master, the SIO does not add wait states. In Figure 5.1, data is successfully transferred on clocks 4 and 5. The SIO negates FRAME# on clock 4 to indicate that the next clock (clock 5) is the last data phase.

As a PCI slave, the SIO responds to both I/O read and memory read transfers. For multiple read transactions, the SIO always target-terminates after the first data read transaction by asserting STOP# and TRDY# at the end of the first data phase (Figure 5-13). For single read transactions, the SIO finishes the cycle in a normal fashion, by asserting TRDY# without asserting STOP#. Figure 5-2 shows the fastest the SIO responds when a read access to one of its internal configuration registers is detected (*note: some SIO configuration registers require a longer access time, and wait states are added*). During read accesses to the ISA Bus or SIO non-configuration registers, the SIO always adds wait states. The SIO does this by holding TRDY# high until the register is accessed, or in the case of an ISA access, until the transfer on the ISA Bus is complete.

After sampling FRAME# active (assuming the SIO has not forced a retry), the SIO drives the AD lines at the rising edge of the following PCICLK (clock 3, Figure 5-2), even though valid data will not be available. The SIO also drives DEVSEL# active at this time, if a positive decode is done. Otherwise, if a subtractive decode is done, DEVSEL# will not be driven active for two to three PCICLKs after FRAME# is sampled active (refer to Sections 5.1.2.2 and 5.5.1.4). After the SIO has presented valid read data during the first data phase, and FRAME# remains active (multiple transaction indicated), TRDY# and STOP are driven active on the rising edge of PCICLK to target-terminate the transfer (clock 5, Figure 5-2). If a single transaction is indicated (FRAME# sampled inactive during the first data phase), the SIO asserts TRDY# without driving STOP# active. Read cycles are subject to retry. If the SIO is locked, if the cycle triggers buffer management activity, or if the ISA Bus is occupied by an ISA master or the DMA, the read cycle will get retried. If the cycle gets retried due to an occupied ISA Bus, the ISA Bus will be requested.



NOTE:
Dashed lines indicate single read - normal termination.
Sold lines indicate multiple read - SIO target termination.

Figure 5-2. PCI Master Read From SIO - I/O Read (SIO as Slave)

For I/O read cycles, the internal SIO registers are positively decoded, and any unclaimed cycle below 64 KBytes is subtratively decoded and forwarded to the ISA Bus. If the internal SIO registers are the target of a PCI master I/O cycle and more than one byte enable is active, the SIO will respond with a target-abort since the registers must be accessed as 8-bit quantities.

For memory read cycles, BIOS accesses are positively decoded and any unclaimed cycle below 16 Mbytes is subtractively decoded and forwarded to the ISA Bus.

5.1.2.4 Basic PCI Write Cycles (I/O and Memory)

Figure 5-3 shows the SIO as a master writing to PCI memory in zero wait states. Figure 5-4 shows the fastest the SIO will respond, as a slave, to a memory write or I/O write transaction generated by a PCI master.

As a PCI master, the SIO generates a PCI memory write cycle when it decodes an ISA-originated/PCI-bound memory write cycle. I/O write cycles are never initiated by the SIO. When writing data to PCI memory, the SIO writes a maximum of 4 bytes via a single data transaction write cycle. If the SIO's internal ISA master/DMA line buffer is programmed for single transaction mode, fewer bytes will be generated (refer to section 5.6.1, DMA/ISA Master Line Buffer). In either case, only one data transaction will be performed. Cycles to PCI memory are generated on behalf of ISA masters, DMA devices, and the SIO when the SIO needs to flush the ISA master/DMA line buffer.

As a PCI master, the SIO drives the AD0 and AD1 signals low during the address phase of the cycle. This is done to indicate to the slave that the address will increment during the transfer. If there is no response on the PCI Bus, the SIO will master-abort due to the DEVSEL# time out.

As a PCI slave, the SIO will respond to both I/O write and memory write transfers. For multiple write transactions, the SIO will always target-terminate after the first data write transaction by asserting STOP# and TRDY# at the end of the first data phase. For single write transactions, the SIO will finish the cycle normally by asserting TRDY# without asserting STOP#. Figure 5-4 shows the fastest the SIO will respond to a write cycle targeted for an internal configuration register (*note: some of the SIO configuration registers will require a longer access time, and wait states will be added*) or to ISA memory when the internal PCI posted write buffer is enabled. During I/O write accesses to the ISA Bus or SIO non-configuration registers, the SIO will always add wait states. During a memory write access to ISA memory where the SIO posted write buffer is enabled, the SIO will perform the access in a one wait state cycle. If the posted write buffer is disabled, the PCI master will be held, in wait states, until the cycle is completed on the ISA Bus.

For I/O write cycles, the internal SIO registers will be positively decoded, and any unclaimed cycle below 64 KBytes will be subtratively decoded and forwarded to the ISA Bus. If the internal SIO registers are the target of a PCI master I/O cycle and more than one byte enable is active, the SIO will respond with a target-abort since the registers must be accessed as 8-bit quantities.

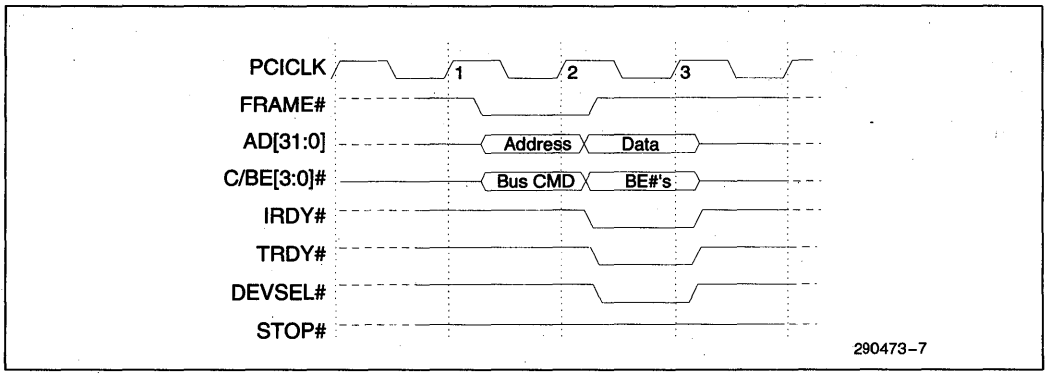
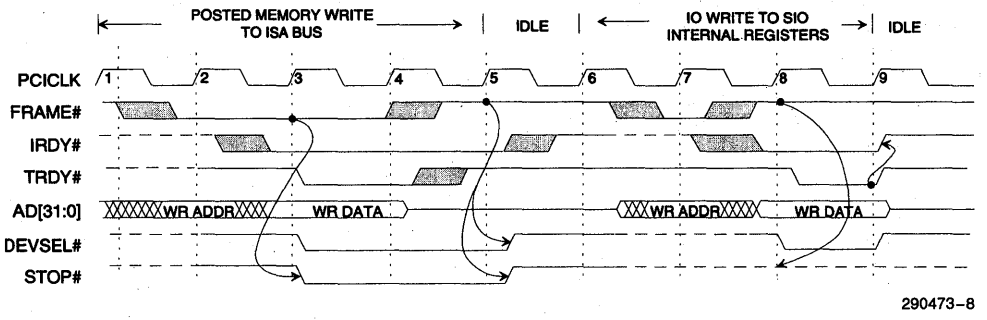


Figure 5-3. SIO Write to PCI Memory (SIO as Master)



NOTE:
Memory write cycle shown is multiple write cycle - SIO target termination.
I/O cycle shown is single write cycle - normal termination.

Figure 5-4. PCI Write to SIO - Posted Write, Register Write (SIO as Slave)

For memory write cycles, BIOS accesses will be positively decoded, and any unclaimed cycle below 16 Mbytes will be subtratively decoded and forwarded to the ISA Bus.

Write cycles are subject to retry. The write cycle will get retried if the SIO is locked or if the cycle triggers buffer management activity. The cycle will also get retried if the PCI Posted Write Buffer is full or disabled (in the case of memory write cycles), or if the ISA Bus is occupied by an ISA master or the DMA.

In the case of no response on the ISA Bus, the SIO will run a standard length ISA write cycle and terminate normally.

5.1.2.5 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 is through the IDSEL configuration mechanism. The SIO is an exception to this since it controls access to the BIOS boot code. All SIO addresses that are enabled after reset are accessible immediately after power up.

The configuration read or write command defined by the bus control signals C/BE[3:0]# is used to configure the SIO. During the address phase of the configuration read or write command, the SIO will sample its IDSEL (ID select). If IDSEL is active and AD[1:0] are both zero, the SIO generates DEVSEL#. Otherwise, the cycle is ignored by the SIO. During the configuration cycle address phase, bits AD[7:2] and C/BE[3:0]# are used to select particular bytes within a configuration register. Reference Figure 5-5 for configuration reads and Figure 5-6 for configuration writes. Note that IDSEL is normally a "don't care" except during the address phase of a transaction. When a configuration cycle is decoded, IDSEL is sampled active, and AD[1:0] = 00, the SIO will respond by asserting DEVSEL# and TRDY#.

Note that an unclaimed configuration cycle is never forwarded to the ISA Bus.

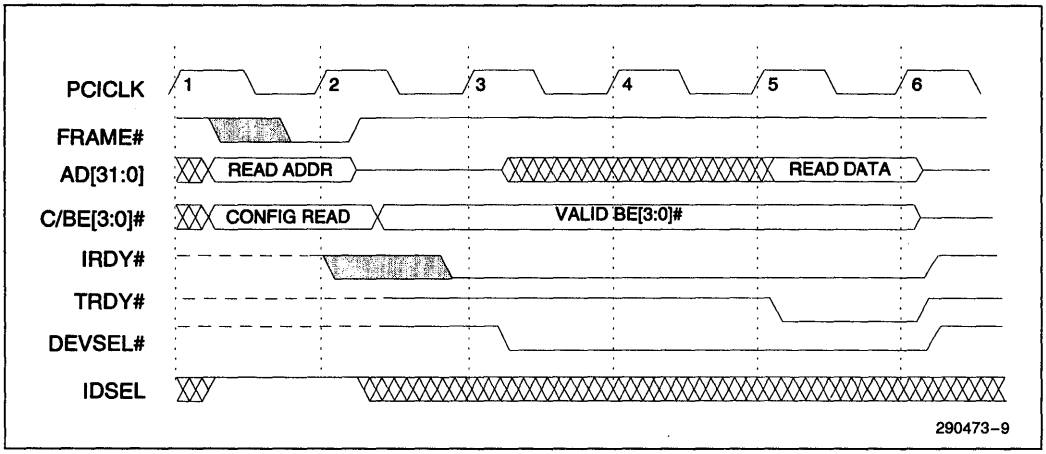


Figure 5-5 Configuration Read Cycle

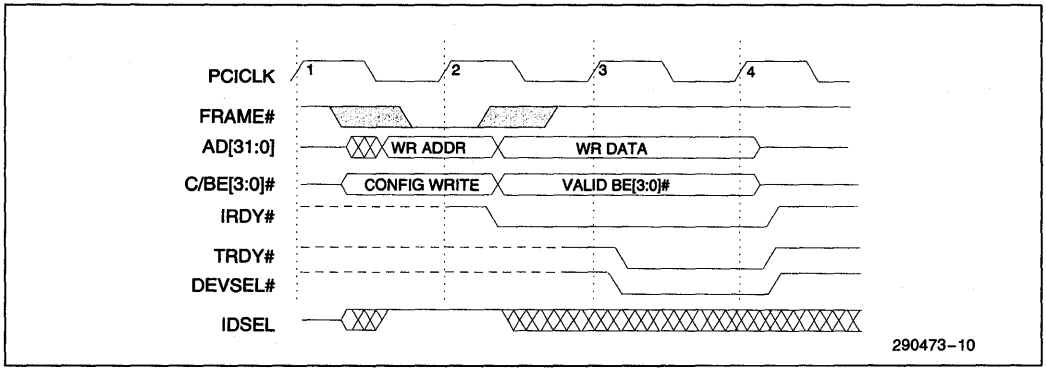


Figure 5-6. Configuration Write Cycle

5.1.2.6 Interrupt Acknowledge Cycle

The interrupt acknowledge command is a single byte read implicitly addressed to the SIO's interrupt controller. The address bits are logical "don't cares" during the address phase and the byte enables will indicate to the SIO an 8-bit interrupt vector is to be returned on AD[7:0]. The SIO converts this single cycle transfer into two cycles that the internal 8259 pair can respond to (see Section 5.8, Interrupt Controller). The SIO will hold the PCI Bus in wait states until the 8 bit interrupt vector is returned.

SIO responses to an interrupt acknowledge cycle can be disabled by setting bit 5 in the PCI Control Register to a 0. However, if disabled, the SIO will still respond to accesses to the interrupt register set and allow poll mode functions.

If the SIO is locked, the interrupt acknowledge cycle triggers buffer management activity, or if the ISA Bus is occupied by an ISA master or the DMA, the interrupt acknowledge cycle will get retried.

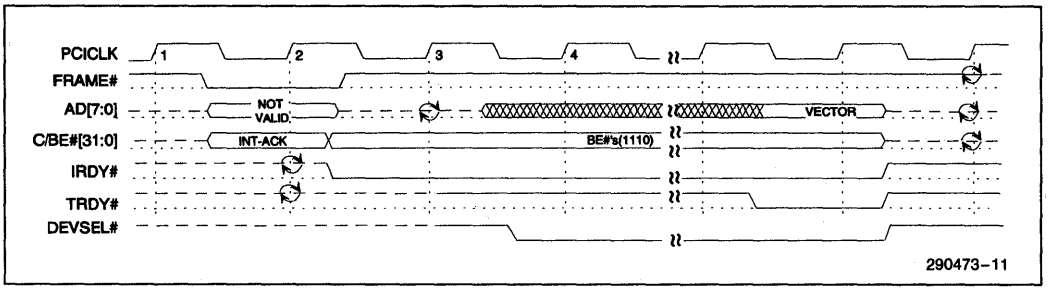


Figure 5-7. Interrupt Acknowledge Cycle

5.1.2.7 Exclusive Access

The SIO, as a resource, can be locked by any PCI master. The advantage of the concept of locking a resource is it allows critical real time accesses to non-locked resources to proceed during a lock. In the context of locked cycles, the entire SIO subsystem is considered a single resource. A locked access to any address contained within the SIO subsystem (including the ISA Bus) locks the entire subsystem. Any subsequent PCI master access to the SIO subsystem, while it is locked, results in a retry.

The SIO marks itself locked anytime it is the slave of the access and LOCK# is sampled negated during the address phase (Figure 5-8). As a locked slave, the SIO responds to a master only when it samples LOCK# negated and FRAME# asserted. The locking master may negate LOCK# at the end of the last data phase (Figure 5-9). The SIO unlocks itself when FRAME# and LOCK# are both negated. The SIO will respond by asserting STOP# with TRDY# negated (retry) to all transactions when LOCK# is asserted during the address phase (refer to Figure 5-10).

Locked cycles are never generated by the SIO.

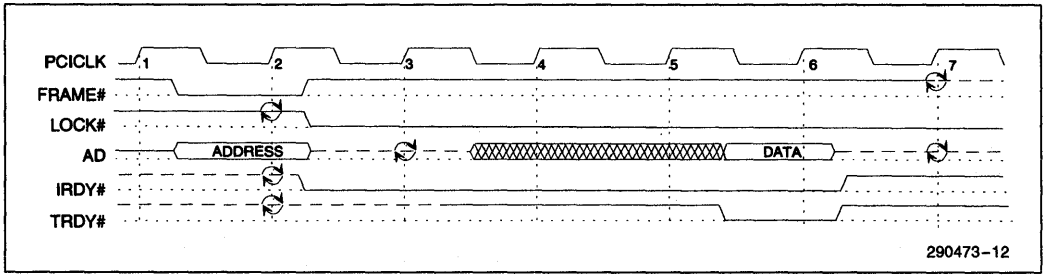


Figure 5-8. Locking the SIO

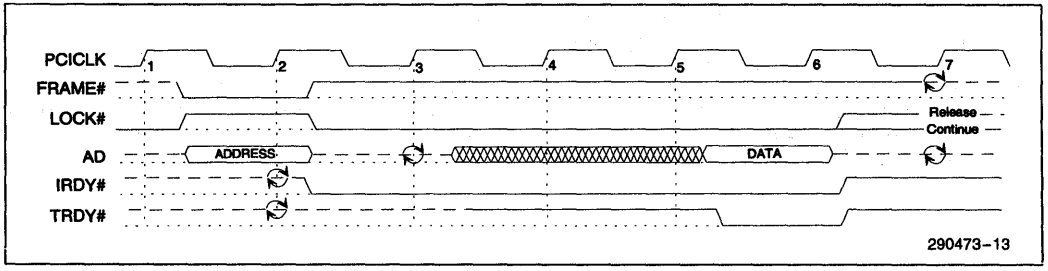


Figure 5-9. Continuing Locked Cycle

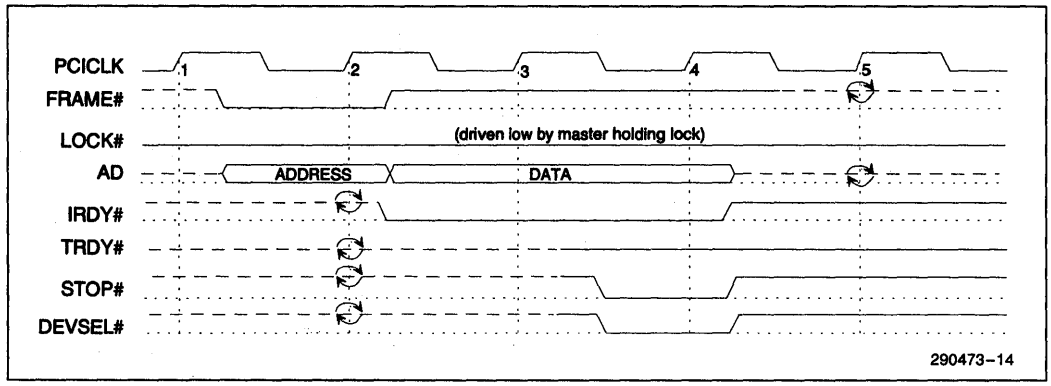


Figure 5-10. Write Access to SIO When SIO is Already Locked

5.1.3 TRANSACTION TERMINATION

The SIO supports both Master-initiated Termination as well as Target-initiated Termination. All transactions are concluded when FRAME# and IRDY# are both sampled negated, indicating the PCI Bus is idle.

5.1.3.1 SIO as Master - Master-Initiated Termination

The SIO supports two forms of master-initiated termination:

1. Normal termination of a completed transaction.
2. Abnormal termination due to no slave responding to the transaction (Abort).

A normal termination completion can be seen in Figures 5-1 and 5-3. Figure 5-11 shows the SIO performing master-abort termination. This occurs when no slave responds to the SIO's master transaction by asserting DEVSEL# within 5 PCICLK's after FRAME# assertion. This master-abort condition is abnormal and it indicates an error condition. The SIO will not retry the cycle. The Received Master-abort Status bit in the PCI Status Register will be set indicating that the SIO experienced a master-abort condition.

If an ISA master or the DMA is waiting for the PCI cycle to terminate (CHRDY negated), the master-abort condition will cause the SIO to assert CHRDY to terminate the ISA cycle. Note that write data will be lost and the read data will be all 1's at the end of the cycle. This is identical to the way an unclaimed cycle is handled on the "normally ready" ISA Bus. If the line buffer is the requester of the PCI transaction, the master-abort mechanism will end the PCI cycle, but no data will be transferred into or out of the line buffer. The line buffer will not be allowed to retry the cycle.

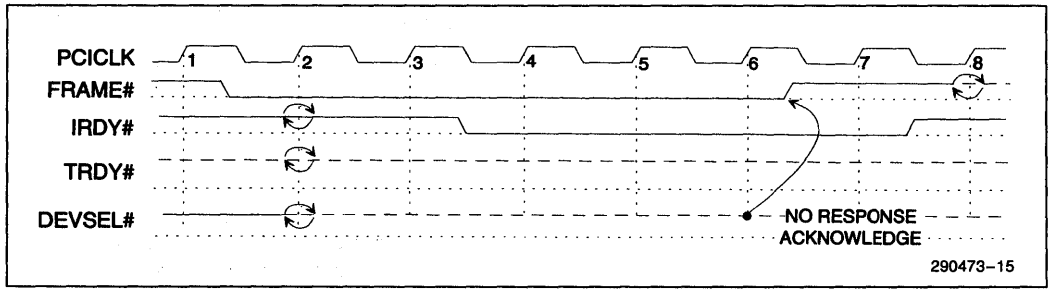


Figure 5-11. Master-Initiated Termination (Master-Abort)

5.1.3.2 SIO as a Master - Response to Target-Initiated Termination

SIO's response as a master to target-termination:

1. For a target-abort, the SIO will not retry the cycle. If an ISA master or the DMA is waiting for the PCI cycle to complete (CHRDY negated), the target-abort condition will cause the SIO to assert CHRDY and end the cycle on the ISA Bus. If the ISA master or DMA device was reading from PCI memory, the SIO will drive all 1's on the data lines of the ISA Bus. The Received Target-abort Status bit in the PCI Status Register will be set indicating that the SIO experienced a target-abort condition.
2. If the SIO is retried as a master on the PCI Bus, it will remove it's request for 2 PCI clks before asserting it again to retry the cycle.
3. If the SIO is disconnected as a master on the PCI Bus, it will respond very much as if it had been retried. The difference between retry and disconnect is that the SIO did not see any data phase for the retry. Disconnect may be generated by a PCI slave when the SIO is running a burst memory read cycle to fill it's 8-byte Line Buffer. In this case, the SIO may need to finish a multi-data phase transfer, and thus, must recycle through arbitration as required for a retry. An example of this is when the on-board DMA requests an 8 byte Line Buffer transfer and the SIO is disconnected before the Line Buffer is completely filled.

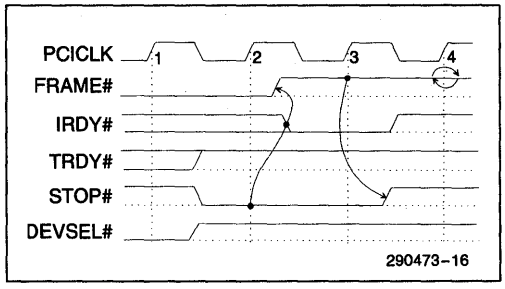


Figure 5-12. SIO Master, Target-Initiated Termination - Abort

5.1.3.3 SIO as a Target - Target-Initiated Termination

The SIO supports three forms of Target-initiated Termination:

Disconnect: Disconnect refers to termination requested because the SIO is unable to respond within the latency guidelines of the PCI specification. Note that this is not usually done on the first data phase.

Retry: Retry refers to termination requested because the target is currently in a state which makes it unable to process the transaction.

Abort: Abort refers to termination requested because the target will never be able to respond to the transaction.

See Figures 5-13, and 5-14 for the disconnect and retry forms of target-initiated termination used by the SIO as a target. As a slave, the SIO aborts as shown in figure 5-12. The SIO will initiate Disconnect for PCI-originated/ISA-bound cycles after the first data phase due to incremental latency requirements. Since the SIO has only one Posted Write Buffer and every PCI to ISA incremental data phase will take longer than the specified 8 clocks, the SIO will always terminate burst cycles with a disconnect protocol. An example of this is when the SIO receives a burst memory write. Since the SIO only has one Posted Write Buffer, the transaction will automatically be disconnected after the first data phase. Figure 5-13 shows the disconnect portion of the PCI cycle after the first data phase has completed.

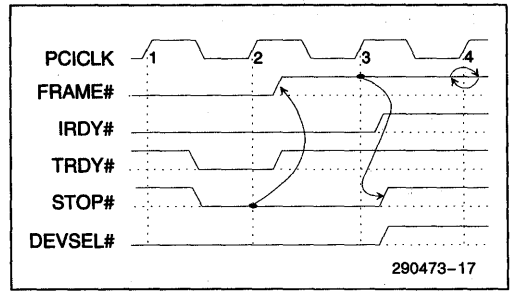


Figure 5-13. Target-Initiated Termination - Disconnect

The difference between disconnect and retry is that the SIO will not assert TRDY# for the retry case. This tells the master that it must retry the transfer at a later time. No data is transferred in a retry termination since TRDY# and IRDY# are never both asserted. The SIO will retry PCI masters as shown below:

1. For memory write cycles when the posted write buffer is full.
2. When the pending PCI cycle initiates some type of buffer management activity.
3. When the SIO is locked as a resource and a PCI master tries to access the SIO without negating the LOCK# signal in the address phase.
4. When the ISA Bus is occupied by an ISA master or DMA.

Target-abort is issued by the SIO when the internal SIO registers are the target of a PCI master I/O cycle and more than one byte enable is active. Accesses to the BIOS Timer Register and the Scatter/Gather Descriptor Table Pointer Registers are exceptions to this rule. Accesses to the Scatter/Gather Descriptor Table Pointer Register must be 32-bits wide and accesses to the BIOS Timer Register must be 16- or 32-bits wide. These accesses will not result in a SIO target abort. The SIO responds with a target-abort since the registers must be accessed as 8-bit quantities. Target-abort resembles a retry, although the SIO also negates DEVSEL# along with the assertion of STOP#. Bit 11 in the Device Status Register is set to a 1 when the SIO target-aborts.

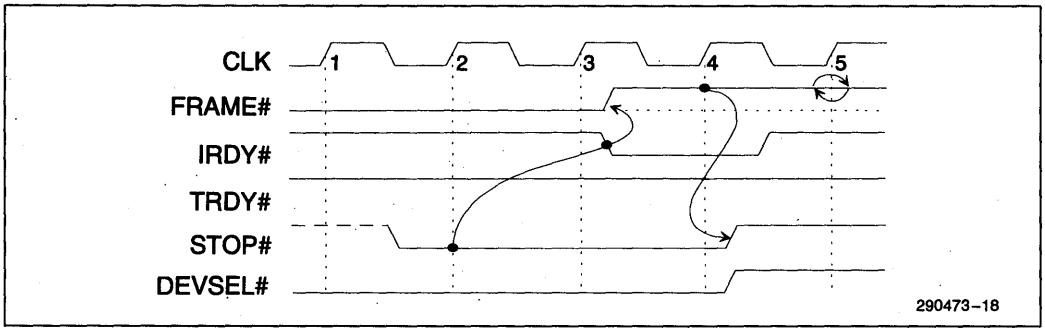


Figure 5-14. Target-initiated Termination - Retry

5.1.4 BUS LATENCY TIME-OUT

5.1.4.1 Master Latency Timer

Because the SIO only bursts a maximum of two dwords, the PCI master latency timer is not implemented.

5.1.4.2 Target Incremental Latency Mechanism

As a slave, the SIO supports the Incremental Latency Mechanism for PCI to ISA 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 8 PCICLK's, then the slave must manipulate TRDY# and STOP# to stop the transaction upon completion of the current data phase (N). Since all PCI-originated (SIO is a slave)/ISA-bound cycles will require greater than the stated 8 PCICLK's, the SIO will automatically terminate these cycles after the first data phase. Note that latency to the first data phase is not restricted by this mechanism.

5.1.5 PARITY SUPPORT

As a master, the SIO generates address parity for read and write cycles, and data parity for write cycles. As a slave, the SIO generates data parity for read cycles. The SIO does not check parity and does not generate SERR#.

PAR is the calculated parity signal. PAR is "even" parity and is calculated on 36 bits; the 32 AD[31:0] signals plus the 4 C/BE[3:0]# signals. "Even" parity means that the number of 1's within the 36 bits plus PAR are counted and the sum is always even. PAR is always calculated on 36 bits, regardless of the valid byte enables. PAR is only guaranteed to be valid one PCI clock after the corresponding address or data phase.

5.1.5.1 SIO Master Write Cycles

When the SIO is a master, it calculates address and data parity for write cycles. Parity is calculated on the address lines (AD[31:0]) and the command lines (C/BE[3:0]#) being driven during the address phase (first clock that FRAME# is asserted). The calculated address parity is driven on PAR exactly one clock after the address phase. Parity is also calculated on the data lines (AD[31:0]) and the Byte Enables (C/BE[3:0]#) every valid write data phase (IRDY# asserted) and is valid on PAR exactly one PCI clock after the corresponding data phase.

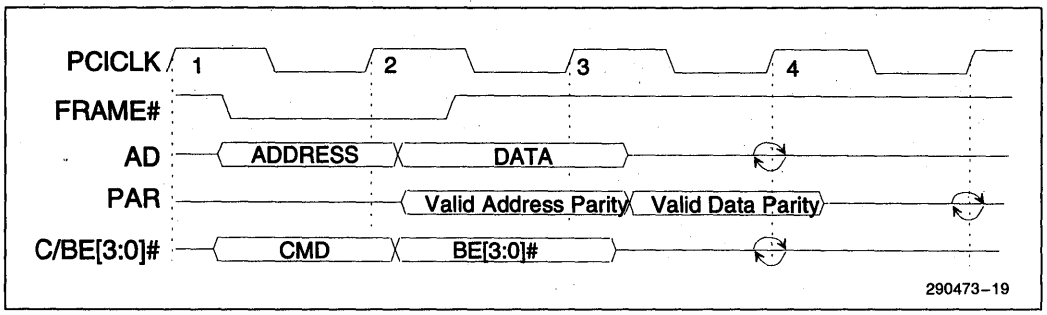


Figure 5-15. SIO Master Write Cycle Parity Operation

5.1.5.2 SIO Master Read Cycles

When the SIO is a master, it calculates address parity for read cycles. Parity is calculated on the address lines (AD[31:0]) and the command lines (C/BE[3:0]#) being driven during the address phase (first clock that FRAME# is asserted). The calculated address parity is driven on PAR exactly one clock after the address phase.

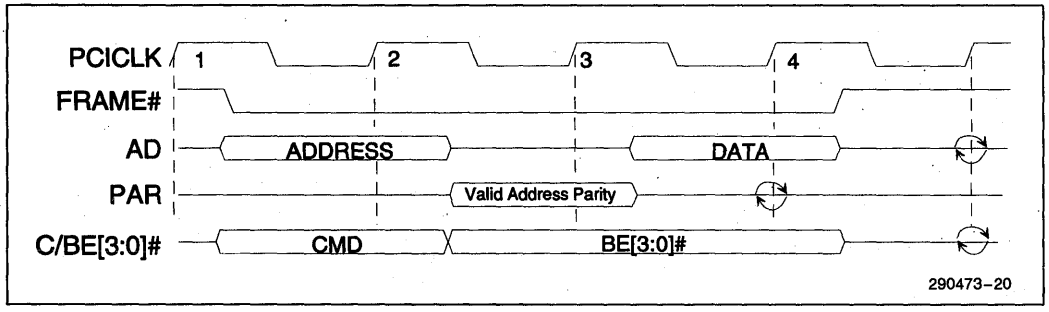


Figure 5-16. SIO Master Read Cycle Parity Operation

5.1.5.3 PCI Master Read Cycles from the SIO as Slave

When the SIO is the slave of a PCI read cycle, it calculates parity on the read data that the SIO is driving (AD[31:0]) and the Byte Enables (C/BE[3:0]#) that the PCI master is driving during a valid read data phase (TRDY# asserted). The calculated data parity is valid on PAR exactly one PCI clock after the corresponding data phase.

5.1.6 RESET SUPPORT

The PCIRST# pin acts as the SIO hardware reset pin.

During Reset

AD[31:0], C/BE[3:0]#, and PAR are always driven low by the SIO from the leading edge of PCIRST#. FRAME#, IRDY#, TRDY#, STOP# DEVSEL#, MEMREQ#, FLSHREQ#, CPUGNT#, GNT0#/SIOREQ#, and GNT1#/RESUME# are tri-stated from the leading edge of PCIRST#.

After Reset

AD[31:0], C/BE[3:0]#, and PAR are always tri-stated from the trailing edge of PCIRST#. If the internal arbiter is enabled (CPUREQ# sampled high on the trailing edge of PCIRST#), the SIO will drive these signals low again (synchronously 2-5 PCICLKs later) until the bus is given to another master. If the internal arbiter is disabled (CPUREQ# sampled high on the trailing edge of PCIRST#), these signals remain tri-stated until the SIO is required to drive them valid as a master or slave.

FRAME#, IRDY#, TRDY#, STOP#, and DEVSEL# remain tri-stated until driven by the SIO as either a master or a slave. MEMREQ#, FLSHREQ# CPUGNT#, GNT0#/SIOREQ#, and GNT1#/RESUME# are tri-stated until driven by the SIO. After PCIRST, MEMREQ# and FLSHREQ# are driven inactive asynchronously from PCIRST# inactive. CPUGNT#, GNT0#/SIOREQ#, and GNT1#/RESUME# are driven based on the arbitration scheme and the asserted REQx#'s.

5.1.7 DATA STEERING

Data steering logic internal to the SIO provides the assembly/disassembly, copy up/copy down mechanism for cycles between the 32-bit PCI data bus and the 16-bit ISA Bus. The steering logic ensures that the correct bytes are steered to the correct byte lane and that multiple cycles are run where applicable.

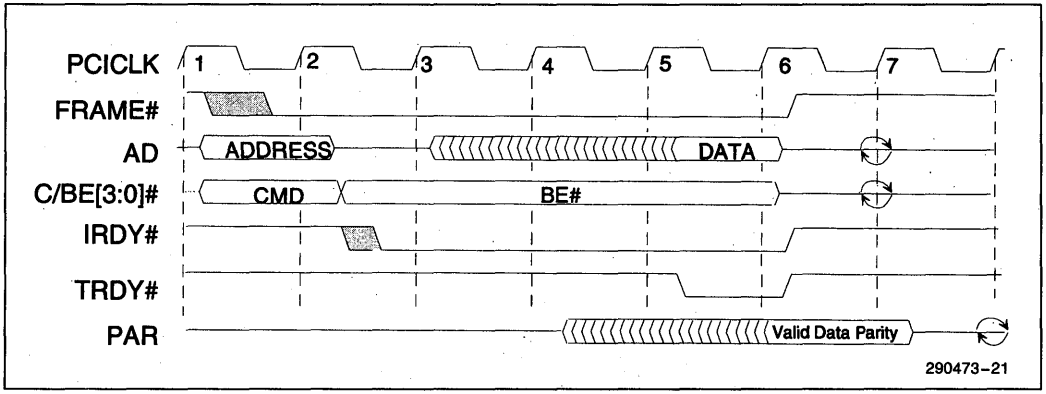


Figure 5-17. PCI Master Read Cycle from SIO as Slave - Parity Generation

5.2 PCI Arbitration Controller

The SIO contains a PCI Bus arbiter that supports four PCI masters; the Host Bridge, SIO, and two other masters. The SIO's REQ#/GNT# lines are internal. The integrated arbiter can be disabled by asserting CPUREQ# during PCIRST# (see Section 5.2.7, Power-up Configuration). When disabled, the SIO's REQ#, GNT#, and RESUME# signals become visible for an external arbiter. The internal arbiter is enabled upon power up.

The internal arbiter contains several features that contribute to system efficiency:

- Use of a RESUME# signal to re-enable a backed-off initiator in order to minimize PCI Bus thrashing when the SIO generates a retry (Section 5.2.4.1).
- A programmable timer to re-enable retried initiators after a programmable number of PCICLK's (Section 5.2.4.2).
- The CPU (host bridge) can be optionally parked on the PCI Bus (Section 5.2.5).
- A programmable PCI Bus lock or PCI resource lock function (Section 5.2.6).

The PCI arbiter is also responsible for control of the Guaranteed Access Time (GAT) mode signals (Section 5.2.3.2).

5.2.1 ARBITRATION SIGNAL PROTOCOL

The internal arbiter follows the PCI arbitration method as outlined in the *Peripheral Component Interconnect (PCI) Specification*. The SIO's arbiter is discussed in this section.

5.2.1.1 REQ# and GNT# Rules

Figure 5-18 illustrates basic arbitration. Two agents are used to illustrate how the arbiter alternates bus accesses.

Note in Figure 5-18 that the current owner of the bus may keep its REQ# (REQ#-a) asserted when it requires additional transactions.

The arbiter may negate an agent's GNT# according to the following rules:

1. 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 GNT# and assertion of the next to comply with the PCI specification.
2. While FRAME# is negated, GNT# may be negated at any time in order to service a higher priority initiator, or in response to the associated REQ# being negated.
3. If the MEMREQ# and MEMACK# are active, once the SIO is granted the PCI Bus (internal or external SIOGNT#), the arbiter will not remove the SIOGNT# until the SIO removes it's request.

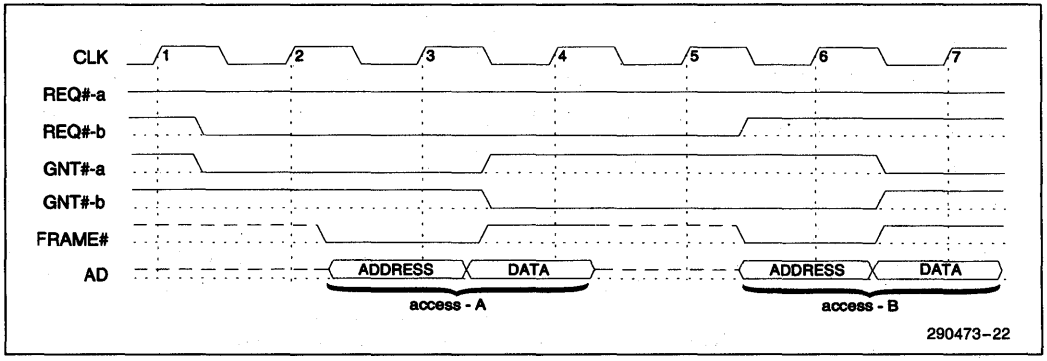


Figure 5-18. Basic PCI Arbitration

5.2.1.2 Back-to-Back Transactions

Figure 5-19 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 and those that do require a turn-around-cycle. The first is limited to the case when the initiator's second transaction is to the same target as the first (to insure no TRDY# contention), and the first transaction is a write. This is a fast back-to-back transaction. 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 the last data is transferred for the previous transaction.

The SIO as a master does not generate fast back-to-back accesses since it does not know if it is accessing the same target.

The SIO as a target supports fast back-to-back transactions. Note that for back-to-back cycles, the SIO treats positively decoded accesses and subtractively decoded accesses as different targets. Therefore, masters can only run fast back-to-back cycles to positively decoded addresses or to subtractively decoded addresses. Fast back-to-back cycles must not mix positive and subtractive decoded addresses. See the address decoding section to determine what addresses the SIO positively decodes and subtractively decodes.

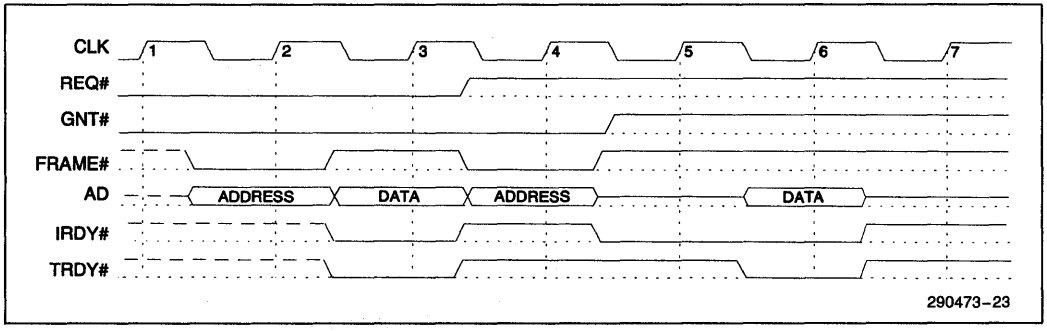


Figure 5-19. Arbitration for Back-to-Back Access

5.2.2 PRIORITY SCHEME

The PCI arbitration priority scheme is programmable through the PCI Arbiter Priority Control Register. The arbiter consists of three banks that can be configured for the four masters to be arranged in a purely rotating priority scheme, one of eight fixed priority schemes, or a hybrid combination.

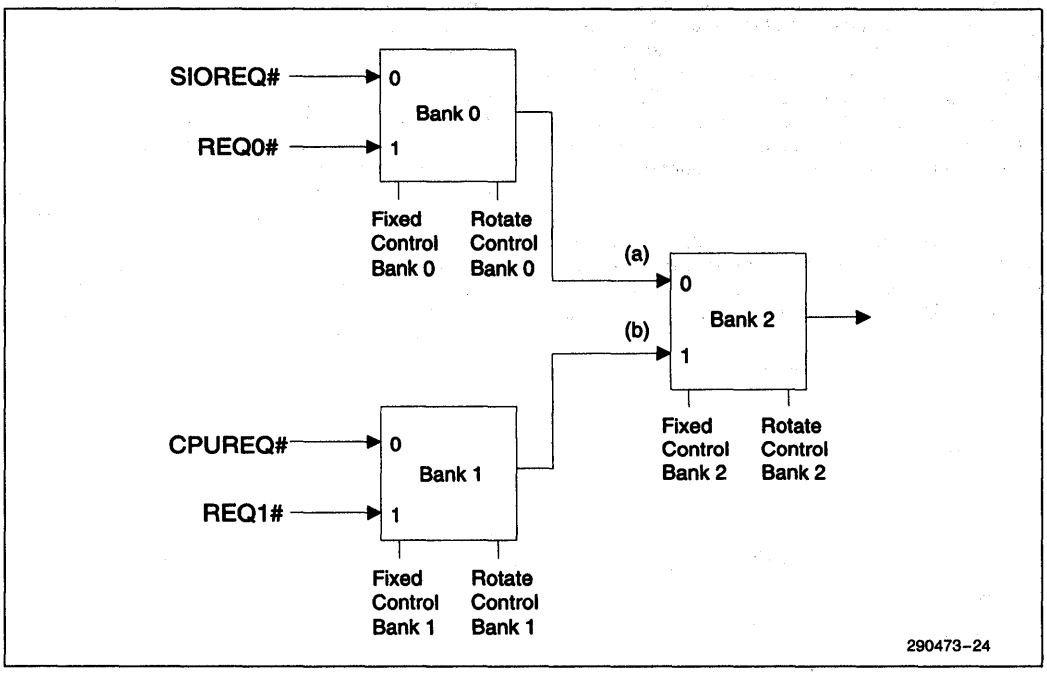


Figure 5-20. Arbiter Configuration Diagram

The PCI arbiter priority configuration register to support this is as follows:

Bit 0 Bank 0 Fixed Priority mode select
Bit 1 Bank 1 Fixed Priority mode select
Bit 2 Bank 2 Fixed Priority mode select
Bit 3 Reserved
Bit 4 Bank 0 Rotate Control
Bit 5 Bank 1 Rotate Control
Bit 6 Bank 2 Rotate Control
Bit 7 Reserved

The PCI Arbiter Priority Register defaults to 00000100b at reset. This selects fixed mode 4 with the CPU the highest priority device guaranteeing that BIOS accesses can take place.

5.2.2.1 Fixed Priority Mode

The eight selectable fixed priority schemes are listed below:

Table 5-2. Fixed Mode Bank Control Bits:

Mode	Bank			Priority			
	2	1	0	Highest			Lowest
0	0	0	0	SIOREQ#	REQ0#	CPUREQ#	REQ1#
1	0	0	1	REQ0#	SIOREQ#	CPUREQ#	REQ1#
2	0	1	0	SIOREQ#	REQ0#	REQ1#	CPUREQ#
3	0	1	1	REQ0#	SIOREQ#	REQ1#	CPUREQ#
4	1	0	0	CPUREQ#	REQ1#	SIOREQ#	REQ0#
5	1	0	1	CPUREQ#	REQ1#	REQ0#	SIOREQ#
6	1	1	0	REQ1#	CPUREQ#	SIOREQ#	REQ0#
7	1	1	1	REQ1#	CPUREQ#	REQ0#	SIOREQ#

The fixed bank control bit(s) select which requester is the highest priority device within that particular bank. Bits 4-6 must all be programmed to 0's (rotate mode disabled) to get these combinations.

The selectable fixed schemes provide 8 fixed modes for the four masters

5.2.2.2 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 three banks are set in rotate mode, the four supported masters are all rotated and the arbiter is in a pure rotating priority scheme.

As an example, consider Bank 2 in Figure 5-20. If the input labeled (a) is the highest priority input at the moment, but does not have an active request outstanding, the lower priority input labeled (b) will be granted the bus if it has an active request. However, this does not change the rotation scheme of Bank 2 (if enabled). When Bank 2 toggles, then (b) will be the highest priority device. Because of this, the maximum latency a device may encounter is two complete rotations.

If all four masters are requesting the bus continuously, and the arbiter starts in mode 0, the sequence of priority schemes is:

Mode 0, Mode 5, Mode 3, Mode 6, Mode 0, Mode 5, Mode3, ...

and the GNT# sequence is:

SIOGNT#, CPUGNT#, GNT0#, GNT1#, SIOGNT#, CPUGNT#, GNT0#, ...

This example is summarized in Table 5-3:

Table 5-3. Rotating Priority Mode Example - All Requests Active - All Banks in Rotation Mode

Arbitration Sequence	Bank State			Mode	Priority Output	Banks Toggled		
	2	1	0			Bank 2	Bank 1	Bank 0
Start (1)	0	0	0	0	SIOGNT#	X	-	X
2	1	0	1	5	CPUGNT#	X	X	-
3	0	1	1	3	GNT0#	X	-	X
4	1	1	0	6	GNT1#	X	X	-
5	0	0	0	0	SIOGNT#	X	-	X
6	1	0	1	5	CPUGNT#	X	X	-
7	0	1	1	3	GNT0#	X	-	X

When the banks are in mixed combinations of fixed mode or rotating mode, the fixed mode table (Table 5-2) along with the rotating bank rules allow the priority sequences to be determined.

5.2.2.3 Locking Masters

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

5.2.3 MEMREQ#, FLSHREQ#, AND MEMACK# PROTOCOL

Before an ISA master or the DMA can be granted the PCI Bus, it is necessary that all PCI system posted write buffers be flushed (including the SIO's Posted Write Buffer). Also, since the ISA originated cycle could access memory on the host bridge, it's possible that the ISA master or the DMA could be held in wait states (via IOCHRDY) waiting for the host bridge arbitration for longer than the 2.5 μs ISA specification. The SIO has an optional mode called the Guaranteed Access Time Mode (GAT) that ensures that this timing specification is not violated. This is accomplished by delaying the ISA REQ# signal to the requesting master or DMA until the ISA Bus, PCI Bus, and the system memory bus are arbitrated for and owned.

Three PCI sideband signals, MEMREQ#, FLSHREQ#, and MEMACK# are used to support the System Posted Write Buffer Flushing and Guaranteed Access Time mechanisms. The MEMACK# signal is the common acknowledge signal for both mechanisms. Note that when MEMREQ# is asserted, FLSHREQ# is also asserted. The table below shows the relationship between MEMREQ# and FLSHREQ#:

Table 5-4. FLSHREQ#, MEMREQ#

FLSHREQ#	MEMREQ#	Meaning
1	1	Idle
0	1	Flush buffers pointing towards PCI to avoid ISA deadlock
1	0	Reserved
0	0	GAT mode. Guarantee PCI Bus immediate access to main memory

5.2.3.1 Flushing the System Posted Write Buffers

Once an ISA master or the DMA begins a cycle on the ISA Bus, the cycle can not be backed-off. It can only be held in wait states via IOCHRDY. In order to know the destination of ISA master cycles, the cycle needs to begin. However, after the cycle has started, no other device can intervene and gain ownership of the ISA Bus until the cycle has completed and arbitration is performed. A potential deadlock condition exists when an ISA originated cycle to the PCI Bus finds the PCI target inaccessible due to an interacting event that also requires the ISA Bus. To avoid this potential deadlock, all PCI posted write buffers in the system must be disabled and flushed before DACK can be returned. The buffers must remain disabled while the ISA Bus is occupied by an ISA master or the DMA.

When an ISA master or the DMA requests the ISA Bus, the SIO asserts FLSHREQ#. FLSHREQ# is an indication to the system to flush all posted write buffers pointing towards the PCI Bus. The SIO also flushes it's own Posted Write Buffer. Once the posted write buffers have been flushed and disabled, the system asserts MEMACK#. Once the SIO receives the MEMACK# acknowledgment signal, it asserts the DACK signal giving the requesting master the bus. FLSHREQ# stays active as long as the ISA master or DMA owns the ISA Bus.

5.2.3.2 Guaranteed Access Time Mode

Guaranteed Access Time (GAT) Mode is enabled/disabled via the PCI Arbiter Control Register. When this mode is enabled, the MEMREQ# and MEMACK# signals are used to guarantee that the ISA 2.5μs IOCHRDY specification is not violated.

When an ISA master or DMA slave requests the ISA Bus (DREQ# active), the ISA Bus, the PCI Bus, and the memory bus must be arbitrated for and all three must be owned before the ISA master or DMA slave is granted the ISA Bus. After receiving the DREQ# signal from the ISA master or DMA slave, MEMREQ# and FLSHREQ# are asserted (FLSHREQ# is driven active, regardless of GAT mode being enabled or disabled). MEMREQ# is a request for direct access to main memory. MEMREQ# and FLSHREQ# will be asserted as long as the ISA master or the DMA owns the ISA Bus. When MEMACK# is received by the SIO (all posted write buffers are flushed and the memory bus is dedicated to the PCI interface), it will request the PCI Bus. When it is granted the PCI Bus, it asserts the DACK signal releasing the ISA Bus to the requesting master or the DMA.

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

5.2.4 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 will then normally request the PCI Bus again. To avoid thrashing the bus with retry after retry, the PCI arbiter provides REQ# masking. The REQ# masking mechanism differentiates between SIO target retries and all other retries.

For initiators which were retried by the SIO as a target, the masked REQ# is flagged to be cleared upon RESUME# active. All other retries trigger the Master Retry Timer, if enabled. Upon expiration of this timer, the mask is cleared.

5.2.4.1 Resume Function (RESUME#)

The conditions under which the SIO forces a retry to a PCI master and will mask the REQ# are:

1. Any required buffer management
2. ISA Bus occupied by ISA master or DMA
3. The PCI to ISA Posted Write Buffer is full
4. The SIO is locked as a resource and LOCK# is asserted during the address process.

The RESUME# signal is pulsed whenever the SIO has retried a PCI cycle for one of the above reasons and that condition has passed. When RESUME# is asserted, the SIO will unmask the REQ#'s that are masked and flagged to be cleared by RESUME#.

If the internal arbiter is enabled, RESUME# is an internal signal. The RESUME# signal becomes visible as an output when the internal arbiter is disabled. This allows an external arbiter to optionally avoid retry thrashing associated with the SIO as a target. The RESUME# signal is asserted for one PCI clock.

5.2.4.2 Master Retry Timer

To re-enable a PCI master's REQ# which resulted in a retry to a slave other than the SIO, a SIO programmable Master Retry Timer has been provided. This timer can be programmed for 0 (disabled), 16, 32, or 64 PCICLKs. Once the SIO has detected that a PCI slave has forced a retry, the timer will be triggered and the corresponding master's REQ# will be masked. All subsequent PCI retries by this REQ# signal will by masked by the SIO. Expiration of this timer will unmask all of the masked requests. This

timer has no effect on the request lines that have been masked due to a SIO retry.

If no other PCI masters are requesting the PCI Bus, all of the REQ#s masked for the timer will be cleared and the timer will be reset. This is necessary to assist the host bridge in determining when to re-enable any disabled posted write buffers.

5.2.5 BUS PARKING

The SIO arbitration logic supplies a mechanism for PCI Bus parking. Parking is only allowed for the device which is tied to CPUREQ# (typically the system CPU). When bus parking is enabled, CPUGNT# will be asserted when no other agent is currently using or requesting the bus. This achieves the minimum PCI arbitration latency possible. Enabling of bus parking is achieved by programming the Arbiter Control Register. REQ0#, REQ1#, and the internal SIOREQ# are not allowed to park on the PCI Bus.

Upon assertion of CPUGNT# due to bus parking enabled and the PCI Bus idle, the CPU (or the parked agent) must ensure that AD[31:0], C/BE[3:0], and (one PCICLK later) PAR are driven. If bus parking is disabled, the SIO takes responsibility for driving the bus when it is idle.

5.2.6 BUS LOCK MODE

As an option, the SIO arbiter can be configured to run in Bus Lock Mode or Resource Lock Mode. 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. Bus Lock Mode emulates the LOCK environment found in today's PC by restricting bus ownership when the PCI Bus is locked. With Bus Lock enabled, the arbiter recognizes a LOCK# being driven by any 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 read and write cycles and run non-exclusive accesses to any unlocked resource.

5.2.7 POWER-UP CONFIGURATION

The SIO's arbiter is enabled if CPUREQ# is sampled high on the trailing edge of PCIRST#. When enabled, the arbiter is set in fixed priority mode 4 with CPU bus parking turned off. Fixed mode 4 guarantees that the CPU will be able to run accesses to the BIOS in order to configure the system, regardless of the state of the other REQ#'s. Note that the Host Bridge should drive CPUREQ# high during the trailing edge of PCIRST#. When the arbiter is enabled, the SIO acts as the central resource responsible for driving the AD[31:0], C/BE[3:0]#, and PAR signals when no one is granted the PCI Bus and the bus is idle. The SIO is always responsible for driving AD[31:0], C/BE[3:0]#, and PAR when it is granted the bus and as appropriate when it is the master of a transaction. After reset, if the arbiter is enabled, CPUGNT#, GNT0#, GNT1#, and the internal SIOGNT# will be driven based on the arbitration scheme and the asserted REQ#'s.

Table 5-5. Arbitration Latency

Bus Condition	Arbitration Latency
Parked	0 PCICLKs for agent 0, 2 PCICLKs for all other
Not Parked	1 PCICLK for all agents

If an external arbiter is present in the system, the CPUREQ# signal should be tied low. When CPUREQ# is sampled low on the trailing edge of PCIRST#, the internal arbiter is disabled. When the internal arbiter is disabled, the SIO does not drive AD[31:0], C/BE[3:0]#, and PAR as the central resource. In this case, the SIO is only responsible for driving AD[31:0], C/BE[3:0]#, and PAR when it is granted the bus. If the SIO's arbiter is disabled, GNT0# becomes SIOREQ#, GNT1# becomes RESUME#, and REQ0# becomes SIOGNT#. This exposes the normally embedded SIO arbitration signals.

5.3 ISA Interface

5.3.1 ISA INTERFACE OVERVIEW

The SIO incorporates a fully ISA Bus compatible master and slave interface. The SIO directly drives six ISA slots without external data or address buffers. The ISA interface also provides byte swap logic, I/O recovery support, wait state generation, and SYSCLK generation.

The ISA interface supports the following types of cycles:

- PCI-initiated I/O and memory cycles to the ISA Bus.
- DMA compatible cycles between PCI memory and ISA I/O and between ISA I/O and ISA memory, DMA type "A", type "B", and type "F" cycles between PCI memory and ISA I/O.
- ISA Refresh cycles initiated by either the SIO or an external ISA master.
- ISA master-initiated memory cycles to the PCI Bus and ISA master-initiated I/O cycles to the internal SIO registers.

The refresh and DMA cycles are shown and described in Section 5.4. A description of the remaining cycles is provided in this section.

5.3.2 SIO AS AN ISA MASTER

The SIO executes ISA cycles as an ISA master whenever a PCI initiated cycle is forwarded to the ISA Bus. The SIO also acts as an ISA master on behalf of DMA and refresh. DMA and refresh cycles are discussed in detail in Section 5.4 and 5.4.9

Figures 5-21 to 5-31 illustrate the various memory and I/O cycles supported by the SIO. As an ISA master, the SIO executes compressed cycles whenever the ZEROWS# signal is detected, except in the case of 16-bit I/O cycles.

In Figures 5-21 to 5-31, ISYSCLK is shown as a reference only. ISYSCLK is an internal 8 MHz clock.

5.3.2.1 Memory Read/Write Standard Cycles

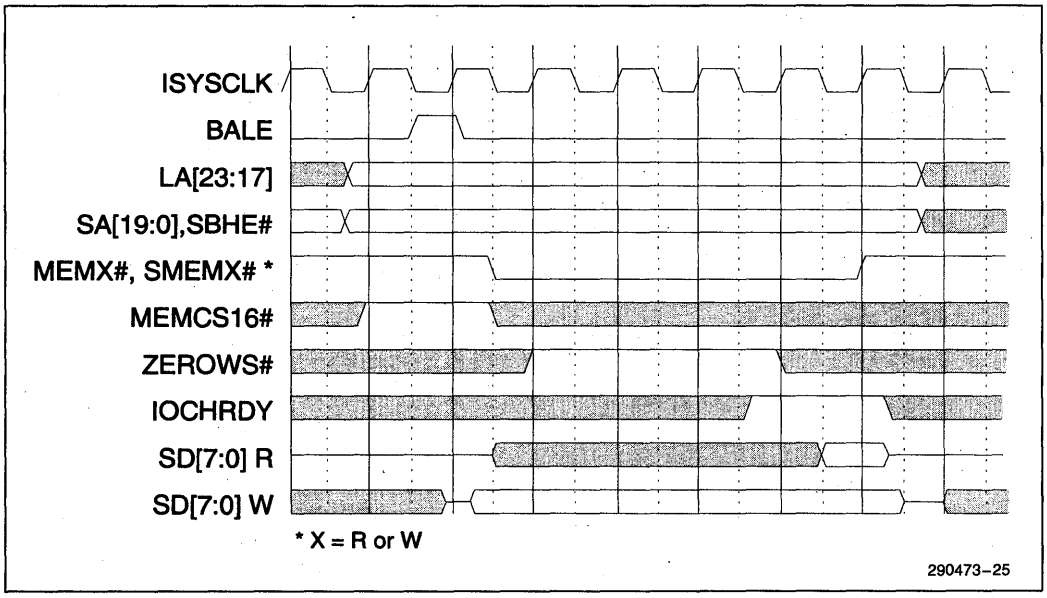


Figure 5-21. 8-Bit Memory Read/Write Standard ISA Cycle (6 SYSCLK)

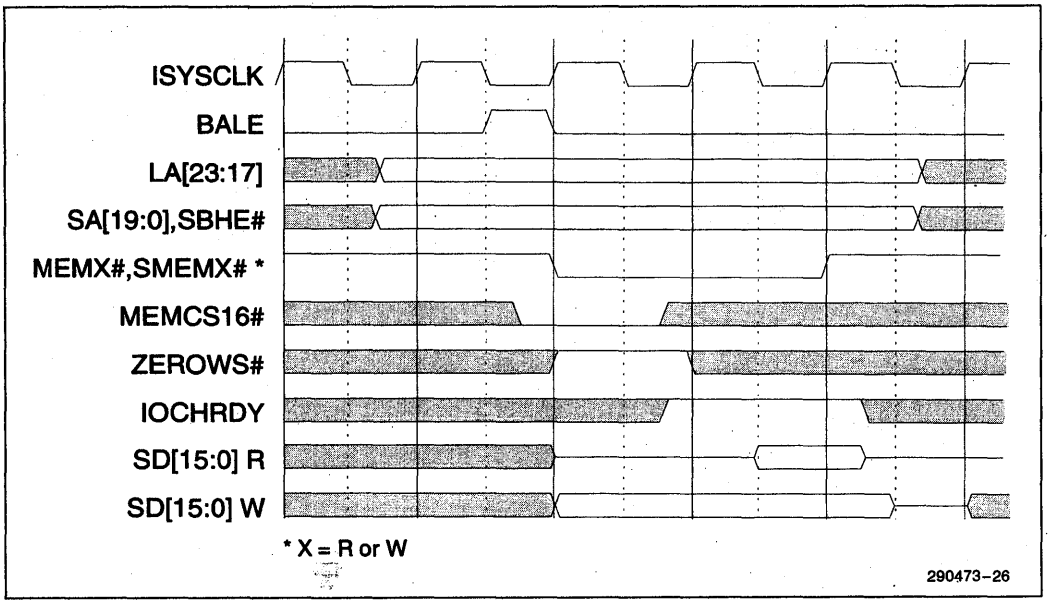


Figure 5-22. 16-Bit Memory Read/Write Standard ISA Cycle (3 SYSCLK)

5.3.2.2 Memory Read/Write Extended Cycles

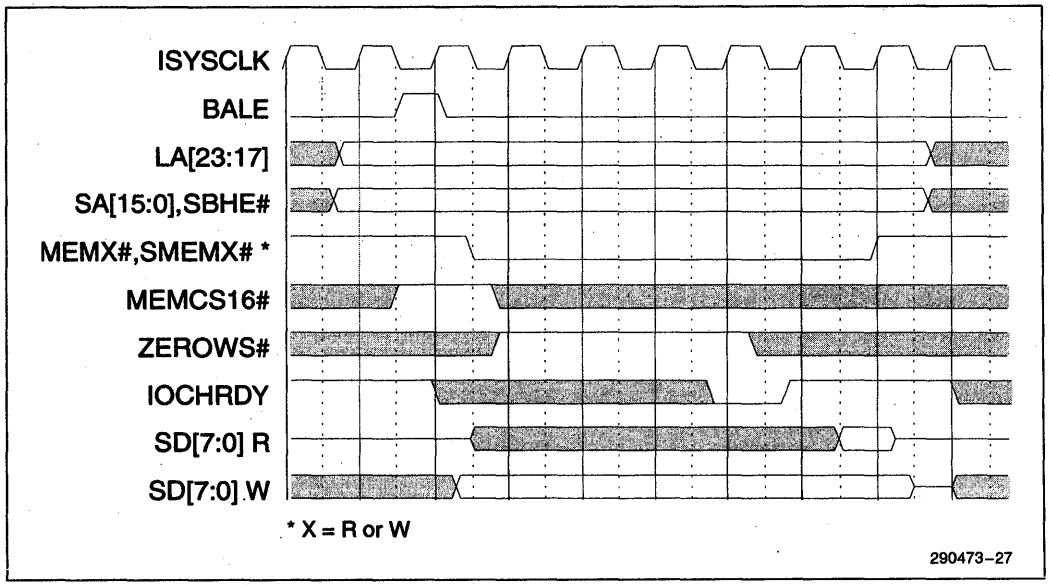


Figure 5-23. 8-Bit Memory Read/Write Extended Cycle (7 SYSCLK shown)

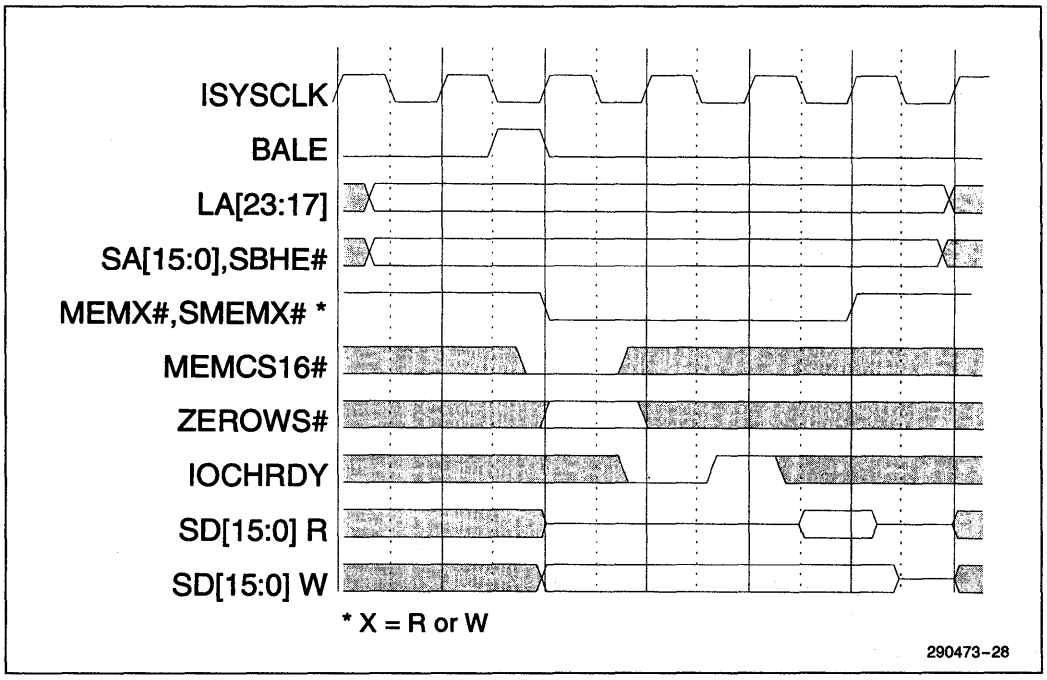


Figure 5-24. 16-Bit Memory Read/Write Extended Cycle (4 SYSCLK shown)

5.3.2.3 Memory Read/Write Compressed Cycles

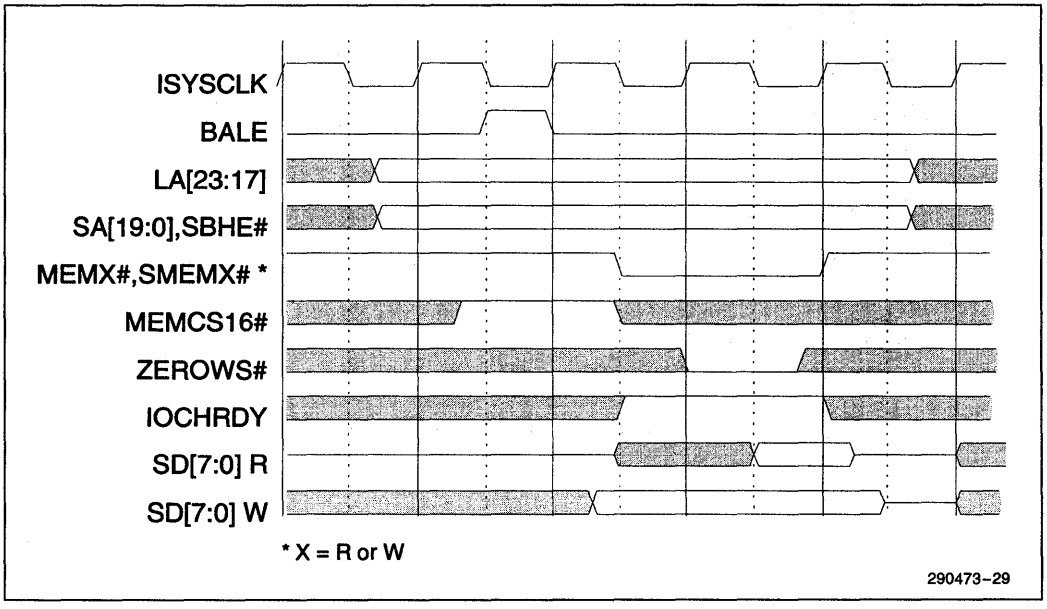


Figure 5-25. 8-Bit Memory Read/Write Compressed Cycle (3 SYSCLK)

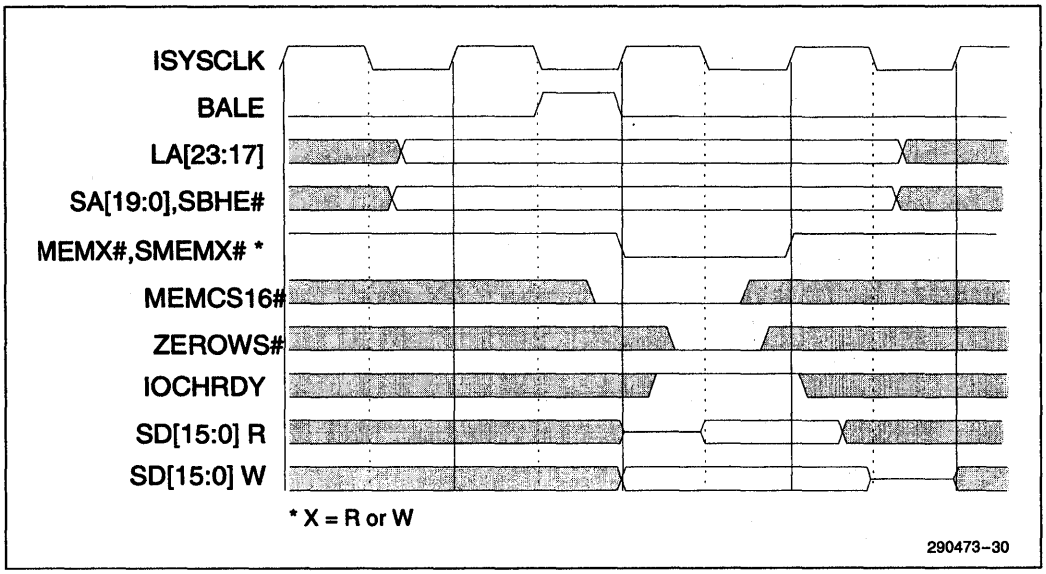


Figure 5-26. 16-Bit Memory Read/Write Compressed Cycle (2 SYSCLK)

5.3.2.4 I/O Read/Write Standard Cycles

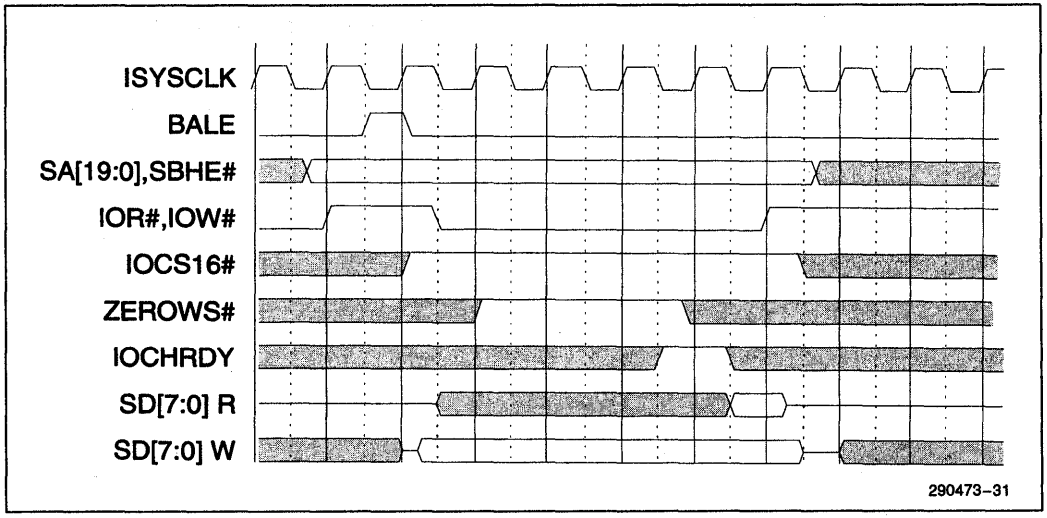


Figure 5-27. 8-Bit I/O Read/Write Standard ISA Cycle (6 SYSCLK)

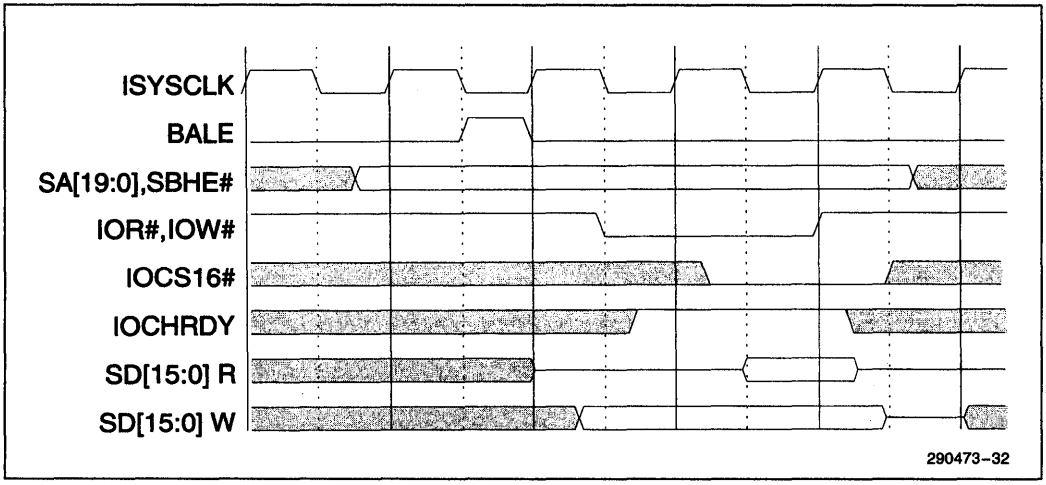


Figure 5-28. 16-Bit I/O Read/Write Standard ISA Cycle (3 SYSCLK)

5.3.2.5 I/O Read/Write Extended Cycles

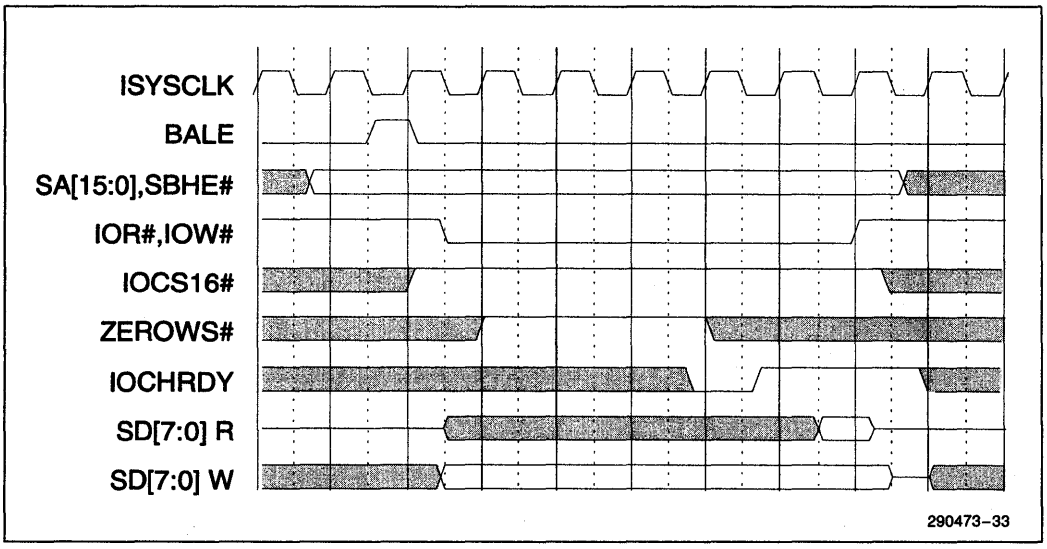


Figure 5-29. 8-Bit I/O Read/Write Extended ISA Cycle (7 SYSCLK shown)

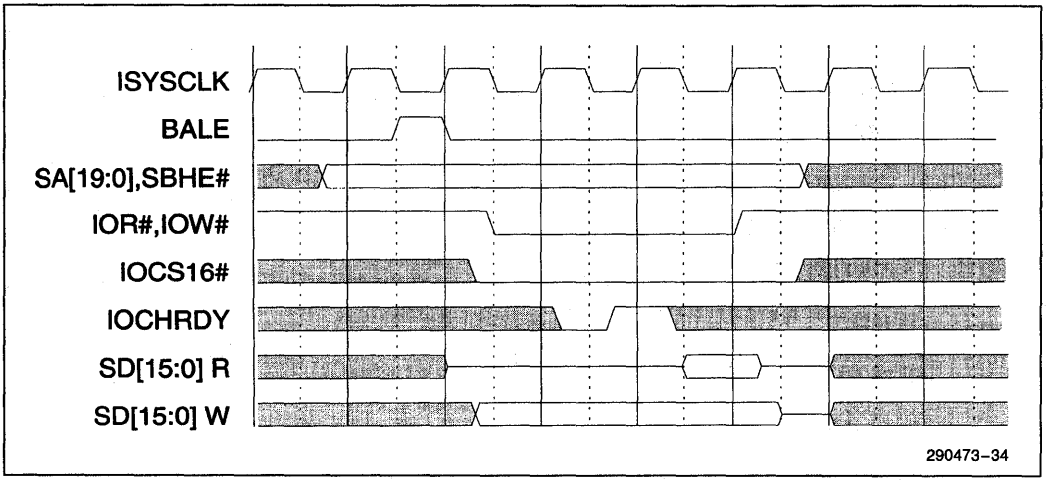


Figure 5-30. 16-Bit I/O Read/Write Extended ISA Cycle (4 SYSCLK shown)

5.3.2.6 I/O Read/Write Compressed Cycles

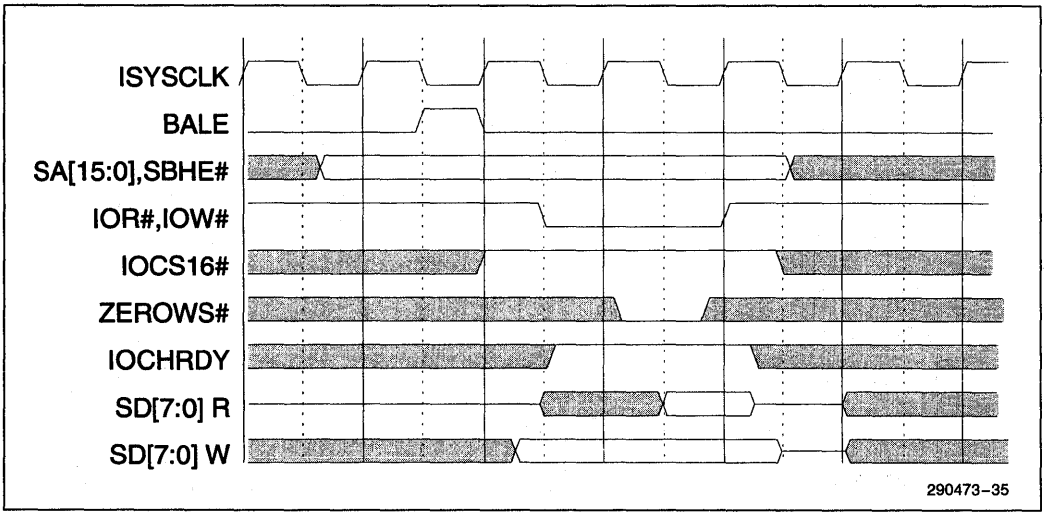


Figure 5-31. 8-Bit I/O Read/Write Compressed ISA Cycle (3 SYSCLK)

5.3.3 SIO AS AN ISA SLAVE

The SIO operates as an ISA slave when:

- An ISA master accesses SIO internal registers.
- An ISA master accesses PCI memory on the PCI Bus.

5.3.3.1 ISA Master Accesses to SIO Registers

An ISA Bus master has access to SIO internal registers as shown in table 5-14. An ISA master to SIO register cycle will always run as an 8-bit extended cycle (IOCHRDY will be held inactive until the cycle is completed).

5.3.3.2 ISA Master Accesses to PCI Resource

An ISA master can access PCI memory, but not I/O devices residing on the PCI Bus. The ISA/DMA address decoder determines which memory cycles should be directed towards the PCI Bus. During ISA master read cycles to the PCI Bus, the SIO will return all 1's if the PCI cycle is target-aborted or does not respond.

If the SIO is programmed for GAT mode, the SIO arbiter will not grant the ISA Bus before gaining ownership of both the PCI Bus and system memory. However, if the SIO is not programmed in this mode, the SIO does not need to arbitrate for the PCI Bus before granting the ISA Bus to the ISA master. For more details on the arbitration, refer to section 5.2.3.

All cycles forwarded to a PCI resource will run as 16 bit extended cycles (i.e. IOCHRDY will be held inactive until the cycle is completed).

Because the ISA bus size is different from the PCI bus size, the data steering logic inside the SIO is responsible for steering the data to the correct byte lanes on both busses, and assembling/disassembling the data as necessary.

5.3.4 ISA MASTER TO ISA SLAVE SUPPORT

During ISA master cycles to ISA slaves, the SIO drives several signals to support the transfer:

BALE: This signal is driven high while the ISA master owns the ISA Bus.

AEN: This signal is driven low while the ISA master owns the ISA Bus.

SMEMR# and SMEMW#: These signals are driven active by the SIO whenever the ISA master drives a memory cycle to an address below 1 Mb.

Utility Bus Buffer Control Signals and Chip Select Signals: These signals are driven active as appropriate whenever an ISA master accesses devices on the Utility Bus. For more details, see section 5.9.

Data Swap Logic: The data swap logic inside the SIO is activated as appropriate to swap data between the even and odd byte lanes. This is discussed in further detail in Section 5.3.5.

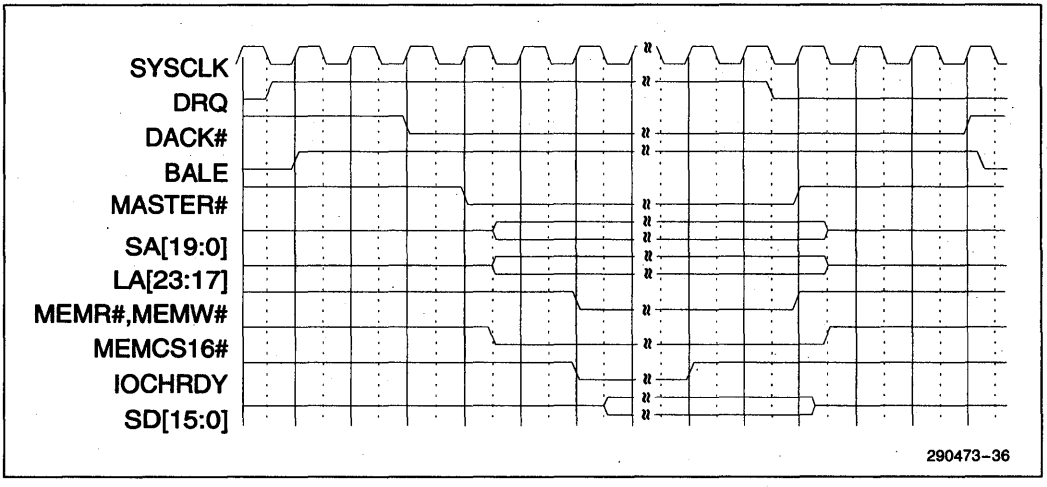


Figure 5-32. ISA Bus Master Access Cycle to PCI Memory

5.3.5 DATA BYTE SWAPPING

The data swap logic is integrated in the SIO. For slaves that reside on the ISA Bus, data swapping is performed if the slave (I/O or memory) and ISA bus master (or DMA) sizes differ and the upper (odd) byte of data is being accessed. Table 5-6 below shows when data swapping is provided during DMA. Table 5-7 shows when data swapping is provided during ISA master cycles to 8-bit ISA slaves.

The SIO monitors the SBHE# and SA0 signals to determine when to swap the data. The SIO ensures that the data is placed on the appropriate byte lane.

Table 5-6. DMA Data Swap

DMA I/O Device Size	ISA Memory Slave Size	Swap	Comments (I/O) <= = = = = = > Memory
8-bit	8-bit	No	SD[7:0] = = SD[7:0]
8-bit	16-bit	Yes	SD[7:0] = = SD[7:0] SD[7:0] = = SD[15:8]
16-bit	8-bit	No	Not Supported
16-bit	16-bit	No	SD[15:0] = = SD[15:0]

Table 5-7. 16-Bit Master to 8-Bit Slave Data Swap

SBHE#	SA0	SD[15:8]	SD[7:0]	Comments
0	0	Odd	Even	Word Transfer (data swapping not required)
0	1	Odd	Odd	Byte Swap 1,2
1	0	-	Even	Byte Transfer (data swapping not required)
1	1	-	-	Not Allowed

NOTES:
1) For ISA master read cycles, the SIO swaps the data from the lower byte to the upper byte.
2) For ISA master write cycles, the SIO swaps the data from the upper byte to the lower byte.

5.3.6 ISA CLOCK GENERATION

The SIO generates the ISA system clock (SYSCLK). SYSCLK is a divided down version of the PCICLK (see Table 5-8). The clock divisor value is programmed through the ISA Clock Divisor Register.

Table 5-8. SYSCLK Generation from PCICLK

PCICLK (MHz)	DIVISOR (Programmable)	SYSCLK (MHz)
25	3	8.33
33	4 (default)	8.33

Note that for PCI frequencies less than 33 Mhz (not including 25 Mhz), a clock divisor value must be selected that ensures that the ISA Bus frequency does not violate the 6 Mhz to 8.33 Mhz SYSCLK specification.

5.3.7 WAIT STATE GENERATION

The SIO will add wait states to the following cycles, if IOCHRDY is sampled active low. Wait states will be added as long as IOCHRDY is low.

- During Refresh and SIO master cycles (not including DMA) to the ISA Bus.
- During DMA compatible transfers between ISA I/O and ISA memory only.

For ISA master cycles targeted for the SIO's internal registers or PCI memory, the SIO will always extend the cycle by driving IOCHRDY low until the transaction is complete.

The SIO will shorten the following cycles, if ZEROWS# is sampled active.

- During SIO master cycles (not including DMA) to 8-bit and 16-bit ISA memory.
- During SIO master cycles (not including DMA) to 8-bit ISA I/O only.

For ISA master cycles targeted for the SIO's internal registers or PCI memory, the SIO will not assert ZEROWS#.

NOTE:
If IOCHRDY and ZEROWS# are sampled active at the same time, IOCHRDY will take precedence and wait states will be added.

5.3.8 I/O RECOVERY

The I/O recovery mechanism in the SIO is used to add additional recovery delay between PCI originated 8-bit and 16-bit I/O cycles to the ISA Bus. The SIO automatically forces a minimum delay of four SYSCLKs between back-to-back 8 and 16 bit I/O cycles to the ISA Bus. This delay is measured from the rising edge of the I/O command (IOR# or IOW#) to the falling edge of the next BALE. If a delay of greater than four SYSCLKs is required, the ISA I/O Recovery Time Register can be programmed to increase the delay in increments of SYSCLKs. No additional delay is inserted for back-to-back I/O "sub cycles" generated as a result of byte assembly or disassembly.

5.4 DMA Controller

5.4.1 DMA CONTROLLER OVERVIEW

The DMA circuitry incorporates the functionality of two 82C37 DMA controllers with seven independently programmable channels (Channels 0-3 and Channels 5-7). DMA Channel 4 is used to cascade the two controllers and will default to cascade mode in the DMA Channel Mode (DCM) Register. In addition to accepting requests from DMA slaves, the DMA controller also responds to requests that are initiated by software. Software may initiate a DMA service request by setting any bit in the DMA Channel Request Register to a 1. The DMA controller for Channels 0-3 is referred to as "DMA-1" and the controller for Channels 4-7 is referred to as "DMA-2".

Each DMA channel may be programmed for 8 or 16 bit DMA device size and ISA-compatible, Type "A", Type "B", or Type "F" transfer timing. Each DMA channel defaults to the compatible settings for DMA device size: channels [3:0] default to 8-bit, count-by-bytes transfers, and channels [7:5] default to 16-bit, count-by-words (address shifted) transfers. The SIO provides the timing control and data size translation necessary for the DMA transfer between the PCI and the ISA Bus. ISA-compatible is the default transfer timing.

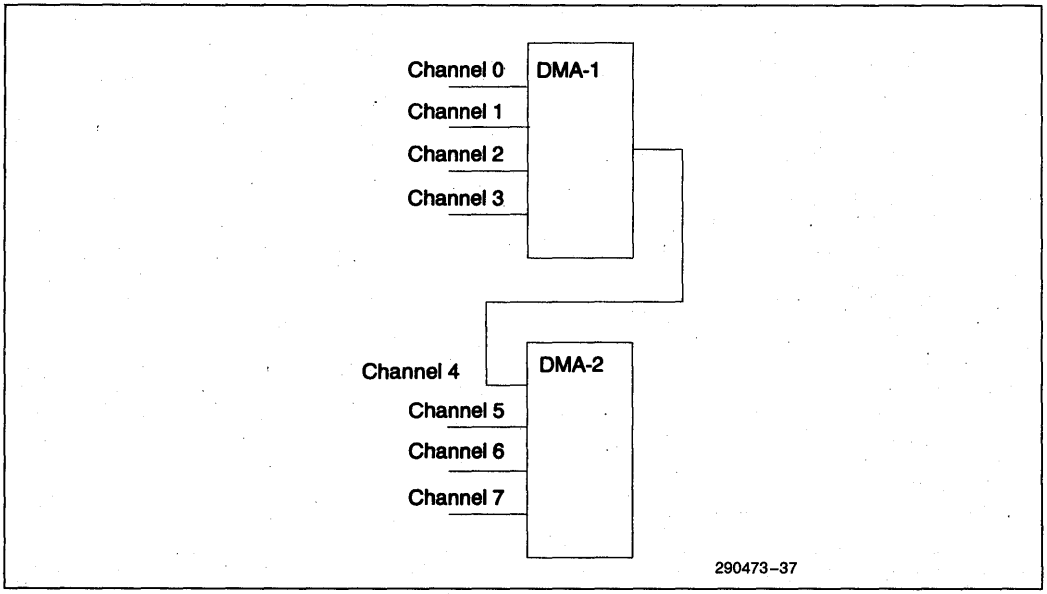


Figure 5-33. Internal DMA Controller

Full 32-bit addressing is supported as an extension of the ISA-compatible specification. Each channel includes a 16-bit ISA compatible Current Register which holds the 16 least-significant bits of the 32-bit address, and an ISA compatible Low Page Register which contains the eight second most significant bits. An additional High Page Register contains the eight most significant bits of the 32-bit address.

The DMA controller also features refresh address generation, and auto-initialization following a DMA termination.

The DMA controller receives commands from the ISA Bus arbiter to perform either DMA cycles or refresh cycles. The arbiter determines which requester from among the requesting DMA slaves, the PCI Bus, and refresh should have the bus.

The DMA controller is at any time either in master mode or slave mode. In master mode, the DMA controller is either servicing a DMA slave's request for DMA cycles, or allowing a 16-bit ISA master to use the bus via a cascaded DREQ signal. In slave mode, the SIO monitors both the ISA Bus and the PCI, decoding and responding to I/O read and write commands that address its registers.

Note that a DMA device (I/O device) is always on the ISA Bus, but the memory device is either on the ISA or PCI Bus. If the memory is decoded to be on the ISA Bus, then the DMA cycle will run as a compatible cycle. If the memory is decoded to be on the PCI Bus, the cycle can run as compatible, "A", "B", or "F" type. The ISA controller will not drive a valid address for type "A", "B", and "F" DMA transfers on the ISA Bus.

When the SIO is running a DMA cycle in compatible timing mode, the SIO will drive the MEMR# or MEMW# strobes if the address is less than 16 Mbytes (00000000h - 00FFFFFFh). These memory strobes will be generated regardless of whether the cycle is decoded for PCI or ISA memory. The SMEMR# and SMEMW# will be generated if the address is less than 1 Mbytes (00000000h - 000FFFFFh). If the address is greater than 16 Mbytes (01000000h - FFFFFFFFh) the MEMR# or MEMW# strobe will not be generated in order to avoid aliasing problems. For type A, B, and F timing mode DMA cycles, the SIO will only generate the MEMR# or MEMW# strobe when the address is decoded for ISA memory. When this occurs, the cycle converts to compatible mode timing.

During DMA cycles, the ISA controller drives AEN high to prevent the I/O devices from misinterpreting the DMA cycle as a valid I/O cycle. The BALE signal is also driven high during DMA cycles. Also, during DMA memory read cycles to the PCI Bus, the SIO will return all 1's to the ISA Bus if the PCI cycle is either target-aborted or does not respond.

5.4.2 DMA TRANSFER MODES

The channels can be programmed for any of four transfer modes. The transfer modes include single, block, demand, or cascade. Each of the three active transfer modes (single, block, and demand), can perform three different types of transfers (read, write, or verify). Note that Memory to Memory transfers are not supported by the SIO.

5.4.2.1 Single Transfer Mode

In single transfer mode, the DMA is programmed to make one transfer only. The byte/word count will be decremented and the address decremented or incremented following each transfer. When the byte/word count "rolls over" from zero to FFFFFFh, or an external EOP is encountered, a Terminal Count (TC) will cause an auto-initialize, if the channel has been programmed to do so.

DREQ must be held active until DACK becomes active in order to be recognized. If DREQ is held active throughout the single transfer, the bus will be released after a single transfer. With DREQ asserted high, the DMA I/O device will re-arbitrate for the bus. Upon winning the bus, another single transfer will be performed. This allows other ISA bus masters a chance to acquire the bus.

5.4.2.2 Block Transfer Mode

In Block Transfer mode, the DMA is activated by DREQ to continue making transfers during the service until a TC, caused by either a byte/word count going to FFFFFFh or an external EOP, is encountered. DREQ need only be held active until DACK becomes active. If the channel has been programmed for it, an autoinitialization will occur at the end of the service. In this mode, it is possible to lock out other devices for a period of time (including refresh) if the transfer count is programmed to a large number and ISA compatible timing is selected. Block transfer mode can effectively be used with Type "A", Type "B", or Type "F" timing since the channel can be interrupted through the 4-sec time-out mechanism, and other devices (or refresh) can arbitrate for and win the bus. See Section 5.4.5 on the ISA Bus Arbiter for a detailed description of the 4-μsec time-out mechanism.

5.4.2.3 Demand Transfer Mode

In Demand Transfer mode, the DMA channel is programmed to continue making transfers until a TC (Terminal Count) is encountered or an external EOP is encountered, or until the DMA I/O device releases DREQ. Thus, transfers may continue until the I/O

device has exhausted its data capacity. After the I/O device catches up, the DMA service is re-established when the DMA I/O device reasserts the channel's DREQ. During the time between services when the system is allowed to operate, the intermediate values of address and byte/word count are stored in the DMA controller Current Address and Current Byte/Word Count Registers. A TC can cause an autoinitialize at the end of the service, if the channel has been programmed for it.

5.4.2.4 Cascade Mode

This mode is used to cascade more than one DMA controller together for simple system expansion. This allows the DMA requests of the additional device to propagate through the priority network circuitry of the preceding device. The priority chain is preserved and the new device must wait for its turn to acknowledge requests. Within the SIO architecture, channel 0 of DMA controller two (DMA-2, Ch 4) is used to cascade DMA controller one (DMA-1) to provide a total of seven DMA channels. Channel 0 on DMA-2 (labeled Ch 4 in this document) connects the second half of the DMA system. This channel is not available for any other purpose.

In Cascade Mode, the DMA controller will respond to DREQ with DACK, but the SIO will not drive IOR#, IOW#, MEMR#, MEMW#, LA[23:17], SA[19:0], and SBHE#.

Cascade mode is also used to allow direct access of the system by 16-bit bus masters. These devices use the DREQ and DACK signals to arbitrate for the ISA Bus. The ISA master asserts its ISA master request line (DREQ[x]) to the DMA internal arbiter. If the ISA master wins the arbitration, the SIO responds with an ISA master acknowledge (DACK[x]) signal active. Upon sampling the DACK[x] line active, the ISA master asserts MASTER# and takes control of the ISA Bus. The ISA master has control of the ISA Bus and may run cycles until it negates the MASTER# and DREQ[x] signals.

5.4.3 DMA TRANSFER TYPES

Each of the three active transfer modes (Single, Block, or Demand) can perform three different types of transfers. These are Read, Write and Verify.

5.4.3.1 Write Transfers

Write transfers move data from an ISA I/O device to memory located either on the ISA Bus or the PCI Bus. The SIO will activate the IOR# command and the appropriate PCI and ISA control signals to indicate a memory write, depending on where the memory is located. If the memory is located on the ISA Bus, then the SIO will generate the MEMW# command. Data steering will be used to steer the data to the correct byte lane during these DMA transfers.

The DMA device (I/O device) is either an 8 or 16-bit device and is located on the ISA Bus. The DMA device size is programmable for each channel.

5.4.3.2 Read Transfers

Read transfers move data from memory (on the ISA or PCI Buses) to an ISA I/O device. The SIO will activate the IOW# command and the appropriate PCI or ISA control signals to indicate a memory read, depending on where the memory is located. If the memory is on the ISA Bus, then the SIO will generate the MEMR# command signal. Data steering will be used to steer the data to the correct byte lane during these DMA transfers.

5.4.3.3 Verify Transfer

Verify transfers are pseudo transfers. The DMA controller operates as in read or write transfers, generating addresses and producing TC, etc. However, the SIO does not activate the memory and I/O control lines. Only the DACK lines will go active. The SIO asserts the appropriate DACK signal for nine SYSCLKs. If Verify transfers are repeated during Block or Demand DMA requests, each additional pseudo transfer will add eight SYSCLKs. The DACK lines will not be toggled for repeated transfers.

5.4.4 DMA TIMINGS

ISA Compatible timing is provided for DMA slave devices. Three additional timings are provided for I/O slaves capable of running at faster speeds. These timings are referred to as Type "A", Type "B", and Type "F".

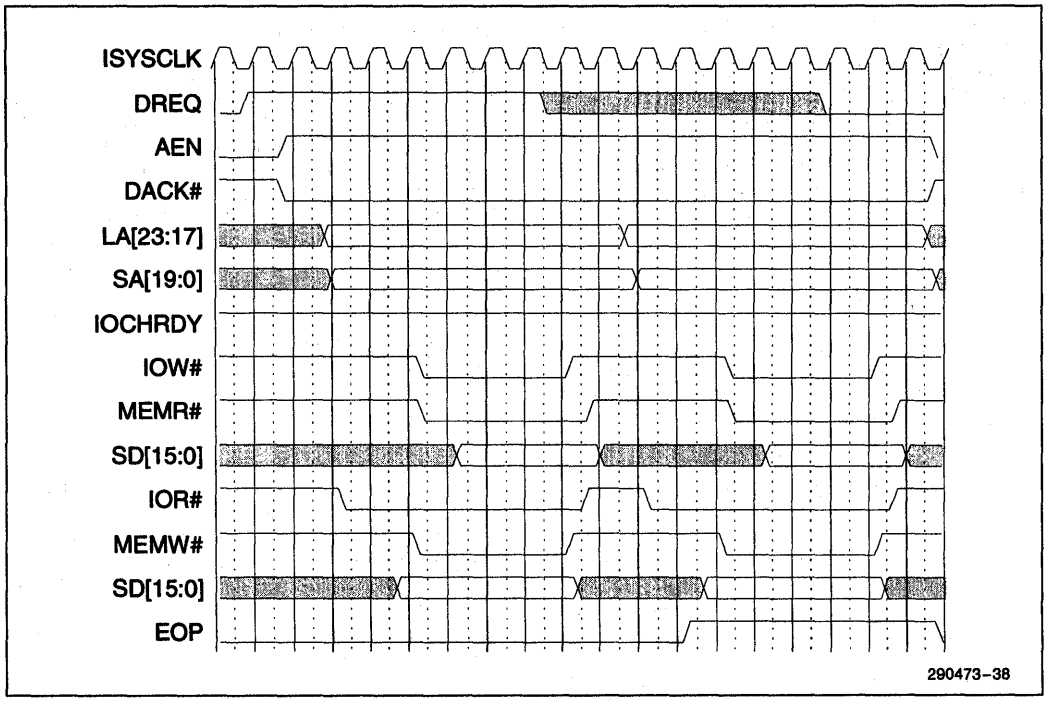


Figure 5-34. Compatible DMA Transfer (8 SYSCLK)

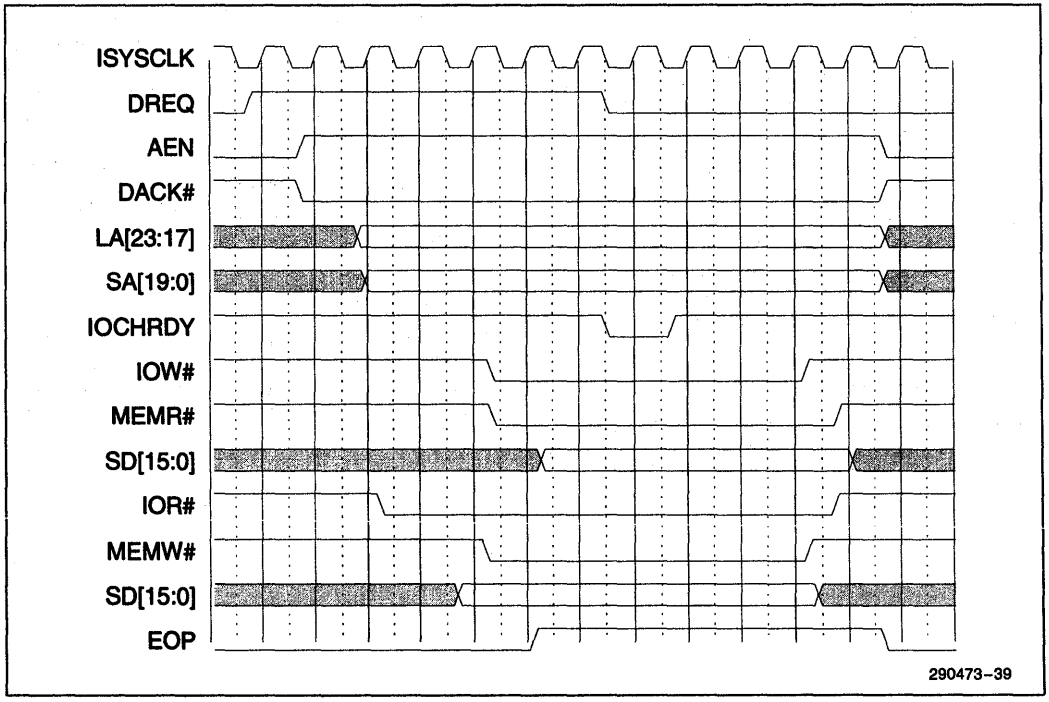


Figure 5-35. Compatible DMA Transfer (with 1 Wait State)

5.4.4.1 Compatible Timing

Compatible timing runs at 8 SYSCLKs during the repeated portion of a Block or Demand mode transfer.

5.4.4.2 Type "A" Timing

Type "A" timing is provided to allow shorter cycles to PCI memory. Type "A" timing runs at 6 SYSCLKs (720 nsec/cycle) during the repeated portion of a block or demand mode transfer. This timing assumes an 8.33 Mhz SYSCLK. Type "A" timing varies from compatible timing primarily in shortening the memory operation to the minimum allowed by system memory. The I/O portion of the cycle (data setup on write, I/O read access time) is the same as with compatible cycles. The actual active command time is shorter, but it is expected that the DMA devices which provide the data access time or write data setup time should not require excess IOR# or IOW# command active time. Because of this, most ISA DMA devices should be able to use type "A" timing.

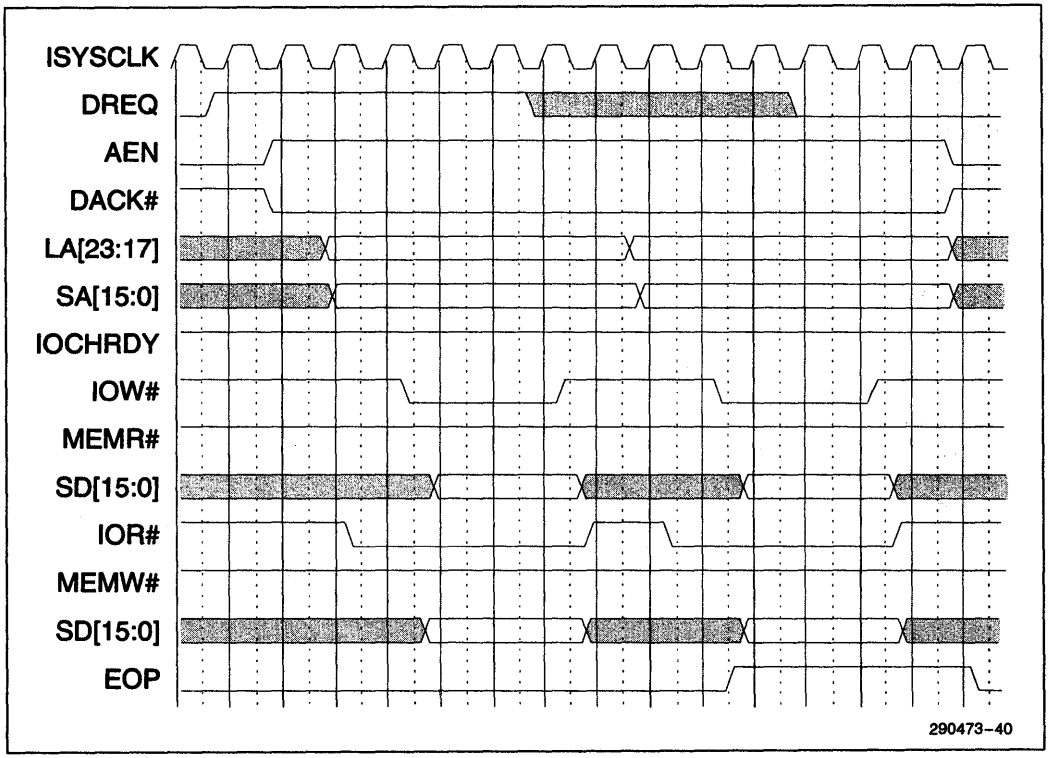


Figure 5-36. Type "A" DMA Transfer (6 SYSCLK)

5.4.4.3 Type "B" Timing

Type "B" timing is provided for 8/16-bit ISA DMA devices which can accept faster I/O timing. Type "B" only works with PCI memory. Type "B" timing runs at 5 SYSCLKs (600 nsec/cycle) during the repeated portion of a Block or Demand mode transfer. This timing assumes an 8.33 Mhz SYSCLK. Type "B" timing requires faster DMA slave devices than compatible timing in that the cycles are shortened so that the data setup time on I/O write cycles is shortened and the I/O read access time is required to be faster. Some of the current ISA devices should be able to support type "B" timing, but these will probably be more recent designs using relatively fast technology.

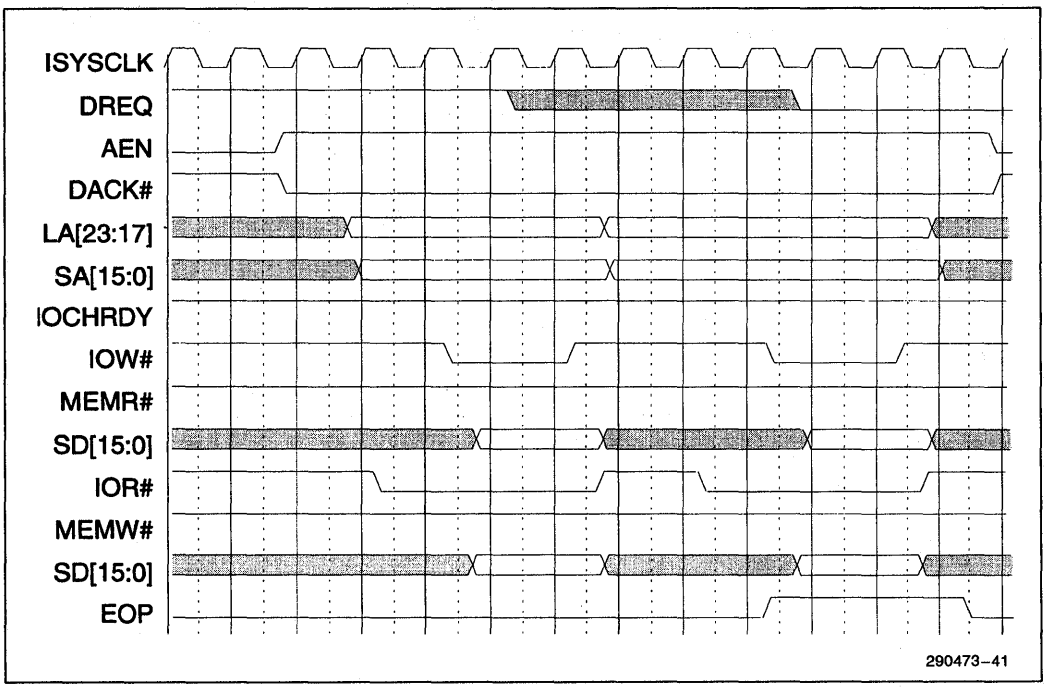


Figure 5-37. Type "B" DMA Transfer (5 SYSCLK)

5.4.4.4 Type "F" Timing

Type "F" timing provides high performance DMA transfer capability. These transfers are mainly for fast I/O devices (i.e. IDE devices). Type "F" timing runs at 3 SYSCLKs (360 nsec/cycle) during the repeated portion of a Block or Demand mode transfer.

5.4.4.5 DREQ And DACK# Latency Control

The SIO DMA arbiter maintains a minimum DREQ to DACK# latency on DMA channels programmed to operate in compatible timing mode. This is to support older devices such as the 8272A. The DREQs are delayed by eight SYSCLKs prior to being seen by the arbiter logic. Software requests will not have this minimum request to DACK# latency.

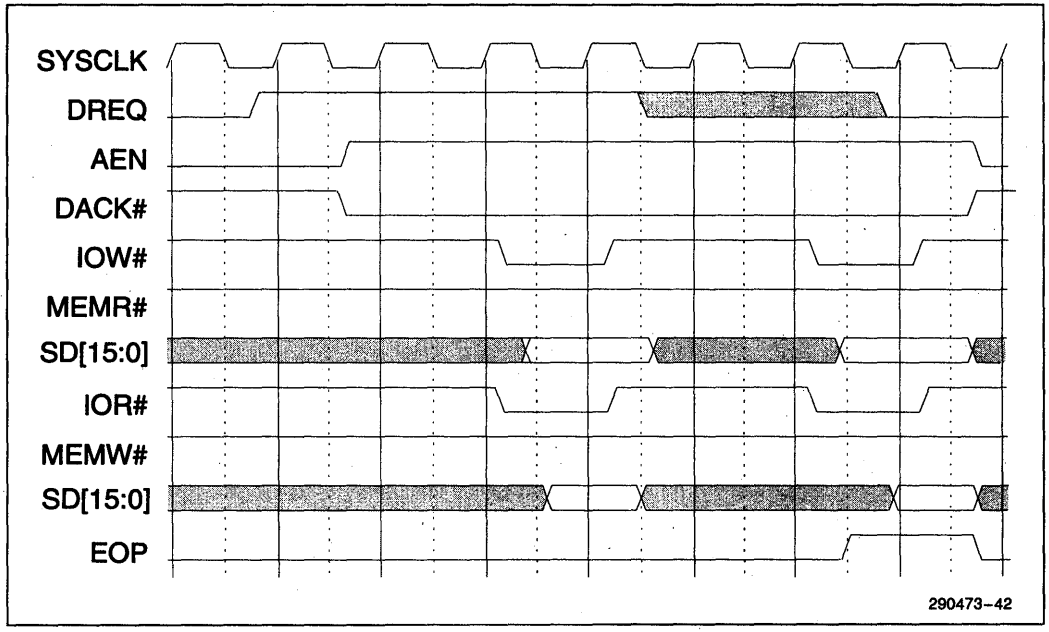


Figure 5-38. Type "F" DMA Transfer (3 SYSCLK)

5.4.5 ISA BUS/DMA ARBITRATION

The ISA Bus arbiter evaluates requests for the ISA Bus coming from several different sources. The DMA unit, the refresh counter, and the PCI Bus (primarily the Host CPU) may all request access to the ISA Bus. Additionally, 16-bit ISA masters may request the bus through a cascaded DMA channel (see the Cascade Mode description in Section 5.4.2.4).

The SIO ISA arbiter uses a three-way rotating priority arbitration method. At each level, the devices which are considered equal are given a rotating priority. On a fully loaded bus, the order in which the devices are granted bus access is independent of the order in which they assert a bus request. This is because devices are serviced based on their position in the rotation. The arbitration scheme assures that DMA channels access the bus with minimal latency.

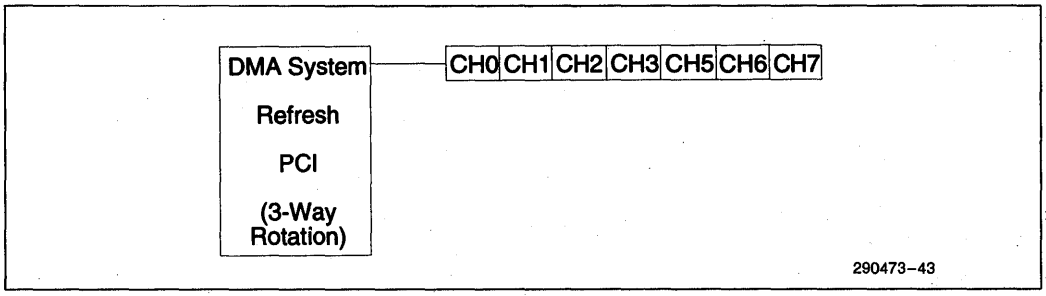


Figure 5-39. ISA Arbiter with DMA in Fixed Priority

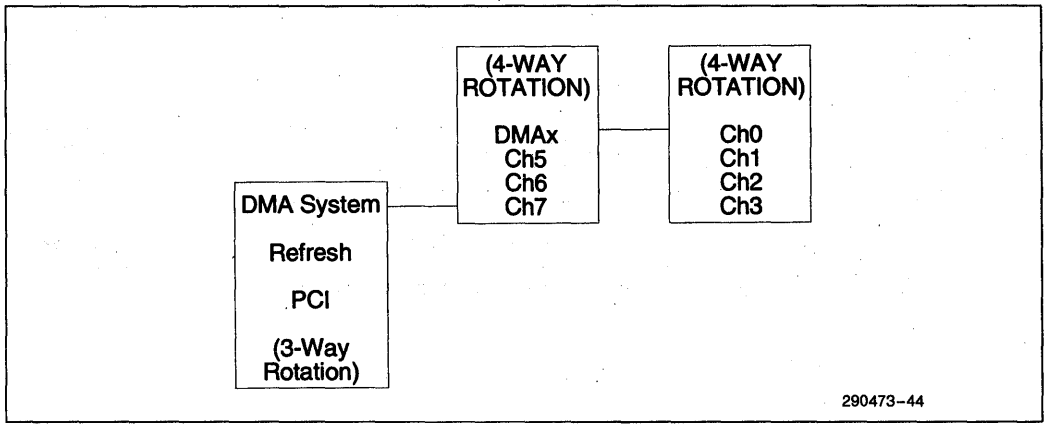


Figure 5-40. ISA Arbiter with DMA in Rotating Priority

5.4.5.1 Channel Priority

For priority resolution the DMA consists of two logical channel groups: channels 0-3 and channels 4-7 (see Figure 5-33). Each group may be in either fixed or rotate mode, as determined by the DMA Command Register.

For prioritization purposes, the source of the DMA request is transparent. DMA I/O slaves normally assert their DREQ line to arbitrate for DMA service. However, a software request for DMA service can be presented through each channel's DMA Request Register. A software request is subject to the same prioritization as any hardware request. Please see the detailed register description in Section 3.5 for Request Register programming information.

Fixed Priority
The initial fixed priority structure is as follows:

Table 5-9. Initial Fixed Priority Structure

High priority.....Low priority
(0, 1, 2, 3) 5, 6, 7

The fixed priority ordering is 0, 1, 2, 3, 5, 6, and 7. In this scheme, Channel 0 has the highest priority, and channel 7 has the lowest priority. Channels [3:0] of DMA-1 assume the priority position of Channel 4 in DMA-2, thus taking priority over channels 5, 6, and 7.

Rotating Priority
Rotation allows for "fairness" in priority resolution. The priority chain rotates so that the last channel serviced is assigned the lowest priority in the channel group (0-3, 5-7).

Channels 0-3 rotate as a group of 4. They are always placed between Channel 5 and Channel 7 in the priority list.

Channel 5-7 rotate as part of a group of 4. That is, channels (5-7) form the first three positions in the rotation, while channel group (0-3) comprises the fourth position in the arbitration.

Table 5-10. demonstrates rotation priority.

Table 5-10. Rotating Priority Example

Programmed Mode	Action	Priority High.....Low
Group (0-3) is in rotation mode Group (4-7) is in fixed mode.	1) Initial Setting 2) After servicing channel 2 3) After servicing channel 3	(0, 1, 2, 3), 5, 6, 7 (3, 0, 1, 2), 5, 6, 7 (0, 1, 2, 3), 5, 6, 7
Group (0-3) in rotation mode Group (4-7) is in rotation mode (note that the first servicing of channel 0 caused double rotation).	1) Initial Setting 2) After servicing channel 0 3) After servicing channel 5 4) After servicing channel 6 5) After servicing channel 7	(0, 1, 2 3), 5, 6, 7 5, 6, 7, (1, 2, 3, 0) 6, 7, (1, 2, 3, 0), 5 7, (1, 2, 3, 0), 5, 6 (1, 2, 3, 0), 5, 6, 7

5.4.5.2 DMA Preemption in Performance Timing Modes

A DMA slave device that is not programmed for compatible timing will be preempted from the bus by another device that requests use of the bus. This will occur, regardless of the priority of the pending request. For DMA devices not using compatible timing mode, the DMA controller stops the DMA transfer and releases the bus within 32 BCLK (4-μsec) of a preemption. Upon the expiration of the 4-μsec timer, the DACK will be inactivated after the current DMA cycle has completed. The bus will then be arbitrated for and granted to the highest priority requester. This feature allows flexibility in programming the DMA for long transfer sequences in a performance timing mode while guaranteeing that vital system services such as refresh are allowed access to the ISA Bus.

The 4-μsec timer is not used in compatible timing mode. It is only used for DMA channels programmed for Type "A", Type "B", or Type "F" timing. It is also not used for 16-bit ISA masters cascaded through the DMA DREQ lines.

If the DMA channel that was preempted by the 4-μsec timer was operating in Block Mode, an internal bit will be set so that the channel will be arbitrated for again, independent of the state of DREQ.

5.4.5.3 Arbitration During Non-Maskable Interrupts

If a non-maskable interrupt (NMI) is pending, and the CPU is requesting the bus, then the DMA controller will be bypassed each time it comes up for rotation. This will give the CPU the bus bandwidth it requires to process the interrupt as fast as possible.

5.4.5.4 Programmable Guaranteed Access Time Mode (GAT Mode)

The PCI Arbiter Register contains a bit for configuring the SIO in "Guaranteed Access Time Mode" (GAT Mode). This mode guarantees that the 2.5 μsec CHRDY time-out specification for ISA masters running cycles to PCI will not be exceeded. When an ISA master or DMA slave arbitrates for the ISA Bus, and the SIO is configured in Guaranteed Access Time Mode, the MEMREQ# pin will be asserted by the PCI arbiter in order to gain ownership of main memory. The arbitration for the PCI and then the main memory bus must be completed prior to granting the DACK# to the ISA master or DMA slave. A MEMACK# signal to the SIO indicates that the SIO now owns main memory and can grant the DACK# to the ISA master or DMA slave. A detailed description is contained in Section 5.2.3.2.

5.4.6 REGISTER FUNCTIONALITY

Please see Section 4.2 for detailed information on register programming, bit definitions, and default values/functions of the DMA registers after a PCIRST#.

DMA Channel 4 is used to cascade the two DMA controllers together and should not be programmed for any mode other than cascade. The DMA Channel Mode Register for channel 4 will default to cascade mode. Special attention should also be take when programming the Command and Mask Registers as related to channel 4.

5.4.6.1 Address Compatibility Mode

Whenever the DMA is operating in address compatibility mode, the addresses do not increment or decrement through the High and Low Page Registers, and the High Page Register is set to 00h. This is compatible with the 82C37 and Low Page Register implementation used in the PC-AT. This mode is set when any of the lower three address bytes of a channel are programmed. If the upper byte of a channel's address is programmed last, the channel will go into extended address mode. In this mode, the high byte may be any value and the address will increment or decrement through the entire 32-bit address.

After PCIRST# is negated, all channels will be set to address compatibility mode. The DMA Master Clear command will also reset the proper channels to address compatibility mode.

5.4.6.2 Summary of DMA Transfer Sizes

Table 5-11 lists each of the DMA device transfer sizes. The column labeled "Current Byte/Word Count Register" indicates that the register contents represents either the number of bytes to transfer or the number of 16-bit words to transfer. The column labeled "Current Address Increment/Decrement" indicates the number added to or taken from the Current Address register after each DMA transfer cycle. The DMA Channel Mode Register determines if the Current Address Register will be incremented or decremented.

Table 5-11. DMA Transfer Size

DMA Device Date Size And Word Count	Current Byte/Word Count Register	Current Address Increment/Decrement
8-Bit I/O, Count By Bytes	Bytes	1
16-Bit I/O, Count By Words (Address Shifted)	Words	1
16-Bit I/O, Count By Bytes	Bytes	2

5.4.6.3 Address Shifting When Programmed for 16-Bit I/O Count by Words

To maintain compatibility with the implementation of the DMA in the PC AT which used the 82C37, the DMA will shift the addresses when the DMA Channel Extended Mode Register is programmed for, or defaulted to, transfers to/from a 16-bit device count-by-words. Note that the least significant bit of the Low Page Register is dropped in 16-bit shifted mode. When programming the Current Address Register when the DMA channel is in this mode, the Current Address must be programmed to an even address with the address value shifted right by one bit. The address shifting is as follows:

Table 5-12. Address Shifting in 16-Bit I/O DMA Transfers

Output Address	8-Bit I/O Programmed Address	16-Bit I/O Programmed Address (Shifted)	16-Bit I/O Programmed Address (No Shift)
A0 A[16:1] A[31:17]	A0 A[16:1] A[31:17]	0 A[15:0] A[31:17]	A0 A[16:01] A[31:17]

Note that the least significant bit of the Low Page Register is dropped in 16-bit shifted mode.

5.4.6.4 Autoinitialize

By programming a bit in the DMA Channel Mode Register, a channel may be set up as an autoinitialize channel. During Autoinitialize initialization, the original values of the Current Page, Current Address and Current Byte/Word Count Registers are automatically restored from the Base Page, Address, and Byte/Word Count Registers of that channel following TC. The Base Registers are loaded simultaneously with the Current Registers by the microprocessor and remain unchanged throughout the DMA service. The mask bit is not set when the channel is in autoinitialize. Following Autoinitialize, the channel is ready to perform another DMA service, without CPU intervention, as soon as a valid DREQ is detected.

5.4.7 SOFTWARE COMMANDS

There are three additional special software commands which can be executed by the DMA controller. The three software commands are:

1) Clear Byte Pointer Flip-Flop
2) Master Clear
3) Clear Mask Register

They do not depend on any specific bit pattern on the data bus.

5.4.7.1 Clear Byte Pointer Flip-Flop

This command is executed prior to writing or reading new address or word count information to/from the DMA controller. This initializes the flip-flop to a known state so that subsequent accesses to register contents by the microprocessor will address upper and lower bytes in the correct sequence.

When the Host CPU is reading or writing DMA registers, two Byte Pointer Flip-Flops are used; one for channels 0-3 and one for channels 4-7. Both of these act independently. There are separate software commands for clearing each of them (0Ch for channels 0-3, 0D8h for channels 4-7).

5.4.7.2 DMA Master Clear

This software instruction has the same effect as the hardware reset. The Command, Status, Request, and Internal First/Last Flip-Flop Registers are cleared and the Mask Register is set. The DMA controller will enter the idle cycle.

There are two independent master clear commands, 0Dh which acts on channels 0-3, and 0DAh which acts on channels 4-7.

5.4.7.3 Clear Mask Register

This command clears the mask bits of all four channels, enabling them to accept DMA requests. I/O port 00Eh is used for channels 0-3 and I/O port 0DCh is used for channels 4-7.

5.4.8 TERMINAL COUNT/EOP SUMMARY

This is a summary of the events that will happen as a result of a terminal count or external EOP when running DMA in various modes.

Table 5-13. Terminal Count/EOP Summary Table

Conditions AUTOINIT	No		Yes	
Event Word Counter Expired	Yes	X	Yes	X
EOP Input	X	Asserted	X	Asserted
Result Status TC	set	set	set	set
Mask	set	set	-	-
SW Request	clr	clr	clr	clr
Current Register	-	-	load	load

load = Load Current From Base
"-" = No Change
X = Don't Care
clr = Clear

5.4.9 ISA REFRESH CYCLES

Refresh cycles are generated by two sources: the refresh controller inside the SIO component or by ISA bus masters other than the SIO. The ISA bus controller will enable the address lines SA[15:0] so that when MEMR# goes active, the entire ISA system memory is refreshed at one time. Memory slaves on the ISA Bus must not drive any data onto the data bus during the refresh cycle.

Counter 1 in the timer register set should be programmed to provide a request for refresh about every 15s. See section 5.7.1.2 for more details.

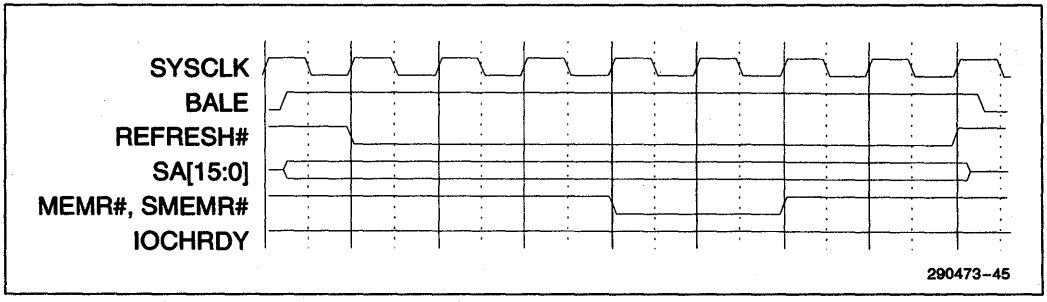


Figure 5-41. SIO Initiated Default Refresh Cycle

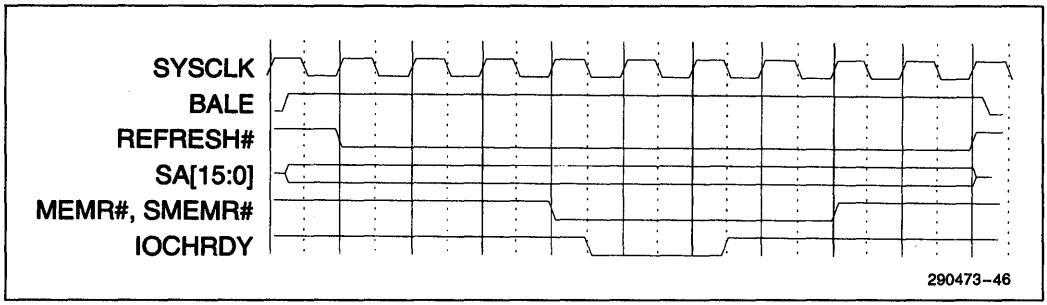


Figure 5-42. SIO Initiated Extended Refresh Cycle

5.4.9.1 Refresh Cycles Initiated By the SIO

This refresh cycle is initiated by the refresh logic inside the SIO. The SIO asserts REFRESH# to indicate a refresh cycle. The SIO then drives the address lines SA[15:0] onto the ISA Bus and generates MEMR# and SMEMR#. The SIO drives AEN and BALE high for the entire refresh cycle. The memory device may extend this refresh cycle by pulling IOCHRDY low.

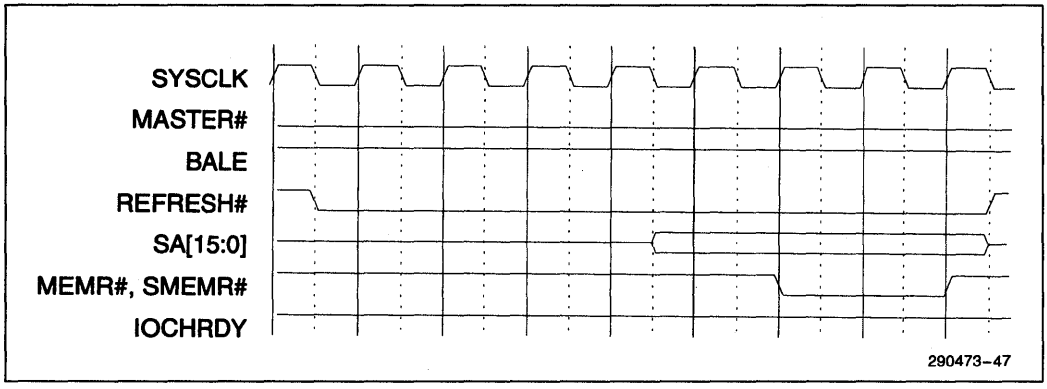


Figure 5-43. Bus Master-Initiated Default Refresh Cycle

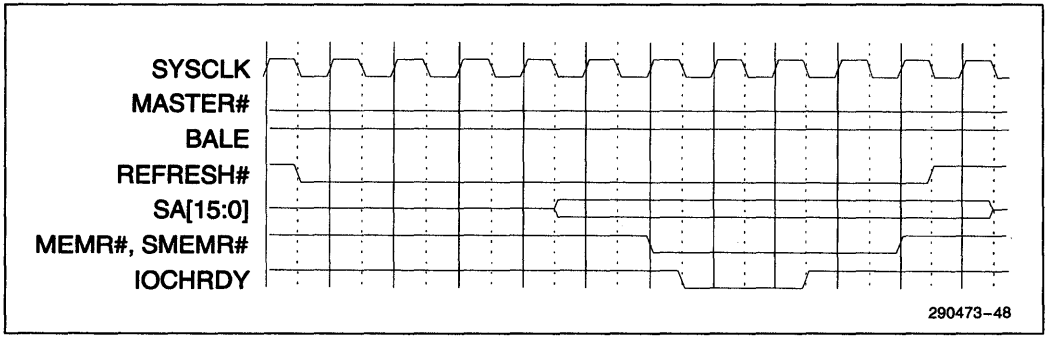


Figure 5-44. Bus Master-Initiated Extended Refresh Cycle

5.4.9.2 Refresh Cycles Initiated By an ISA Bus Master

If an ISA Bus master holds the ISA Bus longer than 15 μs, the ISA master must initiate memory refresh cycles. If the ISA Bus master initiates a refresh cycle before it relinquishes the bus, it floats the address lines and control signals and asserts the REFRESH# to the SIO. The SIO drives address lines SA[15:0] and MEMR# and SMEMR# onto the ISA Bus. BALE is driven high and AEN signal is driven low for the entire refresh cycle.

5.4.10 SCATTER/GATHER DESCRIPTION

NOTE:
The Base Registers and Current Registers described in the Scatter/Gather sections are hidden registers that are not accessible by software.

Scatter/Gather (S/G) provides the capability of transferring multiple buffers between memory and I/O without CPU intervention. In Scatter/Gather, the DMA can read the memory address and word count from an array of buffer descriptors, located in system memory (ISA or PCI), 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 transferred.

The S/G Command and Status Registers are used to control the operational aspect of S/G transfers. The SGD Table Pointer Register holds the address of the next buffer descriptor in the SGD Table.

The next buffer descriptor is fetched from the SGD Table by a DMA read transfer. DACK# will not be asserted for this transfer because the IO device is the SIO itself. The SIO will fetch the next buffer descriptor from either PCI memory or ISA memory, depending on where the SGD Table is located. If the SGD table is located in PCI memory, the memory read will use the line buffer to temporarily store the PCI read before loading it into the DMA S/G registers. The line buffer mode (8 byte or single transaction) for the S/G fetch operation will be the same as what is set for all DMA operations. If set in 8 byte mode, the SGD Table fetches will be PCI burst memory reads. The SGD Table PCI cycle fetches are subject to all types of PCI cycle termination (retry, disconnect, target-abort, master-abort). The fetched SGD Table data is subject to normal line buffer coherency management and invalidation. EOP will be asserted at the end of the complete link transfer.

To initiate a typical DMA Scatter/Gather transfer between memory and an I/O device, the following steps are required:

1) Software prepares a SGD Table in system memory. Each SGD is 8 bytes long and consists of an address pointer to the starting address and the transfer count of the memory buffer to be transferred. In any given SGD Table, two consecutive SGDs are offset by 8-bytes and are aligned on a 4-byte boundary. Each Scatter-Gather Descriptor for the linked list contains the following information:
   a) Memory Address (buffer start) 4 bytes
   b) Byte Count (buffer size) 3 bytes
   c) End of Link List 1 bit (MSB)

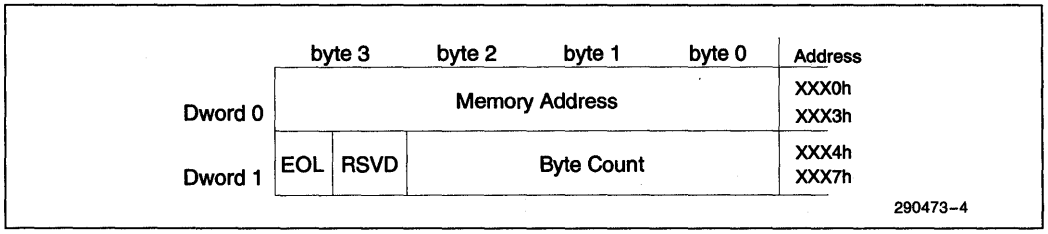


Figure 5-45. SGD Format

2) Initialize the DMA Channel Mode and DMA Channel Extended Mode Registers with transfer specific information like 8/16 bit I/O device, Transfer Mode, Transfer Type, etc.
3) Software provides the starting address of the SGD Table by loading the SGD Table Pointer Register.
4) Engage the Scatter/Gather function by writing a Start command to the S/G Command Register.
5) The Mask register should be cleared as the last step of programming the DMA register set. This is to prevent the DMA from starting a transfer with a partially loaded command description.
6) Once the register set is loaded and the channel is unmasked, the DMA will generate an internal request to fetch the first buffer from the SGD Table.

After the above steps are finished, the DMA will then respond to DREQ or software requests. The first transfer from the first buffer moves the memory address and word count from the Base register set to the Current register set. As long as S/G is active and the Base register set is not loaded and the last buffer has not been fetched, the channel will generate a request to fetch a reserve buffer into the Base register set. The reserve buffer is loaded to minimize latency problems going from one buffer to another. Fetching a reserve buffer has a lower priority than completing DMA transfers for the channel.

The DMA controller will terminate a Scatter/Gather cycle by detecting an End of List (EOL) bit in the SGD Table. After the EOL bit is detected, the channel transfers the buffers in the Base and Current register sets, if they are loaded. At terminal count the channel asserts EOP or IRQ13, depending on its programming and set the terminate bit in the S/G Status Register. If the channel asserted IRQ13, then the appropriate bit is set in the Scatter/Gather Interrupt Status Register. The active bit in the S/G Status Register will be reset and the channel's Mask bit will be set.

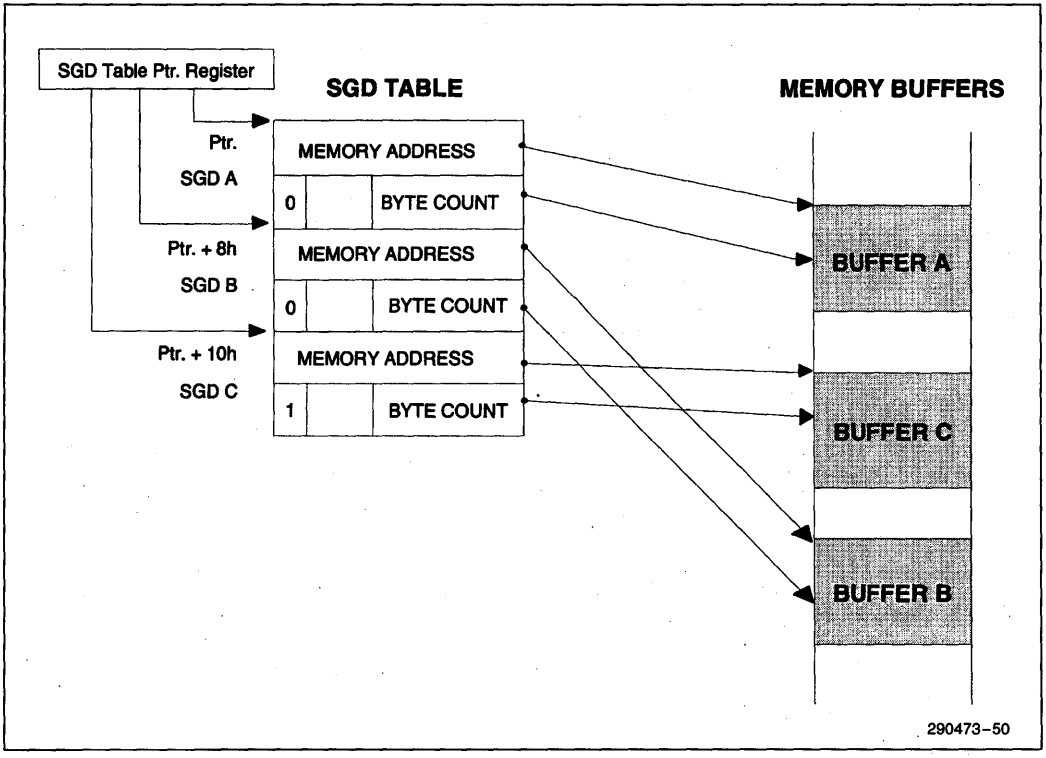


Figure 5-46. Link List Example

5.5 Address Decoding

The SIO contains two address decoders; one to decode PCI master cycles and one to decode DMA/ISA master cycles. Two decoders are required to support the PCI and ISA Busses running concurrently. The PCI address decoder decodes the address from the multiplexed PCI address/data bus. The DMA/ISA master address decoder decodes the address from the ISA address bus for DMA and ISA master cycles. The address decoders determine how the cycle is handled.

5.5.1 PCI ADDRESS DECODER

PCI address decoding is always a function of AD[31:2]. The information contained in the two low order bits (AD[1:0]) varies for memory, I/O, and configuration cycles.

For I/O cycles, AD[31:0] are all decoded to provide a byte address. The byte enables determine which byte lanes contain valid data. The SIO requires that PCI accesses to byte-wide internal registers must assert only one byte enable.

For memory cycles, accesses are decoded as dword accesses. This means that AD[1:0] are ignored for decoding memory cycles. The byte enables are used only to determine which byte lanes contain valid data.

For configuration cycles, DEVSEL# is a function of IDSEL# and AD[1:0]. DEVSEL# is generated only when AD[1:0] are both zero. If either AD[1:0] are non-zero, the cycle is ignored by the SIO. Individual bytes of a configuration register can be accessed with the byte enables. A particular configuration register is selected using AD[7:2]. Again, the byte enables determine which byte lanes contain valid data.

All PCI cycles decoded in one of the following ways result in the SIO generating DEVSEL#. The PCI master cycle decoder decodes the following addresses based on the settings of the relevant configuration registers:

SIO I/O Addresses: Positively decodes I/O addresses for registers contained within the SIO (exceptions : 60h, 92h, 3F2h, 372h, and F0h).

BIOS Memory Space: Positively decodes BIOS memory space.

MEMCS# Address Decoding: Decodes memory addresses that reside on the other side of the Host bridge and generates the MEMCS# signal. (SIO does not generate DEVSEL# in this case). The address range(s) used for this decoding is selected via the MCSCON, MEMCSBOH, MEMCSTOH, MCSTOM, MAR1, MAR2, and MAR3 Registers (see Section 4.1).

Subtractively Decoding Cycles to ISA: Subtractively decodes cycles to the ISA Bus. Accesses to registers 60h, 92h, 3F2h, 372h, and F0h are also subtractively decoded to the ISA Bus.

One of the PCI requirements 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 it is enabled through configuration. The SIO is an exception to this, since it controls access to the BIOS boot code. All addresses decoded by the PCI address decoder, that are enabled after chip reset, are accessible immediately after power-up.

5.5.1.1 SIO I/O Addresses

These addresses are the internal, non-configuration SIO register locations and are shown in the SIO Address Decoding Table, Table 5-14. Note that the Configuration Registers, listed in Table 4-1, are accessed with PCI configuration cycles as described in section 5.1.2.5

In general, PCI accesses to the internal SIO registers will not be broadcast to the ISA Bus. However, PCI accesses to addresses 70h, 60h, 92h, 3F2h, 372h, and F0h are exceptions. Read and write accesses to these SIO locations are broadcast onto the ISA Bus. PCI master accesses to SIO registers will be retried if the ISA Bus is owned by an ISA master or the DMA controller. All of the registers are 8 bit registers. Accesses to these registers must be 8 bit accesses. Target-abort is issued by the SIO when the internal SIO non-configuration registers are the target of a PCI master I/O cycle and more than one byte enable is active. Refer to Table 5-14 for the SIO Address Decoding Map.

Accesses to the BIOS Timer Register (78h–7Bh) are broadcast to the ISA bus only if this register is disabled. If this register is enabled, the cycle is not broadcast to the ISA bus.

The address decoding logic includes the read/write cycle type. For example, read cycles to write only registers are not positively decoded and get forwarded to the ISA Bus via subtractive decoding.

Table 5-14. SIO Address Decoding

Address	Address				Type	Name	Block
	FEDC	BA98	7654	3210			
0000h	0000	0000	000x	0000	r/w	DMA1 CH0 Base and Current Address	DMA
0001h	0000	0000	000x	0001	r/w	DMA1 CH0 Base and Current Count	DMA
0002h	0000	0000	000x	0010	r/w	DMA1 CH1 Base and Current Address	DMA
0003h	0000	0000	000x	0011	r/w	DMA1 CH1 Base and Current Count	DMA
0004h	0000	0000	000x	0100	r/w	DMA1 CH2 Base and Current Address	DMA
0005h	0000	0000	000x	0101	r/w	DMA1 CH2 Base and Current Count	DMA
0006h	0000	0000	000x	0110	r/w	DMA1 CH3 Base and Current Address	DMA
0007h	0000	0000	000x	0111	r/w	DMA1 CH3 Base and Current Count	DMA
0008h	0000	0000	000x	1000	r/w	DMA1 Status(r) Command(w) register	DMA
0009h	0000	0000	000x	1001	wo	DMA1 Write Request register	DMA
000Ah	0000	0000	000x	1010	wo	DMA1 Write Single Mask Bit	DMA
000Bh	0000	0000	000x	1011	wo	DMA1 Write Mode register	DMA
000Ch	0000	0000	000x	1100	wo	DMA1 Clear Byte Pointer	DMA
000Dh	0000	0000	000x	1101	wo	DMA1 Master Clear	DMA
000Eh	0000	0000	000x	1110	wo	DMA1 Clear Mask register	DMA
000Fh	0000	0000	000x	1111	r/w	DMA1 Read/Write All Mask Register Bits	DMA
0020h	0000	0000	001x	xx00	r/w	INT 1 Control register	PIC
0021h	0000	0000	001x	xx01	r/w	INT 1 Mask register	PIC
0040h	0000	0000	010x	0000	r/w	Timer Counter 1 - Counter 0 Count	TC
0041h	0000	0000	010x	0001	r/w	Timer Counter 1 - Counter 1 Count	TC
0042h	0000	0000	010x	0010	r/w	Timer Counter 1 - Counter 2 Count	TC
0043h	0000	0000	010x	0011	wo	Timer Counter 1 Command Mode register	TC
0060h	0000	0000	0110	0000	r	Reset UBus IRQ12	Control
0061h	0000	0000	0110	0xx1	r/w	NMI Status and Control	Control
0070h	0000	0000	0111	0xx0	wo	CMOS RAM Address and NMI Mask reg.	Control
0078h[1]	0000	0000	0111	10xx	r/w	BIOS Timer	TC
0080h	0000	0000	100x	0000	r/w	DMA Page Register (Reserved)	DMA
0081h	0000	0000	100x	0001	r/w	DMA Channel 2 Page register	DMA
0082h	0000	0000	1000	0010	r/w	DMA Channel 3 Page register	DMA
0083h	0000	0000	100x	0011	r/w	DMA Channel 1 Page register	DMA
0084h	0000	0000	100x	0100	r/w	DMA Page Register (Reserved)	DMA
0085h	0000	0000	100x	0101	r/w	DMA Page Register (Reserved)	DMA
0086h	0000	0000	100x	0110	r/w	DMA Page Register (Reserved)	DMA
0087h	0000	0000	100x	0111	r/w	DMA Channel 0 Page register	DMA
0088h	0000	0000	100x	0100	r/w	DMA Page Register (Reserved)	DMA
0089h	0000	0000	100x	1001	r/w	DMA Channel 6 Page register	DMA

Table 5-14. SIO Address Decoding (Continued)

Address	Address				Type	Name	Block
	FEDC	BA98	7654	3210			
008Ah	0000	0000	100x	1010	r/w	DMA Channel 7 Page register	DMA
008Bh	0000	0000	100x	1011	r/w	DMA Channel 5 Page register	DMA
008Ch	0000	0000	100x	1100	r/w	DMA Page Register (Reserved)	DMA
008Dh	0000	0000	100x	1101	r/w	DMA Page Register (Reserved)	DMA
008Eh	0000	0000	100x	1110	r/w	DMA Page Register (Reserved)	DMA
008Fh	0000	0000	100x	1111	r/w	DMA low page Register Refresh	DMA
0090h	0000	0000	100x	0100	r/w	DMA Page Register (Reserved)	DMA
0092h	0000	0000	1001	0010	r/w	System Control Port	Control
0094h	0000	0000	100x	0100	r/w	DMA Page Register (Reserved)	DMA
0095h	0000	0000	100x	0101	r/w	DMA Page Register (Reserved)	DMA
0096h	0000	0000	100x	0110	r/w	DMA Page Register (Reserved)	DMA
0098h	0000	0000	100x	0100	r/w	DMA Page Register (Reserved)	DMA
009Ch	0000	0000	100x	1100	r/w	DMA Page Register (Reserved)	DMA
009Dh	0000	0000	100x	1101	r/w	DMA Page Register (Reserved)	DMA
009Eh	0000	0000	100x	1110	r/w	DMA Page Register (Reserved)	DMA
009Fh	0000	0000	100x	1111	r/w	DMA low page Register Refresh	DMA
00A0h	0000	0000	101x	xx00	r/w	INT 2 Control register	PIC
00A1h	0000	0000	101x	xx01	r/w	INT 2 Mask register	PIC
00C0h	0000	0000	1100	000x	r/w	DMA2 CH0 Base and Current Address	DMA
00C2h	0000	0000	1100	001x	r/w	DMA2 CH0 Base and Current Count	DMA
00C4h	0000	0000	1100	010x	r/w	DMA2 CH1 Base and Current Address	DMA
00C6h	0000	0000	1100	011x	r/w	DMA2 CH1 Base and Current Count	DMA
00C8h	0000	0000	1100	100x	r/w	DMA2 CH2 Base and Current Address	DMA
00CAh	0000	0000	1100	101x	r/w	DMA2 CH2 Base and Current Count	DMA
00CCh	0000	0000	1100	110x	r/w	DMA2 CH3 Base and Current Address	DMA
00CEh	0000	0000	1100	111x	r/w	DMA2 CH3 Base and Current Count	DMA
00D0h	0000	0000	1101	000x	r/w	DMA2 Status(r) Command(w) register	DMA
00D2h	0000	0000	1101	001x	wo	DMA2 Write Request register	DMA
00D4h	0000	0000	1101	010x	wo	DMA2 Write Single Mask Bit	DMA
00D6h	0000	0000	1101	011x	wo	DMA2 Write Mode register	DMA
00D8h	0000	0000	1101	100x	wo	DMA2 Clear Byte Pointer	DMA
00DAh	0000	0000	1101	101x	wo	DMA2 Master Clear	DMA
00DCh	0000	0000	1101	110x	wo	DMA2 Clear Mask register	DMA
00DEh	0000	0000	1101	111x	r/w	DMA2 Read/Write All Mask Register Bits	DMA
00F0h	0000	0000	1111	0000	wo	Coprocessor Error	Control
0372h	0000	0011	0111	0010	wo	Secondary Floppy Disk Digital Output Reg.	Control

Table 5-14. SIO Address Decoding (Continued)

Address	Address				Type	Name	Block
	FEDC	BA98	7654	3210			
03F2h	0000	0011	1111	0001	wo	Primary Floppy Disk Digital Output Register	Control
040Ah	0000	0100	0000	1010	ro	Scatter/Gather Interrupt Status Register	DMA
040Bh	0000	0100	0000	1011	wo	DMA1 Extended Mode register	DMA
0410h(1)	0000	0100	0001	0000	w	CH0 Scatter/Gather Command	DMA
0411h(1)	0000	0100	0001	0001	w	CH1 Scatter/Gather Command	DMA
0412h(1)	0000	0100	0001	0010	w	CH2 Scatter/Gather Command	DMA
0413h(1)	0000	0100	0001	0011	w	CH3 Scatter/Gather Command	DMA
0415h(1)	0000	0100	0001	0101	w	CH5 Scatter/Gather Command	DMA
0416h(1)	0000	0100	0001	0110	w	CH6 Scatter/Gather Command	DMA
0417h(1)	0000	0100	0001	0111	w	CH7 Scatter/Gather Command	DMA
0418h(1)	0000	0100	0001	1000	r	CH0 Scatter/Gather Status	DMA
0419h(1)	0000	0100	0001	1001	r	CH1 Scatter/Gather Status	DMA
041Ah(1)	0000	0100	0001	1010	r	CH2 Scatter/Gather Status	DMA
041Bh(1)	0000	0100	0001	1011	r	CH3 Scatter/Gather Status	DMA
041Dh(1)	0000	0100	0001	1101	r	CH5 Scatter/Gather Status	DMA
041Eh(1)	0000	0100	0001	1110	r	CH6 Scatter/Gather Status	DMA
041Fh(1)	0000	0100	0001	1111	r	CH7 Scatter/Gather Status	DMA
0420h(1)	0000	0100	0010	00xx	r/w	CH0 Scatter/Gather Descriptor Table Pointer	DMA
0424h(1)	0000	0100	0010	01xx	r/w	CH1 Scatter/Gather Descriptor Table Pointer	DMA
0428h(1)	0000	0100	0010	10xx	r/w	CH2 Scatter/Gather Descriptor Table Pointer	DMA
042C(1)	0000	0100	0010	11xx	r/w	CH3 Scatter/Gather Descriptor Table Pointer	DMA
0434h(1)	0000	0100	0011	01xx	r/w	CH5 Scatter/Gather Descriptor Table Pointer	DMA
0438h(1)	0000	0100	0011	10xx	r/w	CH6 Scatter/Gather Descriptor Table Pointer	DMA
043Ch(1)	0000	0100	0011	11xx	r/w	CH7 Scatter/Gather Descriptor Table Pointer	DMA
0481h	0000	0100	1000	0001	r/w	DMA CH2 High Page register	DMA
0482h	0000	0100	1000	0010	r/w	DMA CH3 High Page register	DMA
0483h	0000	0100	1000	0011	r/w	DMA CH1 High Page register	DMA
0487h	0000	0100	1000	0111	r/w	DMA CH0 High Page register	DMA
0489h	0000	0100	1000	1001	r/w	DMA CH6 High Page register	DMA
048Ah	0000	0100	1000	1010	r/w	DMA CH7 High Page register	DMA
048Bh	0000	0100	1000	1011	r/w	DMA CH5 High Page register	DMA
04D6h	0000	0100	1101	0010	wo	DMA2 Extended Mode register	DMA

NOTE:
1. The I/O address of this register is relocatable. The value shown in this table is the default address location.

5.5.1.2 BIOS Memory Space

The 128 KByte BIOS memory space is located at 000E0000h to 000FFFFFh (top of 1 MByte), and is aliased at FFFE0000h to FFFFFFFFh (top of 4 GByte) and FFEE0000h to FFEFFFFFh (top of 4 Gb-1 Mb). The aliased regions account for the CPU reset vector and the uncertainty of the state of A20GATE when a software reset occurs. This 128 Kb block is split into two 64 Kb blocks. The top 64 Kb is always enabled while the bottom 64 Kb can be enabled or disabled (the aliases automatically match). Enabling the lower 64 Kb BIOS space (000E0000h to 000EFFFFh, 896 Kb-960 Kb) results in positively decoding this region and enables the BIOSCS# signal generation. The upper 64 Kb is positively decoded only if bit 6=1 in the ISA Clock Divisor Register. Otherwise this region is subtractively decoded. Positively decoding these cycles expedites BIOS cycles to the ISA Bus. Note that both of these regions are subtractively decoded if bit 4 in the MEMCS# Control Register is set to a 1.

When PCI master accesses to the 128 Kb BIOS space at 4 Gb-1 Mb are forwarded to the ISA Bus, the LA20 line is driven to a 1 to avoid aliasing at the 15 Mb area. The 4 Gb-1 Mb BIOS decode area accounts for the condition when A20M# is asserted and an ALT-CTRL-DEL reset is generated. The CPU's reset vector will access 4 Gb-1 Mb. When this gets forwarded to ISA, AD[32:24] are truncated and the access is aliased to 16 Mb-1 Mb=15 Mb space. If ISA memory is present at 15 Mb, there will be contention. Forcing LA20 high aliases this region to 16 Mb. The alias here is permissible since this is the 80286 reset vector location.

In addition to the normal 128 Kb byte BIOS space, the SIO supports an additional 384 Kb byte BIOS space. The SIO can support a total of 512 Kb byte BIOS space. The additional 384 Kb byte region can only be accessed by PCI masters and resides at FFF80000h to FFFDFFFFh. When enabled via the UBCSA Register, memory accesses within this region will be positively decoded, forwarded to the ISA Bus, and encoded BIOSCS# will be generated. When forwarded to the ISA Bus, the PCI AD[23:20] signals will be propagated to the ISA LA[23:20] lines as all 1's which will result in aliasing this 512 Kb byte region at the top of the 16 Mb space. To avoid contention, ISA add-in memory must not be present in this space.

All PCI cycles positively decoded in the enabled BIOS space will be broadcast to the ISA Bus. Since the BIOS device is 8 or 16 bits wide and typically has very long access times, PCI burst reads from the BIOS space will invoke "disconnect target termination" semantics after the first data transaction in order to meet the PCI incremental latency guidelines.

The following tables and diagrams describe the operation of the SIO in response to PCI BIOS space accesses.

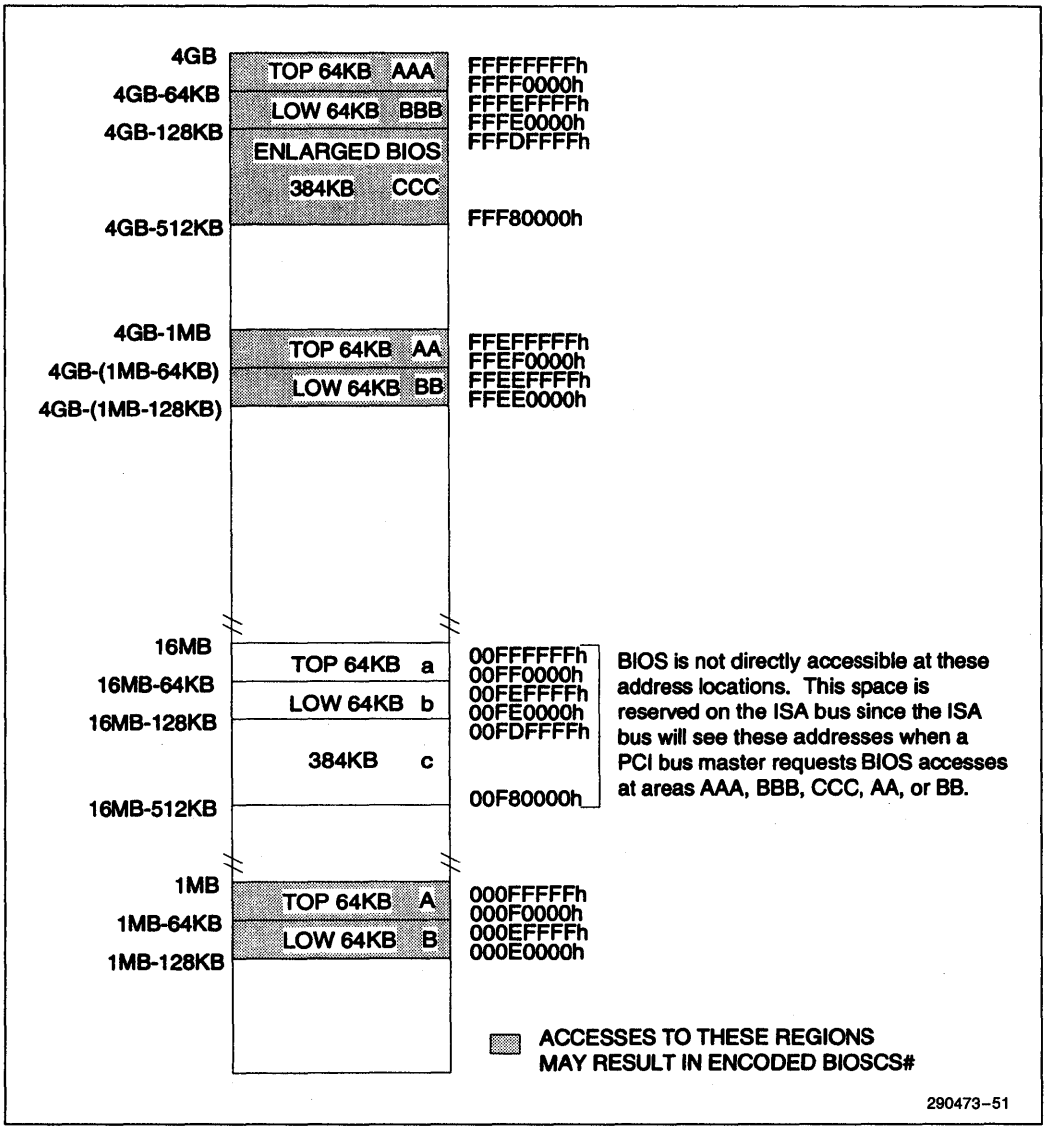


Figure 5-47. BIOS Space Decode Map

The BIOS space decode map, Figure 5-47, shows the possible BIOS spaces and the aliases throughout the memory space. The various regions are designated with code letters; "a's" for the top 64 Kb, "b's" for the low 64 Kb, and "c's" for the enlarged space.

Table 5-15 indicates the SIO's response to PCI BIOS space accesses based on its configuration state.

Table 5-15. PCI Master BIOS Space Decoding

CYCLE		SIO CONFIGURATION			SIO RESPONSE			
Master	Region	Top 64 Kb BIOS Positive Decode Enabled[1]	Low 64 Kb BIOS Enabled[2]	Enlarged BIOS Enabled[3]	Encoded BIOSCS# Generated	LA20	Positive PCI Decode	Subtractive PCI Decode
PCI	A	0	x	x	Yes	Pass (0)	No	Yes
PCI	A	1	x	x	Yes	Pass (0)	Yes[5]	No[5]
PCI	B	x	0	x	No	Pass (0)	No	Yes
PCI	B	x	1	x	Yes	Pass (0)	Yes[5]	No[5]
PCI	a	1	x	x	No	Pass (1)	No	Yes
PCI	b	x	0	x	No	Pass (1)	No	Yes
PCI	c	x	x	0	No	Pass (1)	No	Yes
PCI	AA	0	x	x	Yes	1	No	Yes 4
PCI	AA	1	x	x	Yes	1	Yes[4, 5]	No[5]
PCI	BB	x	0	x	No	x	No	No
PCI	BB	x	1	x	Yes	1	Yes[4, 5]	No[5]
PCI	AAA	0	x	x	Yes	Pass (1)	No	Yes[4]
PCI	AAA	1	x	x	Yes	Pass (1)	Yes[4, 5]	No[5]
PCI	BBB	x	0	x	No	x	No	No
PCI	BBB	x	1	x	Yes	Pass (1)	Yes[4, 5]	No[5]
PCI	CCC	x	x	0	No	x	No	No
PCI	CCC	x	x	1	Yes	Pass (1)	Yes[4]	No

NOTES:
1) The column labeled "Top 64 Kb BIOS Positive Decode Enable" shows the value of the ISA Clock Register bit 6. This bit determines the decoding for memory regions A, AA, and AAA (1=positive, 0= negative decoding). Note that if bit 4 in the MEMCS# Control Register is set to a 1 (Global MEMCS# decode enabled), the positive decoding function enabled by having ISA Clock Register bit 6=1 is ignored. Subtractive decoding is provided, instead.
2) The column labeled "Low 64 Kb BIOS Enable" shows the value of the Utility Bus Chip Select Enable A Bit 6. This bit determines whether memory regions B, BB, and BB are enabled (bit =1) or disabled (bit=0).
3) The column labeled "Enlarged BIOS Enabled" shows the value of the Utility Bus Chip Select Enable A Bit 7. This bit determines whether memory region CCC is enabled (bit=1) or disabled (bit=0).
4) ISA memory is not allowed to be enabled at the corresponding aliased areas or contention will result.
5) When bit 4 in the MEMCS# Control Register is set to a 1 (Global MEMCS# decode enabled), positive decoding for these areas will be disabled. The SIO will only provide subtractive decoding in this case.

5.5.1.3 MEMCS# Decoding

For MEMCS# decoding, the SIO decodes sixteen ranges. Fourteen of these ranges can be enabled or disabled independently for both read and write cycles. The fifteenth range (0-512k) and sixteenth range (programmable from 1 Mb up to 512 Mb in 2 Mb increments) can be enabled or disabled only. Addresses within these enabled regions generate a MEMCS# signal that can be used by the host bridge to know when to forward PCI cycles to main memory. A seventeenth range is available that can be used to identify a "memory hole". Addresses within this hole will not generate a MEMCS#. The address regions are summarized.

- 0-to-512 Kb Memory (can only be disabled if MEMCS# is completely disabled)
- 512 Kb to 640 Kb Memory
- (1 Mb-64 Kb) -to- 1 Mb Memory (BIOS Area)
- 768 Kb -to- 918 Kb in 16 Kb sections (total of 8 sections)
- 918 Kb -to- 983 Kb in 16 Kb sections (total of 4 sections
- 1M-to-programmable boundary on 2 Mb increments from 2 Mb up to 512 Mb
- programmable "memory hole" in 64 Kb increments between 1 Mb and 16 Mb

Table 5-16 and Figure 5-48 show the registers and decode areas for MEMCS#.

Table 5-16. MEMCS# Decoding Register Summary

MAR Registers	Attribute	Memory Segments	Comments
MCSCON[4] = 0	Disable	Disable MEMCS# function	Enable/Disable MEMCS# function
MCSCON[4] = 1	Enable	Enable MEMCS# function	When enabled, 0 to 512 KB range is also automatically enabled (RE/WE)
MCSTOH/MCSBOH	MEMCS# Hole	100000h - 0FFFFFFh	1 MB to 16 MB Hole in MEMCS# region
MCSTOM	MEMCS# Top	200000h - 1FFFFFFFh	2 MB to 512 MB Top of MEMCS# region
MCSCON[1:0]	[0] = RE [1] = WE	080000h - 09FFFFh	512 KB to 640 KB R/W Enable
MCSCON[3:2]	[2] = RE [3] = WE	0F0000h - 0FFFFFh	BIOS Area R/W Enable
MAR1[1:0]	[0] = RE [1] = WE	0C0000h - 0C3FFFh	ISA Add-on BIOS R/W Enable
MAR1[3:2]	[2] = RE [3] = WE	0C4000h - 0C7FFFh	ISA Add-on BIOS R/W Enable
MAR1[5:4]	[4] = RE [5] = WE	0C8000h - 0CBFFFh	ISA Add-on BIOS R/W Enable
MAR1[7:6]	[6] = RE [7] = WE	0CC000h - 0CFFFFh	ISA Add-on BIOS R/W Enable
MAR2[1:0]	[0] = RE [1] = WE	0D0000h - 0D3FFFh	ISA Add-on BIOS R/W Enable
MAR2[3:2]	[2] = RE [3] = WE	0D4000h - 0D7FFFh	ISA Add-on BIOS R/W Enable
MAR2[5:4]	[4] = RE [5] = WE	0D8000h - 0DBFFFh	ISA Add-on BIOS R/W Enable
MAR2[7:6]	[6] = RE [7] = WE	0DC000h - 0DFFFFh	ISA Add-on BIOS R/W Enable
MAR3[1:0]	[0] = RE [1] = WE	0E0000h - 0E3FFFh	BIOS Extension R/W Enable
MAR3[3:2]	[2] = RE [3] = WE	0E4000h - 0E7FFFh	BIOS Extension R/W Enable
MAR3[5:4]	[4] = RE [5] = WE	0E8000h - 0EBFFFh	BIOS Extension R/W Enable
MAR3[7:6]	[6] = RE [7] = WE	0EC000h - 0EFFFFh	BIOS Extension R/W Enable

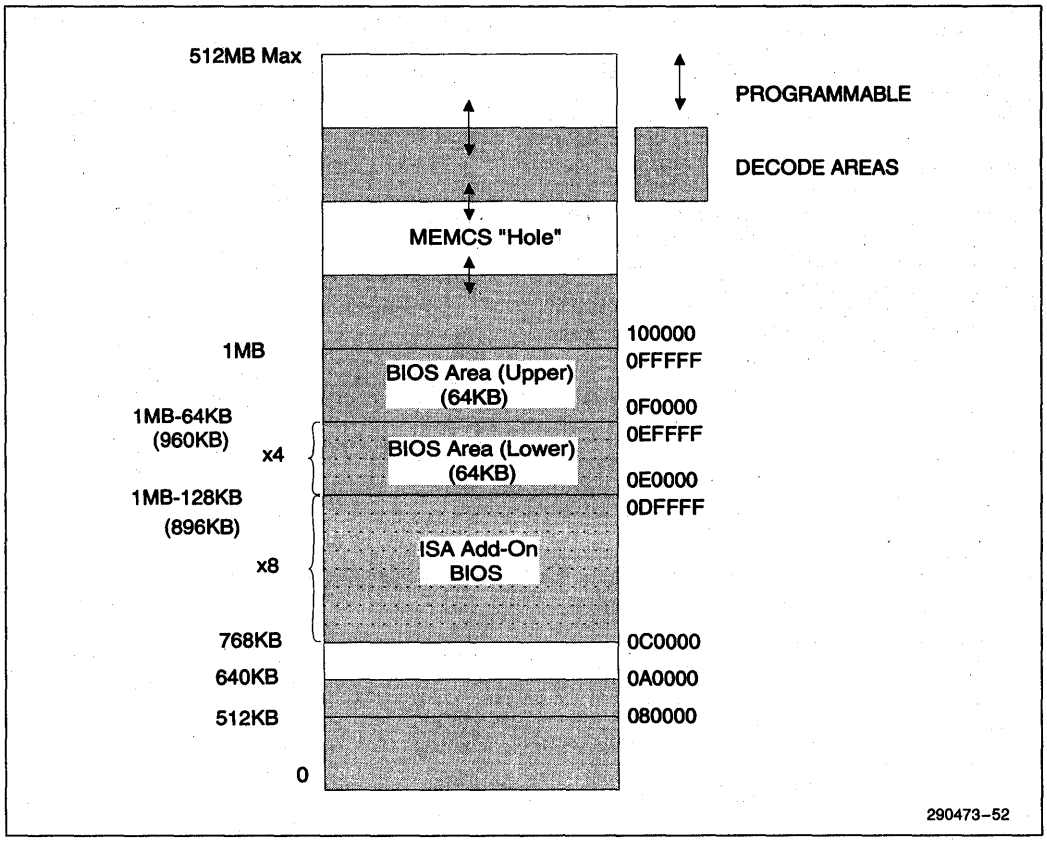


Figure 5-48. MEMCS# Decode Areas

The SIO generates MEMCS# from the PCI address. MEMCS# is generated from the clock edge after FRAME# is sampled active. MEMCS# will only go active for one PCI clock period. The SIO does not take any other action as a result of this decode other than generating MEMCS#. It is the responsibility of the device using the MEMCS# signal to generate DEVSEL#, TRDY# and any other cycle response. The device using 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.

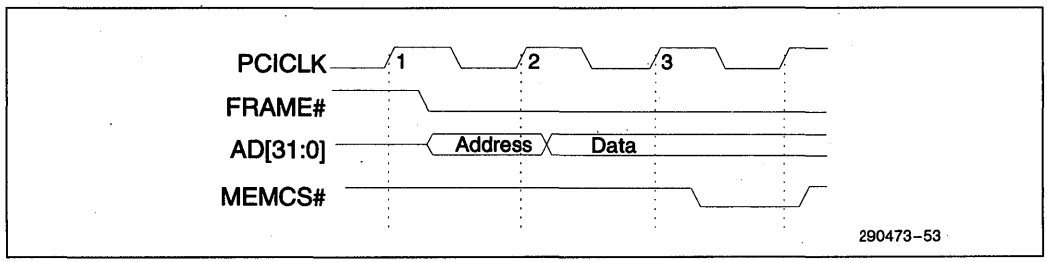


Figure 5-49. MEMCS# Generation

5.5.1.4 Subtractively Decoded Cycles to ISA

The addresses that reside on the ISA Bus could be highly fragmented. For this reason, subtractive decoding is used to forward PCI cycles to the ISA Bus. An inactive DEVSEL# will cause the SIO to forward the PCI cycle to the ISA Bus. The DEVSEL# sample point can be configured for three different settings. If the "fast" point is selected, the cycle will be forwarded to ISA when DEVSEL# is inactive at the F sample point as shown in Figure 5-50. If the "typical" point is selected, DEVSEL# will be sampled on both F and T, and if inactive, will be forwarded to the ISA Bus. Likewise, if the "slow" point is selected, DEVSEL# will be sampled at F, T, and S. The sample point should be configured to match the slowest PCI device in the system. This capability reduces the latency to ISA slaves when all PCI devices are "fast" and also allows for devices with slow decoding. Note that when these unclaimed cycles are forwarded to the ISA Bus, the SIO will drive the DEVSEL# active.

Since an active MEMCS# will always result in an active DEVSEL# at the "Slow" sample point, MEMCS# is used as an early indication of DEVSEL#. In this case, if the device using MEMCS# is the only "slow" agent in the system, the sample point can be moved in to the "typical" edge.

Unclaimed PCI cycles with memory addresses above 16M and I/O addresses above 64K will not be forwarded to the ISA Bus. The SIO will not respond with DEVSEL# (BIOS accesses are an exception to this). This is required to avoid the possibility of aliasing. Under this condition, these unclaimed cycles will be recognized as such by the originating master and the master will use "master-abort" semantics to terminate the PCI cycle.

5.5.2 DMA/ISA MASTER CYCLE ADDRESS DECODER

The SIO also contains a decoder which is used to determine the destination of ISA master and DMA master cycles. This decoder provides:

Positive Decode to PCI: Positively decodes addresses to be forwarded to the PCI Bus. This includes addresses residing directly on PCI as well as addresses that reside on the back side of PCI bridges (Host Bridges).

Access to SIO Internal Registers: Positively decodes addresses to registers within the SIO.

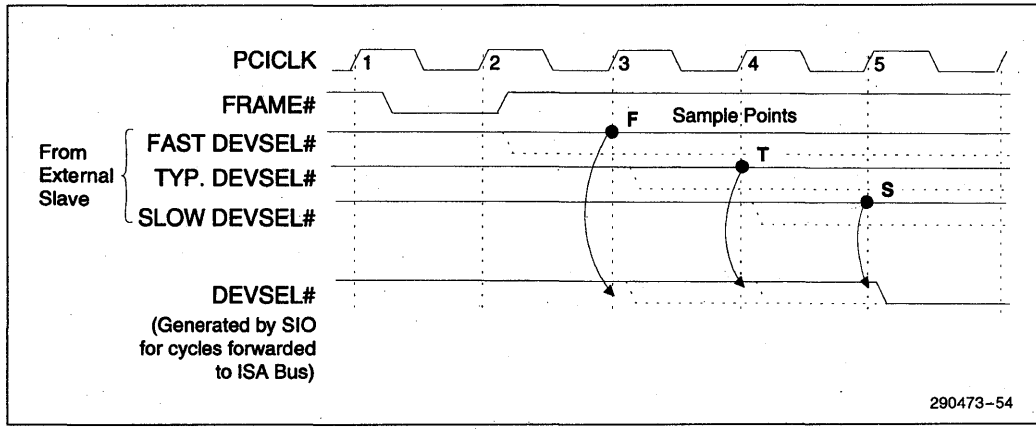


Figure 5-50. DEVSEL# Generation

BIOS Accesses: Positively decodes BIOS memory accesses and generates encoded BIOSCS#.

Utility Bus Chip Selects: Positively decodes utility bus chip selects.

Subtractive Decode: Subtractively decodes cycles to be contained to the ISA Bus.

5.5.2.1 Positive Decode to PCI

ISA master or DMA addresses that are positively decoded by this decoder will be propagated to the PCI Bus. This is the only way to forward a cycle from an ISA master or the DMA to the PCI Bus. If the cycle is not decoded by this decoder it will not be forwarded to the PCI Bus.

This decoder has several memory address regions to positively decode cycles that should be forwarded to the PCI Bus. These regions are listed below. Regions "a" through "e" are fixed and can be enabled or disabled independently. Region "f" defines a space starting at 1M with a programmable upper boundary up to 16 Mb. Within this region a hole can be opened. Its size and location are programmable to allow a hole to be opened in the memory space. A memory address above 16 Mb will be forwarded to the PCI Bus automatically. This is possible only during DMA cycles in which the DMA has been programmed for 32 bit addressing above 16 Mb.

a. Memory: 0-512 Kb
b. Memory: 512 Kb-640 Kb
c. Memory: 640 Kb- 768 Kb (Video buffer)
d. Memory: 768 Kb - 896 Kb in eight 16K sections (Expansion ROM)
e. Memory: 896 Kb - 960 Kb (lower BIOS area)
f. Memory: 1 Mb-to-X Mb (up to 16 Mb) 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 64 KByte boundaries up to 16 Mb. The hole can be between 64 Kb and 8 Mb in size in 64 Kb increments located on any 64 Kb boundary. (Refer to the ISA Address Decoder Register in the register description section, section 5.5.2)
g. Memory: >16 Mb automatically forwarded to PCI

Figure 5-47 shows a map of the ISA master/DMA decode regions and Table 5-17 summarizes the registers used to configure the decoder.

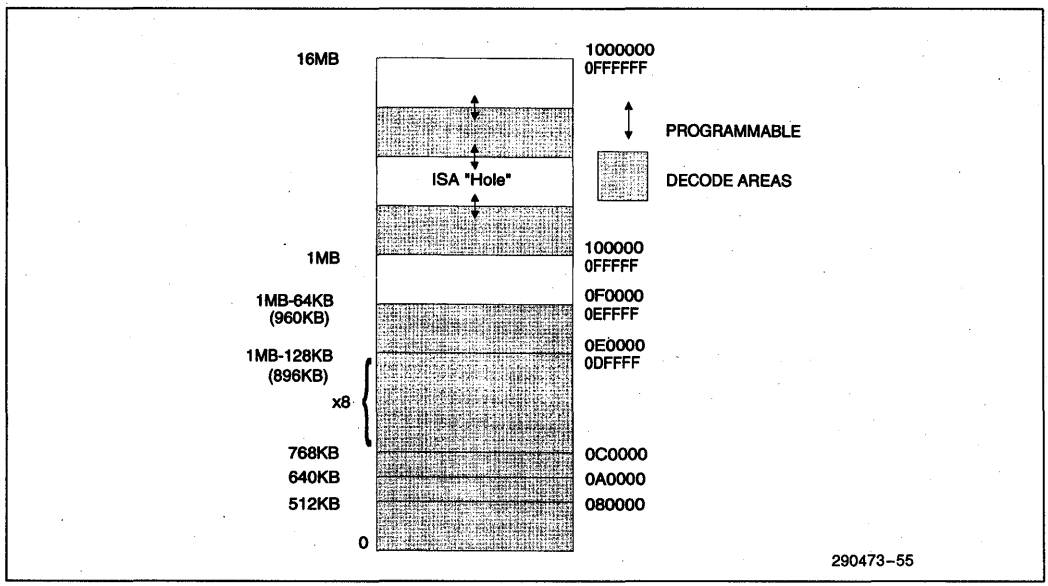


Figure 5-51. ISA Master/DMA to PCI Bus Decoder Regions

Table 5-17. ISA Master/DMA to PCI Bus Decoding Register Summary

MAR Registers	Attribute	Memory Segments	Comments
IADCON[7:4]	ISA Memory Top	100000h - 0FFFFFFh	1 MB to 16 MB Top of ISA region
IADTOH/IADBOH	ISA Hole	100000h - 0FFFFFFh	1 MB to 16 MB Hole in ISA region
IADCON[0]	Enable/Disable	000000h - 07FFFFh	0 to 512 KB Enable / Disable
IADCON[1]	Enable/Disable	080000h - 09FFFFh	512 KB to 640 KB Enable/Disable
IADCON[2]	Enable/Disable	0A0000h - 0BFFFFh	640 KB to 768 KB Enable/Disable
IADCON[3] *	Enable/Disable	0E0000h - 0EFFFFh	896 KB to 960 KB Lower BIOS Enable/Disable
IADRBE[0]	Enable/Disable	0C0000h - 0C3FFFh	ISA Add-on BIOS (Expansion ROM) Enable
IADRBE[1]	Enable/Disable	0C4000h - 0C7FFFh	ISA Add-on BIOS (Expansion ROM) Enable
IADRBE[2]	Enable/Disable	0C8000h - 0CBFFFh	ISA Add-on BIOS (Expansion ROM) Enable
IADRBE[3]	Enable/Disable	0CC000h - 0CFFFFh	ISA Add-on BIOS (Expansion ROM) Enable
IADRBE[4]	Enable/Disable	0D0000h - 0D3FFFh	ISA Add-on BIOS (Expansion ROM) Enable
IADRBE[5]	Enable/Disable	0D4000h - 0D7FFFh	ISA Add-on BIOS (Expansion ROM) Enable
IADRBE[6]	Enable/Disable	0D8000h - 0DBFFFh	ISA Add-on BIOS (Expansion ROM) Enable
IADRBE[7]	Enable/Disable	0DC000h - 0DFFFFh	ISA Add-on BIOS (Expansion ROM) Enable

* This can be overridden by bit 6 of the UBCSA Register being set to a 1.

5.5.2.2 SIO Internal Registers

Most of the internal SIO registers are accessible by ISA masters. Table 5-14 lists the registers that are not accessible by ISA masters. Registers accessed by ISA masters are run as 8-bit extended I/O cycles.

5.5.2.3 BIOS Accesses

The 128K BIOS memory space is located at 000E0000h to 000FFFFFh, and is aliased at FFFE0000h to FFFFFFFFh (top of 4 Gb) and FFEE0000h to FFEFFFFFh (top of 4 Gb-1 Mb). The aliased regions account for the CPU reset vector and the uncertainty of the state of A20GATE when a software reset occurs. This 128K block is split into two 64K blocks. The top 64K is always enabled while the bottom 64K can be enabled or disabled (the aliases automatically match). ISA masters can only access BIOS in the 000E0000 to 000FFFFFh region.

ISA originated accesses to the enabled 64K sections of the BIOS space (000E0000h-000FFFFFh) will activate the encoded BIOSCS# signal. ISA originated cycles will not be forwarded to the PCI Bus. Encoded BIOSCS# is combinatorially generated from the ISA SA and LA address bus. Encoded BIOSCS# is disabled during refresh and DMA cycles. The ISA Master/DMA BIOS Decoding Table indicates the SIO's response to BIOS accesses based on the configuration state.

Table 5-18. ISA Master/DMA BIOS Decoding

CYCLE		SIO CONFIGURATION			SIO RESPONSE		
Master	Region[1]	Top 64 Kb PCI Positive Decode Enabled[2]	Low 64 Kb BIOS Enabled[3]	Forward Low 64 Kb to PCI Enabled[4]	Encoded BIOSCS# Generated	Forward to PCI	Contain to ISA
ISA/DMA	A	x	x	x	Yes	No	Yes
ISA/DMA	B	x	0[5]	0	No	No	Yes
ISA/DMA	B	x	0[5]	1	No	Yes	No
ISA/DMA	B	x	1	x	Yes	No	Yes
ISA/DMA ISA/DMA ISA/DMA	a b c	These cycles will be forwarded to PCI dependent on the state of the ISA Address Decoder Configuration Registers. Encoded BIOSCS# will not be generated for any of these cycles					

NOTES:
1) The memory sections referenced can be found in Figure 5-47.
2) The column labeled "Top 64 Kb BIOS Positive Decode Enabled" shows the value of the ISA Clock Divisor Configuration Register bit 6. This bit determines how the memory regions is decode (0=subtractively decoded, 1=positively decoded).
3) The column labeled "Low 64 Kb BIOS Enable" shows the value of the Utility Bus Chip Select Enable A Configuration Register bit 6. This bit determines if the memory regions is enabled (bit=1) or disabled (bit=0).
4) The column labeled "Forward Low 64 Kb to PCI Enables" shows the value of the ISA Address Decoder Control Configuration Register Bit 3. This bit determines whether PCI Bus forwarding is enabled (bit=1) or disabled (bit=0).
5) Forward to PCI if IADCON Bit 6=1.

5.5.2.4 Utility Bus Encoded Chip Selects

The SIO generates encoded chip selects for certain functions that are located on the utility bus (formerly X-bus). The encoded chip selects are generated combinatorially from the ISA SA[15:0] address bus. The encoded chip selects are decoded externally (see Figure 5-57).

The encoded chip select table (Table 5-19) shows the addresses that result in encoded chip select generation. Chip selects can be enabled or disabled via configuration registers. In general, the addresses shown in Table 5-19 do not reside in the SIO itself. Write only addresses 70h, 372h, 3F2h are exceptions since particular bits from these registers reside in the SIO. For ISA master cycles, the SIO will respond to writes to address 70h, 372h, and 3F2h by generating IOCHRDY and writing to the appropriate bits.

Note that the SIO monitors read accesses to address 60h to support the mouse function. In this case, IOCHRDY is not generated.

Table 5-19. Encoded Chip Select Table

Address	Address				Type	Name	Encoded Chip Select
	FEDC	BA98	7654	3210			
0060h	0000	0000	0110	00x0	r/w	Keyboard Controller	KEYBRDCS#
0064h	0000	0000	0110	01x0	r/w	Keyboard Controller	KEYBRDCS#
0070h	0000	0000	0111	0xx0	w	Real Time Clock Address	RTCALE#
0071h	0000	0000	0111	0xx1	r/w	Real Time Clock Data	RTCCS#
0170h	0000	0001	0111	0000	r/w	Secondary Data Register	IDECS0#
0171h	0000	0001	0111	0001	r/w	Secondary Error Register	IDECS0#
0172h	0000	0001	0111	0010	r/w	Secondary Sector Count Register	IDECS0#
0173h	0000	0001	0111	0011	r/w	Secondary Sector Number Register	IDECS0#
0174h	0000	0001	0111	0100	r/w	Secondary Cylinder Low Register	IDECS0#
0175h	0000	0001	0111	0101	r/w	Secondary Cylinder High Register	IDECS0#
0176h	0000	0001	0111	0110	r/w	Secondary Drive/head Register	IDECS0#
0177h	0000	0001	0111	0111	r/w	Secondary Status Register	IDECS0#
01F0h	0000	0001	1111	0000	r/w	Primary Data Register	IDECS0#
01F1h	0000	0001	1111	0001	r/w	Primary Error Register	IDECS0#
01F2h	0000	0001	1111	0010	r/w	Primary Sector Count Register	IDECS0#
01F3h	0000	0001	1111	0011	r/w	Primary Sector Number Register	IDECS0#
01F4h	0000	0001	1111	0100	r/w	Primary Cylinder Low Register	IDECS0#
01F5h	0000	0001	1111	0101	r/w	Primary Cylinder High Register	IDECS0#
01F6h	0000	0001	1111	0110	r/w	Primary Drive/head Register	IDECS0#
01F7h	0000	0001	1111	0111	r/w	Primary Status Register	IDECS0#
0278h	0000	0010	0111	1x00	r/w	LPT3 PP Data Latch	LPTCS#
0279h	0000	0010	0111	1x01	r	LPT3 PP Status	LPTCS#
027Ah	0000	0010	0111	1x10	r/w	LPT3 PP Control	LPTCS#
027Bh	0000	0010	0111	1x11	r/w		LPTCS#
02F8h	0000	0010	1111	1000	r/w	COM2 SP Transmit/ Receive register	COM2CS#
02F9h	0000	0010	1111	1001	r/w	COM2 SP Interrupt Enable register	COM2CS#
02FAh	0000	0010	1111	1010	r	COM2 SP Interrupt Enable register	COM2CS#
02FBh	0000	0010	1111	1011	r/w	COM2 SP Line Control Register	COM2CS#
02FCh	0000	0010	1111	1100	r/w	COM2 SP Modem Control Register	COM2CS#
02FDh	0000	0010	1111	1101	r	COM2 SP Line Status Register	COM2CS#
02FEh	0000	0010	1111	1110	r	COM2 SP Modem Status Register	COM2CS#
02FFh	0000	0010	1111	1111	r/w	COM2 SP Scratch Register	COM2CS#
0370h	0000	0011	0111	0000	r/w	Secondary Floppy Disk Extended Mode Register	FLOPPYCS#
0371h	0000	0011	0111	0001	r/w	Secondary Floppy Disk Extended Mode Register	FLOPPYCS#
0372	0000	0011	0111	0010	w	Secondary Floppy Disk Digital Output Register	FLOPPYCS#

Table 5-19. Encoded Chip Select Table (Continued)

Address	Address				Type	Name	Encoded Chip Select
	FEDC	BA98	7654	3210			
0373h	0000	0011	0111	0011	r/w	Reserved	FLOPPY CS#
0374h	0000	0011	0111	0100	r/w	Secondary Floppy Disk Status Register	FLOPPYCS#
0375h	0000	0011	0111	0101	r/w	Secondary Floppy Disk Data Register	FLOPPYCS#
0376h	0000	0011	0111	0110	r/w	Secondary Alternate Status Register	IDECS1#
0377h	0000	0011	0111	0111	r	Secondary Drive Address Register	IDECS1#
0377h *	0000	0011	0111	011x	r/w	Secondary Floppy Disk Digital Input Register	FLOPPYCS#
0378h	0000	0011	0111	1x00	r/w	LPT2 PP Data Latch	LPTCS#
0379h	0000	0011	0111	1x01	r	LPT2 PP Status	LPTCS#
037Ah	0000	0011	0111	1x10	r/w	LPT2 PP Control	LPTCS#
037Bh	0000	0011	0111	1x11	r/w		LPTCS#
03BCh	0000	0011	1011	1100	r/w	LPT1 PP Data Latch	LPTCS#
03BDh	0000	0011	1011	1101	r	LPT1 PP Status	LPTCS#
03BEh	0000	0011	1011	1110	r/w	LPT1 PP Control	LPTCS#
03BFh	0000	0011	1011	1111	r/w		LPTCS#
03F0h	0000	0011	1111	0000	r/w	Primary Floppy Disk Extended Mode Register	FLOPPYCS#
03F1h	0000	0011	1111	0001	r/w	Primary Floppy Disk Extended Mode register	FLOPPYCS#
03F2h	0000	0011	1111	0010	w	Primary Floppy Disk Digital Output Register	FLOPPYCS#
03F3h	0000	0011	1111	0011	r/w	Reserved	FLOPPY CS#
03F4h	0000	0011	1111	0100	r/w	Primary Floppy Disk Status Register	FLOPPYCS#
03F5h	0000	0011	1111	0101	r/w	Primary Floppy Disk Data Register	FLOPPYCS#
03F6h	0000	0011	1111	0110	r/w	Primary Drive Alternate Status Register	IDECS1#
03F7h	0000	0011	1111	0111	r	Primary Drive Address Register	IDECS1#
03F7h *	0000	0011	1111	011x	r/w	Primary Floppy Disk Digital Input Register	FLOPPYCS#
03F8h	0000	0011	1111	1000	r/w	COM1 SP Transmit/Receive Register	COM1CS#
03F9h	0000	0011	1111	1001	r/w	COM1 SP Interrupt Enable Register	COM1CS#
03FAh	0000	0011	1111	1010	r	COM1 SP Interrupt Identification Register	COM1CS#
03FBh	0000	0011	1111	1011	r/w	COM1 SP Line Control Register	COM1CS#
03FCh	0000	0011	1111	1100	r/w	COM1 SP Modem Control Register	COM1CS#
03FDh	0000	0011	1111	1101	r	Register	COM1CS#
03FEh	0000	0011	1111	1110	r	COM1 SP Modem Status Register	COM1CS#
03FFh	0000	0011	1111	1111	r/w	COM1 SP Scratch Register	COM1CS#
0800h-08FFh	0000	1000	xxxx	xxxx	r/w		CFIGMEMCS#
0C00h	0000	1100	0000	0000	r/w		CPAGECS#

* If both the IDE and Floppy Drive are located on the UBUS, FLOPPYCS# will not be generated, IDECS1# will be generated.

5.5.2.5 Subtractive Decode to ISA

ISA master and DMA cycles not positively decoded by the ISA decoder are contained to the ISA Bus.

5.6 Data Buffering

The SIO contains data buffers to isolate the PCI Bus from the ISA Bus. The buffering is described from two perspectives: PCI master accesses to the ISA Bus (Posted Write Buffer) and DMA/ISA master accesses to the PCI Bus (Line Buffer). Temporarily buffering the data requires buffer management logic to ensure that the data buffers remain coherent.

5.6.1 DMA/ISA MASTER LINE BUFFER

An 8-byte Line Buffer is used to isolate the ISA Bus's slower I/O devices from the PCI Bus. The Line Buffer is bi-directional and is used by ISA masters and the DMA controller to assemble and disassemble data. Only memory data written to or read from the PCI Bus by an ISA master or DMA is assembled/disassembled using this 8 byte line buffer. I/O cycles do not use the buffer.

Bits 0 and 1 of the PCI Control Register set the buffer to operate in either single transaction mode (bit = 0) or 8 byte mode (bit = 1). Note that ISA masters and DMA controllers can have their buffer modes configured separately.

In single transaction mode, the buffer will store only one transaction. For DMA/ISA master writes, this single transaction buffer looks like a posted write buffer. As soon as the ISA cycle is complete, a PCI cycle is scheduled. Subsequent DMA/ISA master writes are held off in wait states until the buffer is empty. For DMA/ISA master reads, only the data requested is read over the PCI Bus. For instance, if the DMA channel is programmed in 16-bit mode, 16 bits of data will be read from PCI. As soon as the requested data is valid on the PCI bus, it is latched into the Line Buffer and the ISA cycle is then completed, as timing allows. Single transaction mode will guarantee strong read and write ordering through the buffers.

In 8-byte mode, for write data assembly, the Line Buffer acts as two individual 4 byte buffers working in ping pong fashion. For read data disassembly, the Line Buffer acts as one 8 byte buffer.

5.6.2 PCI MASTER POSTED WRITE BUFFER

PCI master memory write cycles destined to ISA memory are buffered in a 32-bit Posted Write Buffer. The PCI Memory Write and Memory Write and Invalidate commands are all treated as a memory write and can be posted, subject to the Posted Write Buffer status. The Posted Write Buffer has an address associated with it. A PCI master memory write can be posted any time the posted write buffer is empty and write posting is enabled (bit 2 of the PCI Control Configuration Register is set to a 1). Also, the ISA Bus must not be occupied. If the posted write buffer contains data, the PCI master write cycle is retried. If the posted write buffer is disabled, the SIO's response to a PCI master memory write is dependent on the state of the ISA Bus. If the ISA Bus is available and the posted write buffer is disabled, the cycle will immediately be forwarded to the ISA Bus (TRDY# will not be asserted until the ISA cycle has completed). If the ISA Bus is busy and the posted write buffer is disabled, the cycle is retried.

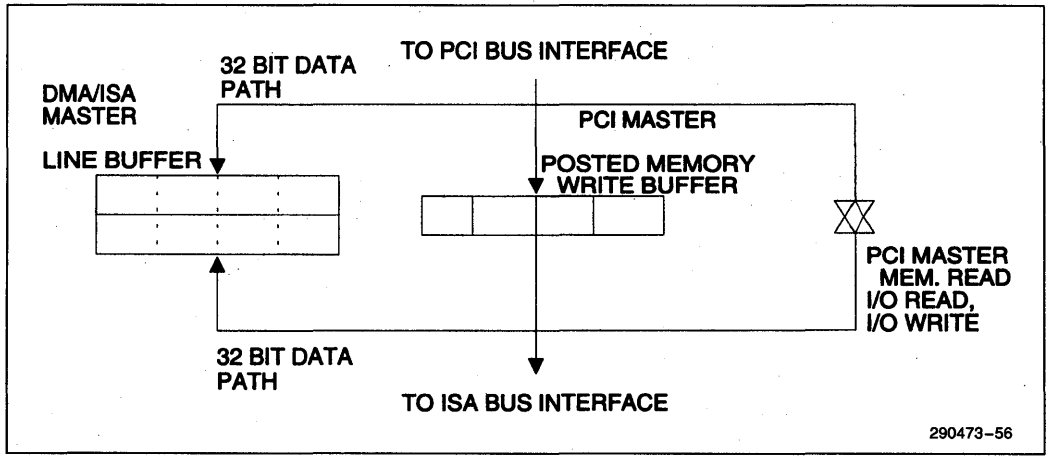


Figure 5-52. SIO Buffer Diagram

Memory read and I/O read and I/O write cycles do not use the 32-bit Posted Write Buffer.

5.6.3 BUFFER MANAGEMENT

Any time data is temporarily stored in the buffers between the ISA Bus and the PCI Bus, there are potential data coherency problems.

The SIO contains buffer management circuitry which guarantees data coherency by intercepting synchronization protocol between the busses and managing the buffers before synchronization communication between the busses is complete. The buffers are flushed or invalidated as appropriate before a bus cycle is allowed to occur in cases where data coherency could be lost.

5.6.3.1 DMA/ISA Master Line Buffer - Write State

When the DMA/ISA Master Line Buffer contains data that is to be written to the PCI Bus, it is in the Write State. The 8-byte line buffer is flushed when the line becomes full, when a subsequent write is a line miss, when a subsequent write would overwrite an already valid byte, or when a subsequent cycle is a read. The ISA master or DMA cycle that triggers the buffer flush will be held in wait states until the flush is complete. The buffer is also flushed whenever there is a change in ISA Bus ownership as indicated by any DACK# signal going inactive.

Once the buffer is scheduled to be flushed to PCI, any PCI cycle to the SIO or ISA Bus will get retried by the SIO.

5.6.3.2 DMA/ISA Master Line Buffer - Read State

When the DMA/ISA Master Line Buffer contains data that has been read from the PCI Bus, it is in the Read State. The data in the buffer will be invalidated when the SIO accepts a PCI memory or I/O write cycle. The line buffer in the read state is also invalidated when a subsequent read is a line miss, or when a subsequent cycle is a write. The line buffer in the read state is not invalidated on a change of ISA ownership. Note that as bytes are disassembled from the line buffer they are invalidated so that subsequent reads to the same byte will cause a line buffer miss.

5.6.3.3 PCI Master Posted Write Buffer

As soon as a PCI master has posted a memory write into the posted write buffer, the buffer is scheduled to be written to the ISA Bus. Any subsequent PCI master cycles to the SIO (including ISA Bus) will be retried until the posted write buffer is empty.

Prior to granting the ISA Bus to an ISA master or the DMA, the PCI master posted memory write buffer is flushed. Also, as long as the ISA master or DMA owns the ISA Bus, the posted write buffer is disabled. A PCI master write can not be posted while an ISA master or the DMA owns the ISA Bus.

5.7 SIO Timers

5.7.1 INTERVAL TIMERS

The SIO contains three counters that are equivalent to those found in the 82C54 programmable interval timer. The three counters are contained in one SIO timer unit, referred to as Timer-1. Each counter output provides a key system function. Counter 0 is connected to interrupt controller IRQ0 and provides a system timer interrupt for a time-of-day, diskette time-out, or other system timing functions. Counter 1 generates a refresh request signal and Counter 2 generates the tone for the speaker. Note that the 14.31818 Mhz counters use OSC for a clock source.

Full details of this counter can be found in the 82C54 data sheet.

Table 5-20. Interval Timer Functions Table

Interval Timer Functions	
Function	**Counter 0 - System Timer**
Gate	Always On
Clock In	1.193 Mhz(OSC/12)
Out	INT-1 IRQ0
Function	**Counter 1 - Refresh Request**
Gate	Always On
Clock In	1.193 MHz(0SC/12)
Out	Refresh Request
Function	**Counter 2 - Speaker Tone**
Gate	Programmable - Port 61h
Clock In	1.193 MHz(OSC/12)
Out	Speaker

5.7.1.1 Interval Timer Address Map

The table below shows the I/O address map of the interval timer counters.

Table 5-21. Interval Timer Counters I/O Address Map

I/O Address	Register Description
040h	System Timer (Counter 0)
041h	Refresh Request (Counter 1)
042h	Speaker Tone (Counter 2)
043h	Control Word Register

Counter 0, System Timer
This counter functions as the system timer by controlling the state of IRQ0 and is typically programmed for Mode 3 operation. The counter produces a square wave with a period equal to the product of the counter period (838 ns) and the initial count value. The counter loads the initial count value one counter period after software writes the count value to the counter I/O address. The counter initially asserts IRQ0 and decrements the count value by two each counter period. The counter negates IRQ0 when the count value reaches 0. It then reloads the initial count value and again decrements the initial count value by two each counter period. The counter then asserts IRQ0 when the count value reaches 0, reloads the initial count value, and repeats the cycle, alternately asserting and negating IRQ0.

Counter 1, Refresh Request Signal
This counter provides the refresh request signal and is typically programmed for Mode 2 operation. The counter negates refresh request for one counter period (833 ns) during each count cycle. The initial count value is loaded one counter period after being written to the counter I/O address. The counter initially asserts refresh request, and negates it for 1 counter period when the count value reaches 1. The counter then asserts refresh request and continues counting from the initial count value.

Counter 2, Speaker Tone
This counter provides the speaker tone and is typically programmed for Mode 3 operation. The counter provides a speaker frequency equal to the counter clock frequency (1.193 Mhz) divided by the initial count value. The speaker must be enabled by a write to port 061h (see Section 4.5.1 on the NMI Status and Control ports).

5.7.1.2 Programming the Interval Timer

The counter/timers are programmed by I/O accesses and are addressed as though they are contained in one 82C54 interval timer. A single Control Word Register controls the operation of all three counters.

The interval timer is an I/O-mapped device. Several commands are available:

- The Control Word Command specifies:
  - which counter to read or write
  - the operating mode
  - the count format (binary or BCD)
- The Counter Latch Command latches the current count so that it can be read by the system. The countdown process continues.
- The Read Back Command reads the count value, programmed mode, the current state of the OUT pins, and the state of the Null Count Flag of the selected counter.

The Read/Write Logic selects the Control Word Register during an I/O write when address lines A[1:0] = 11. This condition occurs during an I/O write to port address 043h, the address for the Control Word Register on Timer 1. If the CPU writes to port 043h, the data is stored in the Control Word Register and is interpreted as the Control Word used to define the operation of the Counters.

The Control Word Register is write-only. Counter status information is available with the read back Command.

The following table lists the six operating modes for the interval counters.

Table 5-22. Counter Operating Modes

Mode	Function
0	Out signal on end of count (= 0)
1	Hardware retriggerable one-shot
2	Rate generator (divide by n counter)
3	Square wave output
4	Software triggered strobe
5	Hardware triggered strobe

Because the timer counters wake up in an unknown state after power up, multiple refresh requests may be queued. To avoid possible multiple refresh cycles after power up, program the timer counter immediately after power up.

Write Operations
Programming the interval timer is a simple process:

1. Write a control word.
2. Write an initial count for each counter.
3. Load the least and/or most significant bytes (as required by Control Word bits 5, 4) of the 16-bit counter.

The programming procedure for the SIO timer is very flexible. Only two conventions need to be observed. First, for each counter, the control word must be written before the initial count is written. Second, the initial count must follow the count format specified in the control word (least significant byte only, most significant byte only, or least significant byte and then most significant byte).

Since the Control Word Register and the three counters have separate addresses (selected by the A1, A0 inputs), and each control word specifies the counter it applies to (SC0, SC1 bits), no special instruction sequence is required. Any programming sequence that follows the conventions above is acceptable.

A new initial count may be written to a counter at any time without affecting the counter's programmed mode. Counting will be affected as described in the mode definitions. The new count must follow the programmed count format.

If a counter is programmed to read/write two-byte counts, the following precaution applies: A program must not transfer control between writing the first and second byte to another routine which also writes into that same counter. Otherwise, the counter will be loaded with an incorrect count.

Interval Timer Control Word Format
The control word specifies the counter, the operating mode, the order and size of the count value, and whether it counts down in a 16-bit or binary-coded decimal (BCD) format. After writing the control word, a new count may be written at any time. The new value will take effect according to the programmed mode.

If a counter is programmed to read/write two-byte counts, the following precaution applies: A program must not transfer control between writing the first and second byte to another routine which also writes into that same counter. Otherwise, the counter will be loaded with an incorrect count. The count must always be completely loaded with both bytes.

Read Operations
It is often desirable to read the value of a counter without disturbing the count in progress. This is easily done in the SIO timer unit.

There are three possible methods for reading the counters: a simple read operation, the Counter Latch Command, and the Read-Back Command. Each is explained below.

Counter I/O Port Read
The first method is to perform a simple read operation. To read the counter, which is selected with the A1, A0 inputs (port 040h, 041h, or 042h), the CLK input of the selected counter must be inhibited by using either the GATE input or external logic. Otherwise, the count may be in the process of changing when it is read, giving an undefined result. When reading the count value directly, follow the format programmed in the control register: read LSB, read MSB, or read LSB then MSB. Within the SIO timer unit, the GATE input on Counter 0 and Counter 1 is tied high. Therefore, the direct register read should not be used on these two counters. The GATE input of Counter 2 is controlled through I/O port 061h. If the GATE is disabled through this register, direct I/O reads of port 042h will return the current count value.

Counter Latch Command
The Counter Latch Command latches the count at the time the command is received. This command is used to ensure that the count read from the counter is accurate (particularly when reading a two-byte count). The count value is then read from each counter's Count Register as was programmed by the Control Register.

The selected counter's output latch (OL) latches the count at the time the Counter Latch Command is received. This count is held in the latch until it is read by the CPU (or until the Counter is reprogrammed). The count is then unlatched automatically and the OL returns to "following" the counting element (CE). This allows reading the contents of the counters "on the fly" without affecting counting in progress. Multiple Counter Latch Commands may be used to latch more than one counter. Each latched counter's OL holds its count until it is read. Counter Latch Commands do not affect the programmed mode of the counter in any way. The Counter Latch Command can be used for each counter in the SIO timer unit.

If a Counter is latched and then, some time later, latched again before the count is read, the second Counter Latch Command is ignored. The count read will be the count at the time the first Counter Latch Command was issued.

With either method, the count must be read according to the programmed format; specifically, if the counter is programmed for two byte counts, two bytes must be read. The two bytes do not have to be read one right after the other. Read, write, or programming operations for other counters may be inserted between them.

Another feature of the SIO timer is that reads and writes of the same counter may be interleaved. For example, if the Counter is programmed for two byte counts, the following sequence is valid:

1. Read least significant byte.
2. Write new least significant byte.
3. Read most significant byte.
4. Write new most significant byte.

If a counter is programmed to read/write two-byte counts, a program must not transfer control between reading the first and second byte to another routine which also reads from that same counter. Otherwise, an incorrect count will be read.

Read Back Command
The third method uses the Read Back Command. The Read Back Command is used to determine the count value, programmed mode, and current states of the OUT pin and Null Count flag of the selected counter or counters. The Read Back Command is written to the Control Word Register, which causes the current states of the above mentioned variables to be latched. The value of the counter and its status may then be read by I/O access to the counter address.

The Read Back Command may be used to latch multiple counter output latches (OL) by setting the COUNT# bit D5 = 0 and selecting the desired counter(s). This single command is functionally equivalent to several counter latch commands, one for each counter latched. Each counter's latched count is held until it is read (or the counter is reprogrammed). Once read, a counter is automatically unlatched. The other counters remain latched until they are read. If multiple count Read-Back Commands are issued to the same counter without reading the count, all but the first are ignored (i.e. the count which will be read is the count at the time the first Read-Back Command was issued).

The Read Back Command may also be used to latch status information of selected counter(s) by setting STATUS# bit D4 = 0. Status must be latched to be read. The status of a counter is accessed by a read from that counter's I/O port address.

If multiple counter status latch operations are performed without reading the status, all but the first are ignored. The status returned from the read is the counter status at the time the first status Read Back Command was issued.

Both count and status of the selected counter(s) may be latched simultaneously by setting both the COUNT# and STATUS# bits [5:4] = 00. This is functionally the same as issuing two consecutive, separate Read Back Commands. The above discussions apply here also. Specifically, if multiple count and/or status Read Back Commands are issued to the same counter(s) without any intervening reads, all but the first are ignored.

If both count and status of a counter are latched, the first read operation from that counter will return the latched status, regardless of which was latched first. The next one or two reads (depending on whether the counter is programmed for one or two type counts) return the latched count. Subsequent reads return unlatched count.

5.7.2 BIOS TIMER

5.7.2.1 Overview

The SIO provides a system BIOS Timer that decrements at each edge of its 1.04 MHz clock (derived by dividing the 8.33MHz SYSCLK by 8). Since the state of the counter is undefined at power-up, it must be programmed before it can be used. Accesses to the BIOS Timer are enabled and disabled through the BIOS Timer Base Address Register. The timer continues to count even if accesses are disabled.

A BIOS Timer Register is provided to start the timer counter by writing an initial clock value. The BIOS Timer Register can be accessed as a single 16-bit I/O port or as a 32-bit port with the upper 16-bits being don't care (reserved). It is up to the software to access the I/O register in the most convenient way. The I/O address of the BIOS Timer Register is software relocatable. The I/O address is determined by the value programmed into the BIOS Timer Base Address Register.

The BIOS Timer clock has a value of 1.04 MHz using an 8.33 Mhz SYSCLK input (an 8 to 1 ratio will always exist between SYSCLK and the timer clock). This allows the counting of time intervals from 0 to approximately 65 milliseconds. Because of the PCI clock rate, it is possible to start the counter and read the value back in less than 1 msec. The expected value of the expired interval is 0, but depending on the state of the internal clock divisor, the BIOS Timer might indicate that 1 msec has expired. Therefore, accuracy of the counter is ±1 msec.

5.7.2.2 BIOS Timer Operations

A write operation to the BIOS Timer Register will initiate the counting sequence. The timer can be initiated by writing either the 16-bit data portion or the whole 32-bit register (upper 16 bits are "don't care"). After initialization, the BIOS timer will start

decrementing until it reaches zero. Then it will stop decrementing (and hold a zero value) until initialized again.

After the timer is initialized, the current value can be read at any time and the timer can be reprogrammed (new initial value written), even before it 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 (wrong count intervals).

5.8 Interrupt Controller

The SIO provides an ISA compatible interrupt controller which incorporates the functionality of two 82C59 interrupt controllers. The two controllers are cascaded so that 14 external and two internal interrupts are possible. The master interrupt controller provides IRQ [7:0] and the slave interrupt controller provides IRQ [15:8] (see Figure 5-53). The two internal interrupts are used for internal functions only and are not available to the user. IRQ2 is used to cascade the two controllers together and IRQ0 is used as a system timer interrupt and is tied to Interval Timer 1, Counter 0. The remaining 14 interrupt lines (IRQ1, IRQ3-IRQ15) are available for external system interrupts. Edge or level sense selection is programmable on a by-controller basis.

The Interrupt Controller consists of two separate 82C59 cores. Interrupt Controller 1 (CNTRL-1) and Interrupt Controller 2 (CNTRL-2) are initialized separately and can be programmed to operate in different modes. The default settings are: 80x86 Mode, Edge Sensitive (IRQ0-15) Detection, Normal EOI, Non-Buffered Mode, Special Fully Nested Mode disabled, and Cascade Mode. CNTRL-1 is connected as the Master Interrupt Controller and CNTRL-2 is connected as the Slave Interrupt Controller.

Note that IRQ13 is generated internally (as part of the coprocessor error support) by the SIO when bit 5 in the ISA Clock Divisor Register is set to a 1. When this bit is set to a 0, then the FERR#/IRQ13 signal is used as an external IRQ13 signal and has the same functionality as the normal IRQ13 signal. IRQ12/M is generated internally (as part of the mouse support) by the SIO when bit 4 in the ISA Clock Divisor Register is set to a 1. When set to a 0, the standard IRQ12 function is provided.

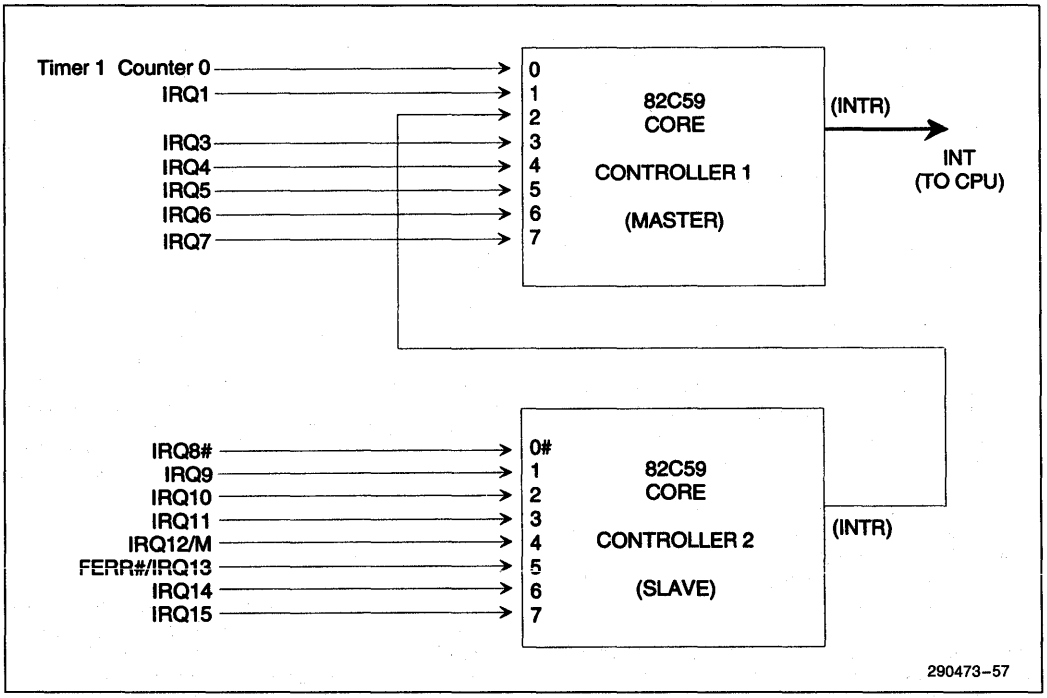


Figure 5-53. Block Diagram Of The Interrupt Controller

Table 5-23 lists the I/O port address map for the interrupt registers:

Table 5-23. Interrupt Registers I/O Port Address Map

Interrupts	I/O Address	# of bits	Register
IRQ[7:0]	0020h	8	CNTRL-1 Control Register
IRQ[7:0]	0021h	8	CNTRL-1 Mask Register
IRQ[15:8]	00A0h	8	CNTRL-2 Control Register
IRQ[15:8]	00A1h	8	CNTRL-2 Mask Register

IRQ0, IRQ2, (and possibly IRQ13 and IRQ12 if the "mouse" or floating point error logic is disabled in the ISA Clock Divisor Register), are connected to the interrupt controllers internally. The other interrupts are always generated internally and their typical functions are shown in Table 5-24.

Table 5-24. Typical Interrupt Functions

Priority	Label	Controller	Typical Interrupt Source
1	IRQ0	1	Interval timer 1, counter 0 OUT
2	IRQ1	1	Keyboard
3-10	IRQ2	1	Interrupt from controller 2
3	IRQ8#	2	Real Time Clock
4	IRQ9	2	Expansion Bus pin B04
5	IRQ10	2	Expansion Bus pin D03
6	IRQ11	2	Expansion Bus pin D04
7	IRQ12/M	2	Mouse interrupt
8	FERR#/IRQ13	2	Coprocessor Error
9	IRQ14	2	Fixed Disk Drive Controller Expansion Bus pin D07
10	IRQ15	2	Expansion Bus pin D06
11	IRQ3	1	Serial Port 2, Exp bus B25
12	IRQ4	1	Serial Port 1, Exp bus B24
13	IRQ5	1	Parallel port 2, Exp bus B23
14	IRQ6	1	Diskette controller, Exp bus B22
15	IRQ7	1	Parallel port 1, Exp bus B21

5.8.1 INTERRUPT CONTROLLER INTERNAL REGISTERS

Several registers are contained internally within each 82C59. The interrupts at the IRQ input lines are handled by two registers in cascade, the Interrupt Request Register (IRR) and the In-Service Register (ISR). The IRR is used to store all the interrupt levels which are requesting service and the ISR is used to store all the interrupt levels which are being serviced.

Internal circuitry determines the priorities of the bits set in the IRR. The highest priority is selected and strobed into the corresponding bit of the ISR during Interrupt Acknowledge Cycles.

The Interrupt Mask Register (IMR) stores the bits which mask the incoming interrupt lines. The IMR operates on the IRR. Masking of a higher priority input will not affect the interrupt request lines of lower priority inputs.

5.8.2 INTERRUPT SEQUENCE

The powerful features of the Interrupt Controller in a microcomputer system are its programmability and the interrupt routine addressing capability. The latter allows direct or indirect jumping to the specific interrupt routine requested without any polling of the interrupting devices. The following list shows the interrupt sequence for an x86 type system (the 8080 mode of the interrupt controller must never be selected when programming the SIO).

Note that externally, the interrupt acknowledge cycle sequence appears different than in a traditional discrete 82C59 implementation. However, the traditional interrupt acknowledge sequence is generated within the SIO and is an ISA compatible implementation. Also, if bit 5 in the PCI Control Register is set to a 0, the SIO will ignore the INTA cycles generated on the PCI Bus.

1. One or more of the Interrupt Request lines (IRQx) are raised high, setting the corresponding IRR bit(s).
2. The Interrupt Controller evaluates these requests, and sends an INT to the CPU, if appropriate.
3. The CPU acknowledges the INT and responds with an interrupt acknowledge cycle. This cycle is translated into a PCI Bus command. This PCI command is broadcast over the PCI Bus as a single cycle as opposed to the two cycle method typically used.
4. Upon receiving an interrupt acknowledge cycle from the CPU over the PCI, the SIO converts the single cycle into the two cycles that the internal 8259 pair can respond to with the expected interrupt vector. The cycle conversion is performed by a functional block in the SIO Interrupt Controller Unit. The internally generated interrupt acknowledge cycle is completed as soon as possible as the PCI Bus is held in wait states until the interrupt vector data is returned. Each cycle appears as an interrupt acknowledge pulse on the INTA# pin of the cascaded interrupt controllers. These two pulses are not observable at the SIO periphery.
5. Upon receiving the first internally generated interrupt acknowledge, the highest priority ISR bit is set and the corresponding IRR bit is reset. The Interrupt Controller does not drive the data bus during this cycle. On the trailing edge of the first cycle pulse, a slave identification code is broadcast by the master to the slave on a private, internal three bit wide bus. The slave controller uses these bits to determine if it must respond with an interrupt vector during the second INTA# cycle.
6. Upon receiving the second internally generated interrupt acknowledge, the Interrupt Controller releases an 8-bit pointer (the interrupt vector) onto the Data Bus where it is read by the CPU.
7. This completes the interrupt cycle. In the AEOI mode the ISR bit is reset at the end of the second interrupt acknowledge cycle pulse. Otherwise, the ISR bit remains set until an appropriate EOI command is issued at the end of the interrupt subroutine.

If no interrupt request is present at step four of the sequence (i.e., the request was too short in duration), the Interrupt Controller will issue an interrupt level 7.

5.8.3 80x86 MODE

When initializing the control registers of the 82C59, an option exists in the Initialization Control Word Four (ICW4) to select either an 80x86 or an MSC-85 microprocessor based system. The interrupt acknowledge cycle is different in an MSC-85 based system than in the 80x86 based system. The interrupt acknowledge takes three INTA# pulses with the MSC-85, rather than the two pulses with the 80x86. The SIO is used only in an 80x86 based system. Each interrupt controller's ICW4 bit 0 must be programmed to a 1 to indicate that the interrupt controller is operating in an 80x86 based system. This setting ensures proper operation during an interrupt acknowledge.

5.8.4 SIO INTERRUPT ACKNOWLEDGE CYCLE

As discussed, the CPU generates an interrupt acknowledge cycle that is translated into a single PCI command and broadcast across the PCI Bus to the SIO. Note: If bit 5 in the PCI Control Register is set to a 0, the SIO will ignore the interrupt acknowledge cycle generated on the PCI Bus. The SIO Interrupt controller translates this command into the two INTA# pulses expected by the interrupt controller subsystem. The Interrupt Controller uses the first interrupt acknowledge cycle to internally freeze the state of the interrupts for priority resolution. The first controller (CNTRL-1), as a master, issues a three bit interrupt code on the cascade lines to CNTRL-2 (internal to the SIO) at the end of the INTA# pulse. On this first cycle the interrupt controller block does not issue any data to the processor and leaves its data bus buffers disabled. CNTRL-2 decodes the information on the cascade lines, compares the code to the byte stored in Initialization Command Word Three (ICW3), and determines if it will have to broadcast the interrupt vector during the second interrupt acknowledge cycle. On the second interrupt acknowledge cycle, the master (CNTRL-1) or slave (CNTRL-2), will send a byte of data to the processor with the acknowledged interrupt code composed as follows:

Table 5-25. Content of Interrupt Vector Byte for 80x86 System Mode

	D7	D6	D5	D4	D3	D2	D1	D0
IRQ7,15	T7	T6	T5	T4	T3	1	1	1
IRQ6,14	T7	T6	T5	T4	T3	1	1	0
IRQ5,13	T7	T6	T5	T4	T3	1	0	1
IRQ4,12	T7	T6	T5	T4	T3	1	0	0
IRQ3,11	T7	T6	T5	T4	T3	0	1	1
IRQ2,10	T7	T6	T5	T4	T3	0	1	0
IRQ1,9	T7	T6	T5	T4	T3	0	0	1
IRQ0,8	T7	T6	T5	T4	T3	0	0	0

T7 - T3 represent the interrupt vector address (refer to the ICW2 Register description).

The byte of data released by the interrupt controller onto the data bus is referred to as the "interrupt vector". The format for this data is illustrated on a per-interrupt basis in the table above.

5.8.5 PROGRAMMING THE INTERRUPT CONTROLLER

The Interrupt Controller accepts two types of command words generated by the CPU or bus master:

1. Initialization Command Words (ICWs): Before normal operation can begin, each Interrupt Controller in the system must be initialized. In the 82C59, this is a two to four byte sequence. However, for the SIO, each controller must be initialized with a four byte sequence. This four byte sequence is required to configure the interrupt controller correctly for the SIO implementation. This implementation is ISA-compatible.

The four initialization command words are referred to by their acronyms: ICW1, ICW2, ICW3, and ICW4.

The base address for each interrupt controller is a fixed location in the I/O memory space, at 0020h for CNTRL-1 and at 00A0h for CNTRL-2.

An I/O write to the CNTRL-1 or CNTRL-2 base address with data bit 4 equal to 1 is interpreted as ICW1. For SIO-based ISA systems, three I/O writes to "base address + 1" (021h for CNTRL-1 and 0A1h for CNTRL-2) must follow the ICW1. The first write to "base address + 1" (021h/0A1h) performs ICW2, the second write performs ICW3, and the third write performs ICW4.

ICW1 starts the initialization sequence during which the following automatically occur:

1. Following initialization, an interrupt request (IRQ) input must make a low-to-high transition to generate an interrupt.
2. The Interrupt Mask Register is cleared.
3. IRQ7 input is assigned priority 7.
4. The slave mode address is set to 7.
5. Special Mask Mode is cleared and Status Read is set to IRR.

ICW2 is programmed to provide bits [7:3] of the interrupt vector that will be released onto the data bus by the interrupt controller during an interrupt acknowledge. A different base [7:3] is selected for each interrupt controller. Suggested values for a typical ISA system are listed in Table 5-26.

ICW3 is programmed differently for CNTRL-1 and CNTRL-2, and has a different meaning for each controller.

For CNTRL-1, the master controller, ICW3 is used to indicate which IRQx input line is used to cascade CNTRL-2, the slave controller. Within the SIO interrupt unit, IRQ2 on CNTRL-1 is used to cascade the INT output of CNTRL-2. Consequently, bit-2 of ICW3 on CNTRL-1 is set to a 1, and the other bits are set to 0's.

For CNTRL-2, ICW3 is the slave identification code used during an interrupt acknowledge cycle. CNTRL-1 broadcasts a code to CNTRL-2 over three internal cascade lines if an IRQ[x] line from CNTRL-2 won the priority arbitration on the master controller and was granted an interrupt acknowledge by the CPU. CNTRL-2 compares this identification code to the value stored in ICW3, and if the code is equal to bits [2:0] of ICW3, CNTRL-2 assumes responsibility for broadcasting the interrupt vector during the second interrupt acknowledge cycle pulse.

ICW4 must be programmed on both controllers. At the very least, bit 0 must be set to a 1 to indicate that the controllers are operating in an 80x86 system.

2. Operation Command Words (OCWs): These are the command words which dynamically reprogram the Interrupt Controller to operate in various interrupt modes.

Any interrupt lines can be masked by writing an OCW1. A 1 written in any bit of this command word will mask incoming interrupt requests on the corresponding IRQx line.

OCW2 is used to control the rotation of interrupt priorities when operating in the rotating priority mode and to control the End of Interrupt (EOI) function of the controller.

OCW3 is used to set up reads of the ISR and IRR, to enable or disable the Special Mask Mode (SMM), and to set up the interrupt controller in polled interrupt mode.

The OCWs can be written into the Interrupt Controller any time after initialization. Table 5-26 shows an example of typical values programmed by the BIOS at power-up for the SIO interrupt controller.

Table 5-26. Suggested Default Values for ICW Registers

Port	Value	Description of Contents
020h	11h	CNTLR-1, ICW1
021h	08h	CNTLR-1, ICW2 Vector Address for 000020h
021h	04h	CNTLR-1, ICW3 Indicates Slave Connection
021h	01h	CNTLR-1, ICW4 8086 Mode
021h	B8h	CNTLR-1, Interrupt Mask (may vary)
0A0h	11h	CNTLR-2, ICW1
0A1h	70h	CNTLR-2, ICW2 Vector Address for 0001C0h
0A1h	02h	CNTLR-2, ICW3 Indicates Slave ID
0A1h	01h	CNTLR-2, ICW4 8086 Mode
0A1h	BDh	CNTLR-2, Interrupt Mask (may vary)

The following flow chart illustrates the sequence software must follow to load the interrupt controller Initialization Command Words (ICWs). The sequence must be executed for CNTRL-1 and CNTRL-2. After writing ICW1, ICW2, ICW3, and ICW4 must be written in order. Any divergence from this sequence, such as an attempt to program an OCW, will result in improper initialization of the interrupt controller and unexpected, erratic system behavior. It is suggested that CNTRL-2 be initialized first, followed by CNTRL-1.

In the SIO, it is required that all four Initialization Command Words (ICWs) be initialized. Also, as shown in Figure 5-54, all ICWs must be programmed prior to programming the OCWs.

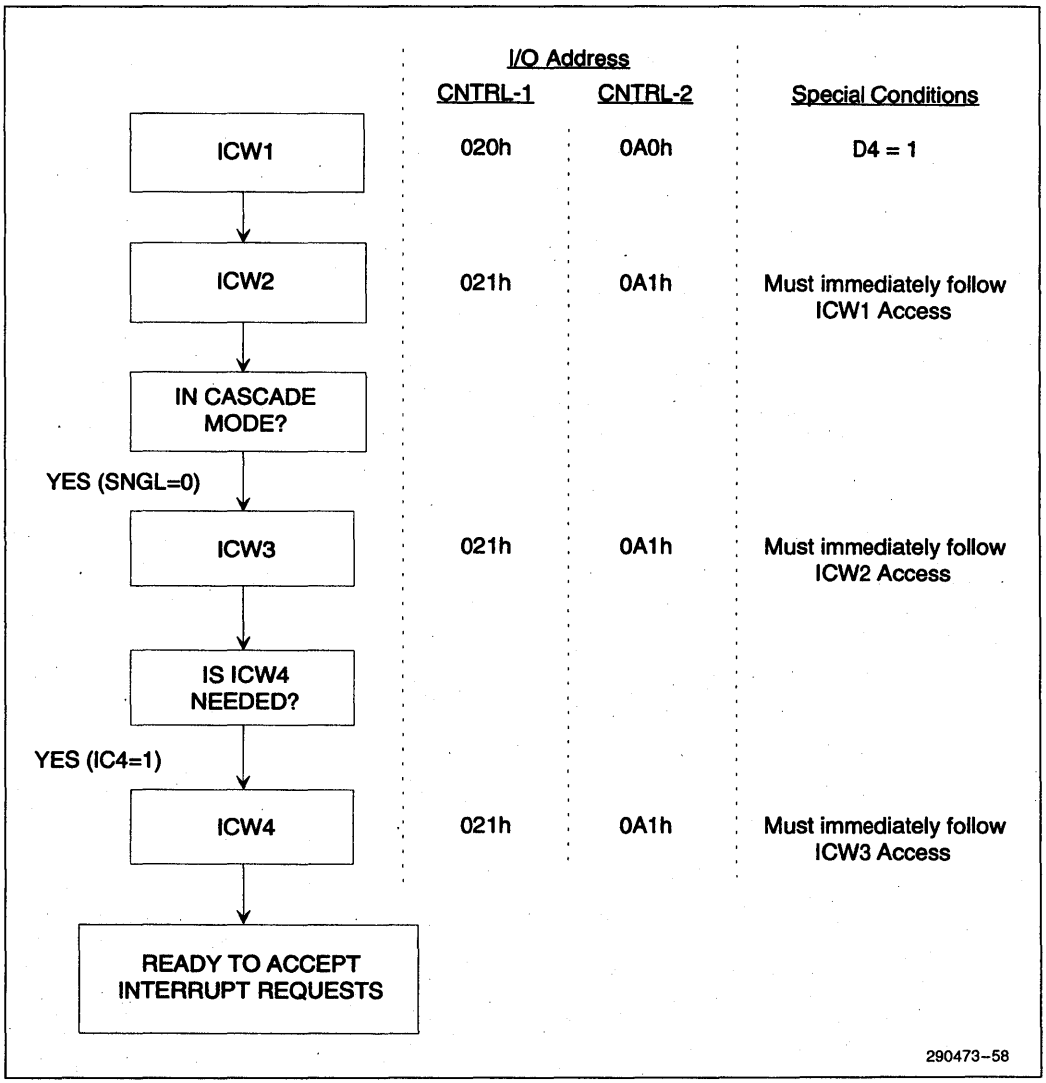


Figure 5-54. Initialization Sequence for SIO Initialization Command Words (ICWs)

5.8.6 END-OF-INTERRUPT OPERATION

5.8.6.1 End of Interrupt (EOI)

The In Service (IS) bit can be set to 0 automatically following the trailing edge of the second INTA# pulse (when AEOI bit in ICW1 is set to 1) or by a command word that must be issued to the Interrupt Controller before returning from a service routine (EOI command). An EOI command must be issued twice with this cascaded interrupt controller configuration, once for the master and once for the slave.

There are two forms of EOI commands: Specific and Non-Specific. When the Interrupt Controller is operated in modes which preserve the fully nested structure, it can determine which IS bit to set to 0 on EOI. When a Non-Specific EOI command is issued, the Interrupt Controller will automatically set to 0 the highest IS bit of those that are set to 1, since in the fully nested mode the highest IS level was necessarily the last level acknowledged and serviced. A non-specific EOI can be issued with OCW2 (EOI = 1, SL = 0, R = 0).

When a mode is used which may disturb the fully nested structure, the Interrupt Controller may no longer be able to determine the last level acknowledged. In this case a Specific End of Interrupt must be issued which includes as part of the command the IS level to be reset. A specific EOI can be issued with OCW2 (EOI = 1, SL = 1, R = 0, and LO-L2 is the binary level of the IS bit to be set to 0).

It should be noted that an IS bit that is masked by an IMR bit will not be cleared by a non-specific EOI if the Interrupt Controller is in the Special Mask Mode.

5.8.6.2 Automatic End of Interrupt (AEOI) Mode

If AEOI = 1 in ICW4, then the Interrupt Controller will operate in AEOI mode continuously until reprogrammed by ICW4. Note that reprogramming ICW4 implies that ICW1, ICW2, and ICW3 must be reprogrammed first, in sequence. In this mode, the Interrupt Controller will automatically perform a non-specific EOI operation at the trailing edge of the last interrupt acknowledge pulse. Note that from a system standpoint, this mode should be used only when a nested multi-level interrupt structure is not required within a single Interrupt Controller. The AEOI mode can only be used in a master Interrupt Controller and not a slave (on CNTRL-1 but not CNTRL-2).

5.8.7 MODES OF OPERATION

5.8.7.1 Fully Nested Mode

This mode is entered after initialization unless another mode is programmed. The interrupt requests are ordered in priority from 0 through 7 (0 being the highest). When an interrupt is acknowledged, the highest priority request is determined and its vector placed on the bus. Additionally, a bit of the Interrupt Service Register (IS[0:7]) is set. This IS bit remains set until the microprocessor issues an End of Interrupt (EOI) command immediately before returning from the service routine. Or, if the AEOI (Automatic End of Interrupt) bit is set, this IS bit remains set until the trailing edge of the second INTA#. While the IS bit is set, all further interrupts of the same or lower priority are inhibited, while higher levels will generate an interrupt (which will be acknowledged only if the microprocessor internal interrupt enable flip-flop has been re-enabled through software).

After the initialization sequence, IRQ0 has the highest priority and IRQ7 the lowest. Priorities can be changed, as will be explained, in the rotating priority mode.

5.8.7.2 The Special Fully Nested Mode

This mode will be used in the case of a system where cascading is used, and the priority has to be conserved within each slave. In this case, the special fully nested mode will be programmed to the master (using ICW4). This mode is similar to the normal nested mode with the following exceptions:

- When an interrupt request from a certain slave is in service, this slave is not locked out from the master's priority logic and further interrupt requests from higher priority IRQs within the slave will be recognized by the master and will initiate interrupts to the processor. (In the normal nested mode, a slave is masked out when its request is in service and no higher requests from the same slave can be serviced.)
- When exiting the Interrupt Service routine, the software has to check whether the interrupt serviced was the only one from that slave. This is done by sending a non-specific End of Interrupt (EOI) command to the slave and then reading its In-Service Register and checking for zero. If it is empty, a non-specific EOI can be sent to the master too. If not, no EOI should be sent.

5.8.7.3 Automatic Rotation (Equal Priority Devices)

In some applications, there are a number of interrupting devices of equal priority. Automatic rotation mode provides for a sequential 8-way rotation. In this mode, a device receives the lowest priority after being serviced. In the worst case, a device requesting an interrupt will have to wait until each of seven other devices are serviced at most once. Figure 5-55 shows an example of automatic rotation.

There are two ways to accomplish automatic rotation using OCW2; the Rotation on Non-Specific EOI Command (R=1, SL=0, EOI=1) and the Rotate in Automatic EOI Mode which is set by (R=1, SL=0, EOI=0) and cleared by (R=0, SL=0, EOI=0).

5.8.7.4 Specific Rotation (Specific Priority)

The programmer can change priorities by programming the bottom priority and thus fixing all other priorities. For example, if IRQ5 is programmed as the bottom priority device, then IRQ6 will be the highest priority device.

The Set Priority Command is issued in OCW2 where: R=1, SL=1; LO-L2 is the binary priority level code of the bottom priority device. See the register description for the bit definitions.

Note that, in this mode, internal status is updated by software control during OCW2. However, it is independent of the End of Interrupt (EOI) command (also executed by OCW2). Priority changes can be executed during an EOI command by using the Rotate on Specific EOI Command in OCW2 (R=1, SL=1, EOI=1 and LO-L2=IRQ level to receive bottom priority).

5.8.7.5 Poll Command

The Polled Mode can be used to conserve space in the interrupt vector table. Multiple interrupts that can be serviced by one interrupt service routine do not need separate vectors if the service routine uses the poll command.

The Polled Mode can also be used to expand the number of interrupts. The polling interrupt service routine can call the appropriate service routine, instead of providing the interrupt vectors in the vector table.

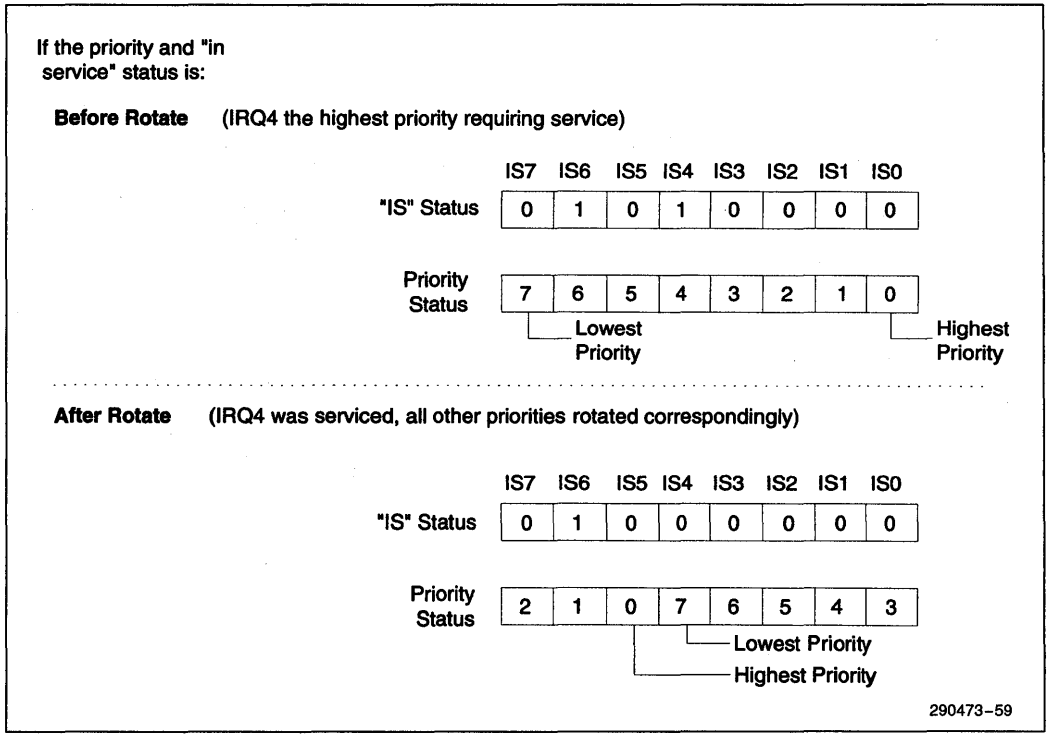
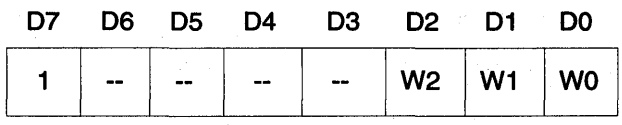


Figure 5-55. Automatic Rotation Mode Example

In this mode, the INT output is not used and the microprocessor internal Interrupt Enable flip-flop is reset, disabling its interrupt input. Service to devices is achieved by software using a Poll Command.

The Poll command is issued by setting P=1 in OCW3. The Interrupt Controller treats the next I/O read pulse to the Interrupt Controller as an interrupt acknowledge, sets the appropriate IS bit if there is a request, and reads the priority level. Interrupts are frozen from the I/O write to the I/O read.

The word enabled onto the data bus during I/O read is:



290473-60

W0-W2: Binary code of the highest priority level requesting service.
Bit D7 – I: Equal to 1 if there is an interrupt.

Figure 5-56. Polled Mode

This mode is useful if there is a routine command common to several levels so that the INTA# sequence is not needed (saves ROM space).

5.8.7.6 Cascade Mode

The Interrupt Controllers in the SIO are interconnected in a cascade configuration with one master and one slave. This configuration can handle up to 15 separate priority levels.

The master controls the slaves through a three line internal cascade bus. When the master drives 010b on the cascade bus, this bus acts like a chip select to the slave controller.

In a cascade configuration, the slave interrupt outputs are connected to the master interrupt request inputs. When a slave request line is activated and afterwards acknowledged, the master will enable the corresponding slave to release the interrupt vector address during the second INTA# cycle of the interrupt acknowledge sequence.

Each Interrupt Controller in the cascaded system must follow a separate initialization sequence and can be programmed to work in a different mode. An EOI Command must be issued twice: once for the master and once for the slave.

5.8.7.7 Edge and Level Triggered Modes

Edge/level sensitivity is programmed per controller. Every IRQ input for a given bank (CNTRL-1 or CNTRL-2) is either edge or level triggered. Bit 3 of ICW1 selects either edge or level triggered interrupts for CNTRL-1 and CNTRL-2.

If an LTIM bit=0, an interrupt request will be recognized by a low to high transition on the corresponding IRQ input. The IRQ input can remain high without generating another interrupt.

If an LTIM bit=1, an interrupt request will be recognized by a high level on the corresponding IRQ input and there is no need for an edge detection. The interrupt request must be removed before the EOI command is issued or the CPU interrupt is enabled to prevent a second interrupt from occurring.

In both the edge and level triggered modes, the IRQ inputs must remain active until after the falling edge of the first INTA#. If the IRQ input goes inactive before this time, a default IRQ7 will occur when the CPU acknowledges the interrupt. This can be a use-

ful safeguard for detecting interrupts caused by spurious noise glitches on the IRQ inputs. To implement this feature, the IRQ7 routine is used for "clean up" simply executing a return instruction, thus ignoring the interrupt. If IRQ7 is needed for other purposes, a default IRQ7 can still be detected by reading the ISR. A normal IRQ7 interrupt will set the corresponding ISR bit; a default IRQ7 will not set this bit. If a default IRQ7 routine occurs during a normal IRQ7 routine, however, the ISR will remain set. In this case, it is necessary to keep track of whether or not the IRQ7 routine was previously entered. If another IRQ7 occurs, it is a default.

5.8.8 REGISTER FUNCTIONALITY

For a detailed description of the Interrupt Controller register set, please see Section 4.4, Interrupt Controller Register Description.

5.8.8.1 Initialization Command Words

Four initialization command words (ICWs) are used to initialize each interrupt controller. Each controller is initialized separately. Following this initialization sequence, the interrupt controller is ready to accept interrupts.

5.8.8.2 Operation Control Words (OCWS)

After the Initialization Command Words (ICWs) are programmed into the Interrupt Controller, the SIO is ready to accept interrupt requests at its input lines. However, Interrupt Controller operation can be dynamically modified to fit specific software/hardware expectations. Different modes of operation are dynamically selected following initialization through the use of Operation Command Words (OCWs).

5.8.9 INTERRUPT MASKS

5.8.9.1 Masking on an Individual Interrupt Request Basis

Each interrupt request input can be masked individually by the Interrupt Mask Register (IMR). This register is programmed through OCW1. Each bit in the IMR masks one interrupt channel, if it is set to a 1. Bit 0 masks IRQ0, Bit 1 masks IRQ1 and so forth. Masking an IRQ channel does not affect the other channel's operation, with one exception. Masking IRQ2 on CNTRL-1 will mask off all requests for service from CNTRL-2. The CNTRL-2 INT output is physically connected to the CNTRL-1 IRQ2 input.

5.8.9.2 Special Mask Mode

Some applications may require an interrupt service routine to dynamically alter the system priority structure during its execution under software control. For example, the routine may wish to inhibit lower priority requests for a portion of its execution but enable some of them for another portion.

The difficulty is that if an Interrupt Request is acknowledged and an End of Interrupt command did not reset its IS bit (i.e., while executing a service routine), the Interrupt Controller would have inhibited all lower priority requests with no easy way for the routine to enable them.

The Special Mask Mode enables all interrupts not masked by a bit set in the Mask Register. Interrupt service routines that require dynamic alteration of interrupt priorities can take advantage of the Special Mask Mode. For example, a service routine can inhibit lower priority requests during a part of the interrupt service, then enable some of them during another part.

In the Special Mask Mode, when a mask bit is set to 1 in OCW1, it inhibits further interrupts at that level and enables interrupts from all other levels (lower as well as higher) that are not masked.

Thus, any interrupts may be selectively enabled by loading the Mask Register with the appropriate pattern.

Without Special Mask Mode, if an interrupt service routine acknowledges an interrupt without issuing an EOI to clear the IS bit, the interrupt controller inhibits all lower priority requests. The Special Mask Mode provides an easy way for the interrupt service routine to selectively enable only the interrupts needed by loading the Mask register.

The special Mask Mode is set by OCW3 where: SSMM = 1, SMM = 1, and cleared where SSMM = 1, SMM = 0.

5.8.10 READING THE INTERRUPT CONTROLLER STATUS

The input status of several internal registers can be read to update the user information on the system. The Interrupt Request Register (IRR) and In-Service Register (ISR) can be read via OCW3, as discussed in Section 3.7. The Interrupt Mask Register (IMR) is read via a read of OCW1, as discussed in Section 3.7. Here are brief descriptions of the ISR, the IRR, and the IMR.

Interrupt Request Register (IRR): 8-bit register which contains the status of each interrupt request line. Bits that are clear indicate interrupts that have not requested service. The Interrupt Controller clears the IRR's highest priority bit during an interrupt acknowledge cycle. (Not affected by IMR).

In-Service Register (ISR): 8-bit register indicating the priority levels currently receiving service. Bits that are set indicate interrupts that have been acknowledged and their interrupt service routine started. Bits that are cleared indicate interrupt requests that have not been acknowledged, or interrupt request lines that have not been asserted. Only the highest priority interrupt service routine executes at any time. The lower priority interrupt services are suspended while higher priority interrupts are serviced. The ISR is updated when an End of Interrupt Command is issued.

Interrupt Mask Register (IMR): 8-bit register indicating which interrupt request lines are masked.

The IRR can be read when, prior to the I/O read cycle, a Read Register Command is issued with OCW3 (RR=1, RIS=0).

The ISR can be read when, prior to the I/O read cycle, a Read Register Command is issued with OCW3 (RR=1, RIS=1).

The interrupt controller retains the ISR/IRR status read selection following each write to OCW3. Therefore, there is no need to write an OCW3 before every status read operation, as long as the current status read corresponds to the previously selected register. For example, if the ISR is selected for status read by an OCW3 write, the ISR can be read over and over again without writing to OCW3 again. However, to read the IRR, OCW3 will have to be reprogrammed for this status read prior to the OCW3 read to check the IRR. This is not true when poll mode is used. Polling Mode overrides status read when P=1, RR=1 in OCW3.

After initialization the Interrupt Controller is set to read the IRR.

As stated, OCW1 is used for reading the IMR. The output data bus will contain the IMR status whenever I/O read is active the address is 021h or 061h (OCW1).

5.8.11 NON-MASKABLE INTERRUPT (NMI)

An NMI is an interrupt requiring immediate attention and has priority over the normal interrupt lines (IRQx). The SIO indicates error conditions by generating a non-maskable interrupt.

NMI interrupts are caused by the following conditions:

1. System Errors on the PCI Bus. SERR# is driven low by a PCI resource when this error occurs.
2. Parity errors on the add-in memory boards on the ISA expansion bus. IOCHK# is driven low when this error occurs.

The NMI logic incorporates two different 8-bit registers. These registers are addressed at locations 061h and 070h. The status of Port (061h) is read by the CPU to determine which source caused the NMI. Bits set to 1 in these ports show which device requested an NMI interrupt. After the NMI interrupt routine processes the interrupt, the NMI status bits are cleared by the software. This is done by setting the corresponding enable/disable bit high. Port (070H) is the mask register for the NMI interrupts. This register can mask the NMI signal and also disable or enable all NMI sources.

Table 5-27. NMI Source Enable/Disable And Status Port Bits

NMI Source	IO Port Bit for status reads	IO Port Bit for enable/disable
IOCHK#	Port 061h, Bit 6	Port 061h, Bit 3
SERR#	Port 061h, Bit 7	Port 061h, Bit 2

The individual enable/disable bits clear the NMI detect flip-flops when disabled.

All NMI sources can be enabled or disabled by setting Port 070h bit 7 to a 0 or 1. This disable function does not clear the NMI detect flip-flops. This means, if NMI is disabled then enabled via Port 070h, then an NMI will occur when Port 070h is re-enabled if one of the NMI detect flip-flops had been previously set.

To ensure that all NMI requests are serviced, the NMI service routine software needs to incorporate a few very specific requirements. These requirements are due to the edge detect circuitry of the host microprocessor, 80386 or 80486. The software flow would need to be the following:

1. NMI is detected by the processor on the rising edge of the NMI input.
2. The processor will read the status stored in port 061h to determine what sources caused the NMI. The processor may then set to 0 the register bits controlling the sources that it has determined to be active. Between the time the processor reads the NMI sources and sets them to a 0, an NMI may have been generated by another source. The level of NMI will then remain active. This new NMI source will not be recognized by the processor because there was no edge on NMI.
3. The processor must then disable all NMI's by setting bit 7 of port 070H to a 1 and then enable all NMI's by setting bit 7 of port 070H to a 0. This will cause the NMI output to transition low then high if there are any pending NMI sources. The CPU's NMI input logic will then register a new NMI.

Section 4.5.1, under Interrupt Control Registers, contains a detailed description of the NMI Status and control register (port 061h) and the NMI Enable/Disable function at port 070h.

5.9 Utility Bus Peripheral Support

The Utility Bus is a secondary bus buffered from the ISA Bus used to interface with peripheral devices that do not require a high speed interface. The buffer control for the lower 8 data signals is provided by the SIO via two control signals; UBUSOE# and UBUSTR. Figure 5-57 shows a block diagram of the external logic required as part of the decode and Utility Bus buffer control.

The SIO provides the address decode and three encoded chip selects to support:

1. Floppy Controller
2. Keyboard Controller
3. Real Time Clock
4. IDE drive
5. 2 Serial Ports (COM1 and COM2)
6. 1 Parallel Port (LPT1, 2, or 3)
7. BIOS memory
8. Configuration Memory (8k I/O mapped)

The SIO also supports the following functions:

1. Floppy DSKCHG function
2. Port 92 Function (Alternate A20 and Alternate Reset)
3. Coprocessor logic (FERR# and IGNNE# function)

The binary code formed by the three Encoded Chip Selects determines which Utility Bus device is selected. The SIO also provides an encoded chip select enable signal (ECSEN#) that is used to select between the two external decoders. A zero selects decoder 1 and a one selects decoder 2. The table below shows the address decode for each of the Utility Bus devices.

Table 5-28. Encoded Chip Select Summary Table

ECS2	ECS1	ECS0	ECSEN#	Address Decoded	External Chip Select	Note	Cycle Type
Decoder 1							
0	0	0	0	70h, 72h, 74, 76h	RTCALE#		I/O W
0	0	1	0	71h, 73h, 75h, 77h	RTCCS#		I/O R/W
0	1	0	0	60h, 62h, 64h, 66h	KEYBRDCS#		I/O R/W
0	1	1	0	000E0000h-000FFFFFh FFFE0000h-FFFFFFFFh FFF80000h-FFFDFFFFh	BIOSCS#	1	MEM R/W
1	0	0	0	3F0h-3F7h (primary) 370h-377h (secondary)	FLOPPYCS#	2	I/O R/W
1	0	1	0	1F0h-1F7h (primary) 170h-177h (secondary)	IDECS0#	2	I/O R/W
1	1	0	0	3F6h-3F7h (primary) 376h-377h (secondary)	IDECS1#	2	I/O R/W
1	1	1	0	Reserved			
Decoder 2							
0	0	0	1	Reserved			
0	0	1	1	0C00h	CPAGECS#	3	I/O R/W
0	1	0	1	0800h-08FFh	CFIGMEMCS#	3	I/O R/W
0	1	1	1	3F8h-3FFh (COM1) -or- 2F8h-37Fh (COM2)	COMACS#	4	I/O R/W
1	0	0	1	3F8h-3FFh (COM1) -or- 2F8h-37Fh (COM2)	COMBCS#	4	I/O R/W
1	0	1	1	3BCh-3BFh (LPT1) 378h-37Fh (LPT2) 278h-27Fh (LPT3)	LPTCS#	5	I/O R/W
1	1	0	1	Reserved			
1	1	1	1	Idle State			

NOTES:

1) The encoded chip select signals for BIOSCS# will always be generated for accesses to the upper 64 KB at the top of 1 MByte (F0000h-FFFFFh) and its aliases at the top of the 4 GB and 4 GB-1 MByte. Access to the lower 64 KByte (E0000h - EFFFFh) and its aliases at the top of 4 GB and 4GB - 1MB can be enabled or disabled through the SIO. An additional 384 KB of BIOS memory at the top of 4 GB (FFFD0000h-FFFDFFFFh) can be enabled for BIOS use.
2) The primary and secondary locations are programmable through the SIO. Only one location range can be enabled at any one time. The floppy and IDE share the same enable and disable bit (i.e. if the floppy is set for primary, the IDE is also set for primary).
3) These signals can be used to select additional configuration RAM.
4) COM1 and COM2 address ranges can be programmed for either port A (COMACS#) or port B (COMBCS#).
5) Only one address range (LPT1, LPT2, or LPT3) can be programmed at any one time.

Port 92h Function:

The SIO integrates the Port 92h Register. This register provides the alternate reset (ALTRST) and alternate A20 (ALT—A20) functions. Figure 5-57 shows how these functions are tied into the system.

DSKCHG Function:

DSKCHG is tied directly to the DSKCHG signal of the floppy controller. This signal is inverted and driven onto system data line 7 (SD7) during I/O read cycles to floppy address locations 3F7h (primary) or 377 (secondary) as indicated by the table below.

Table 5-29. DSKCHG Summary Table

FLOPPYCS# Decode	IDECSx# Decode	State of SD7 (output)	State of UBUSOE#
Enabled	Enabled	Tri-stated	Enabled
Enabled	Disabled	Driven via DSKCHG	Disabled
Disabled	Enabled	Tri-stated	Disabled 1
Disabled	Disabled	Tri-stated	Disabled

NOTE:
1. This mode is not supported because of potential contention between the Utility bus buffer and a floppy on the ISA Bus driving the system bus at the same time during shared I/O accesses.

Coprocessor Error Support:

If bit 5 in the ISA Clock Divisor Register is set to a one, the SIO will support coprocessor error reporting through the FERR#/IRQ13 signal.

FERR# is tied directly to the Coprocessor error signal of the CPU. If FERR# is driven active in this mode (coprocessor error detected by the CPU), an internal IRQ13 is generated and the INT output from the SIO is driven active. When a write to I/O location F0h is detected, the SIO negates IRQ13 and drives IGNNE# active. IGNNE# remains active until FERR# is driven inactive. Note that IGNNE# is not generated unless FERR# is active.

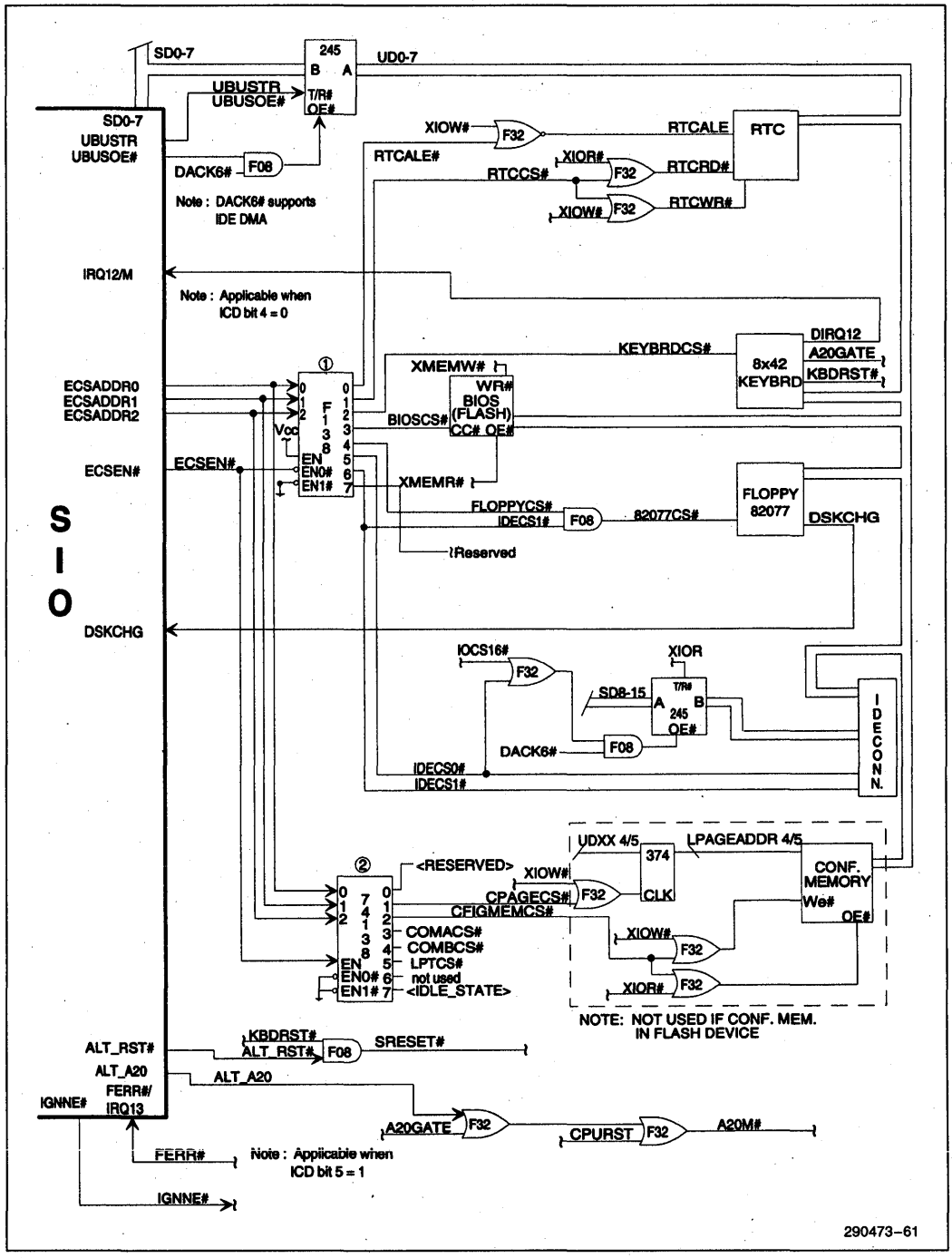


Figure 5-57. Utility Bus External Support Logic

Utility Bus accesses by the SIO, by an ISA master, and by the DMA is shown in Figure 5-58 and 5-59. UBUSOE# and UBUSTR are driven differently for DMA cycles as shown in figure 5-59.

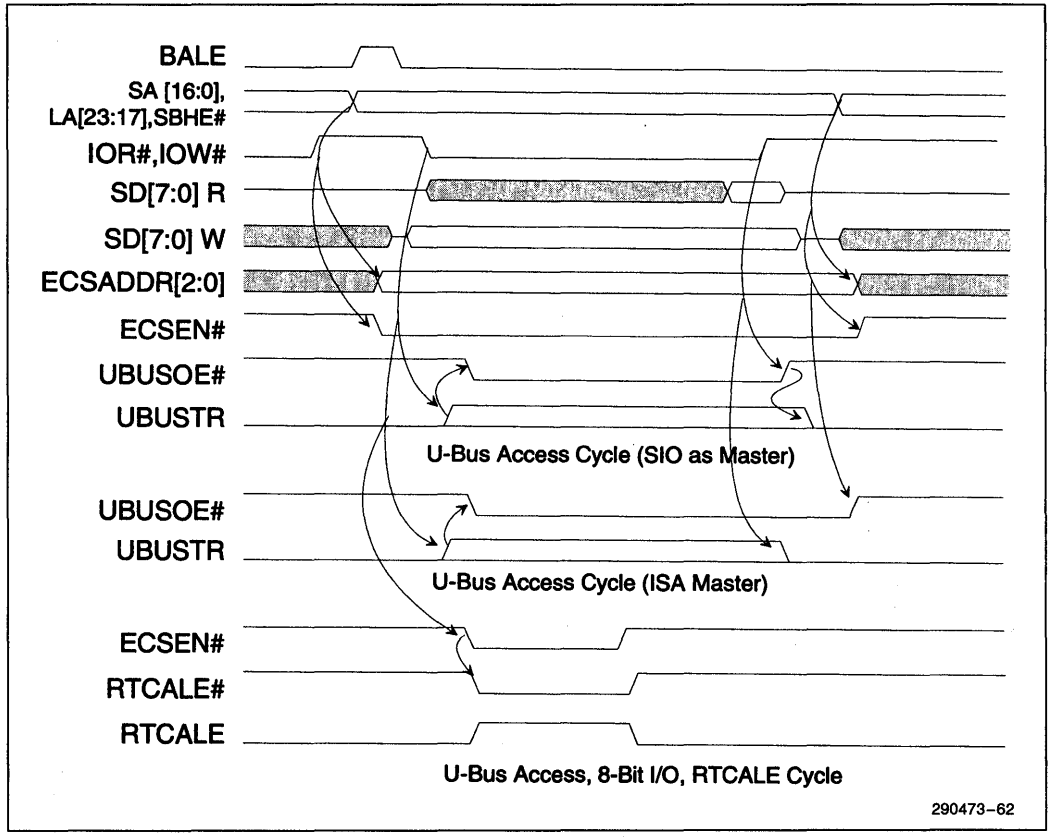


Figure 5-58. Utility Bus Access (SIO and ISA Master)

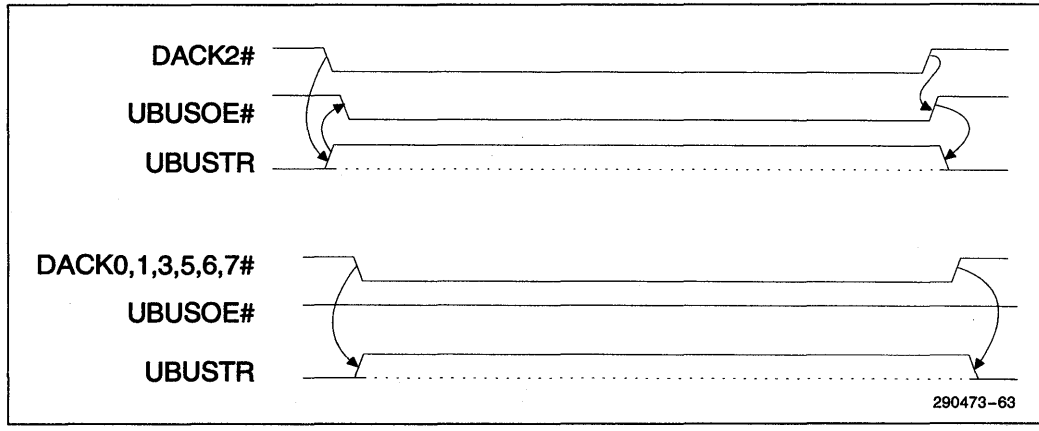


Figure 5-59. Utility Bus Access (DMA)

6.0 ELECTRICAL CHARACTERISTICS

6.1 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

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 reliability.

6.2 DC Characteristics

6.2.1 ISA AND UTILITY BUS DC SPECIFICATIONS

ISA Signals
SD[15:0] (I/O), SA[19:0] (I/O), LA[23:17] (I/O), SBHE (I/O), MEMR# (I/O), AEN (O), BALE (O), SYSCLK (O), IOR# (I/O), IOW# (I/O), SMEMR# (O), SMEMW# (O), RSTDRV (O), REFRESH# (I/O), IOCS16# (I/O), IOCHRDY (I/O), MEMCS16# (I/O), EOP (I/O), DACK[7:5, 3:0] (O), OSC (I), IOCHK# (I), MASTER# (I), ZEROWS# (I), DREQ[3:0, 7:5] (I), INT (O), NMI (O), IRQ [15,14,11:3,1] (I)

Utility Bus Signals
UBUSTR (O), UBUSOE# (O), ECSADDR[2:0] (O), ECSEN# (O), ALT_RST# (O), ALT_A20 (O), DSKCHG (I), IGNNE# (O), SPKR (O), FERR#/IRQ13 (I), TEST (I), IRQ12/M (I)

Table 6-1. ISA and Utility Bus D.C. Characteristics (V_{DD} = 5V ±5%, T_{case} = 0 to 85°C)

Symbol	Parameter	Min	Max	Units	Test Conditions	Notes
V_{IL}	Input Low Voltage		0.8	V		
V_{IH}	Input High Voltage	2.0		V		
V_{OL1}	Output Low Voltage		0.45	V	I_{OL} = 24mA	1
V_{OH1}	Output High Voltage	2.4		V	I_{OH} = −4.0mA	1
V_{OL2}	Output Low Voltage		0.45	V	I_{OL} = 8.0mA	2
V_{OH2}	Output High Voltage	2.4		V	I_{OH} = −2.0mA	2
V_{OL3}	Output Low Voltage		0.45	V	I_{OL3} = 4mA	3
V_{OH3}	Output High Voltage	2.4		V	I_{OH3} = −2mA	3
I_{LI1}	Input Leakage Current		±15	μA	$0V < V_{IN} < V_{CC}$	
I_{LI2}	Input Leakage Current		−350	μA	$0V < V_{IN} < V_{CC}$	4
I_{LO}	Output Leakage		±15	μA	$0.45 < V_{IN} < V_{CC}$	
C_{IN}	Capacitance Input		8	pF		
C_{OUT}	Capacitance Output or I/O		15	pF	@1MHz	
C_{CLK}	SYSCLK Capacitance		15	pF	@1MHz	
I_{CC}	V_{CC} Supply Current		200	ma	TBD	

NOTE:
1. V_{OL1}, V_{OH1} = SD[15:0], SA[19:0], LA[23:17], SBHE#, MEMR#, MEMW#, AEN, SPKR, BALE, SYSCLK, IOR#, IOW#, SMEMR#, SMEMW#, RSTDRV, REFRESH#, IOCS16#, IOCHRDY, MEMCS16#, EOP
2. V_{OL2} V_{OH2} = DACK#[7:5,3:0]
3. V_{OL3}, V_{OH3} = UBUSTR, UBUSOE#, ECSADDR[2:0], ECSEN#, ALT_RST#, ALT_A20, IGNNE#, INT, NMI
4. This applies to pins that include a weak internal pull-up (IRQ8#).

6.2.2 PCI INTERFACE DC SPECIFICATIONS

PCI System Signal Group
PCICLK (I), PCIRST# (I)

PCI Shared Signal Group
AD[31:0] (I/O), C/BE#[3:0] (I/O), FRAME# (I/O), TRDY# (I/O), IRDY# (I/O), STOP# (I/O), LOCK# (I), IDSEL (I), DEVSEL# (I/O), PAR (O), SERR# (I)

PCI Sideband Signal Group
MEMREQ# (O), MEMACK# (I), FLSHREQ# (O), MEMCS# (O), GNT1#/RESUME# (O), GNT0#/SIOREQ (O), CPUGNT# (O), REQ1# (I), REQ0#/SIOGNT# (I), CPUREQ# (I)

Table 6-2. D.C. Characteristics (V_{DD} = 5V ±5%, T_{case} = 0 to 85°C)

Symbol	Parameter	Min	Max	Units	Test Conditions	Notes
V_{IL}	Low-level Input Voltage		0.8	V		
V_{IH}	High-level Input Voltage	2.0	5.5	V		
I_{IL}	Low-level Input Current		−70	μA	V_{IN} = 0.5V	1
I_{IH}	High-level Input Current		70	μA	V_{IN} = 2.7V	1
V_{OH}	High Level Output Voltage	2.4		V	I_{OH} = −2mA	
V_{OL}	Low-level Output Voltage		.55	V	I_{OL1} = 3mA I_{OL2} = 6mA	2
$C_{I/O}$	Input/Output Capacitance		10	pF	Frequency = 1MHz	3
C_{PCICLK}	PCICLK Signal Input Capacitance		17	pF	Frequency = 1MHz	
I_{CC}	V_{CC} Supply Current		200	ma		

NOTE:
1. For the bi-directional signals, these parameters also include any leakage current from a tri-state output, IOZL or IOZH.
2. I_{OL2} Applies to those signals requiring external pull-ups: FRAME#, TRDY#, IRDY#, STOP#, LOCK#, DEVSEL#, PAR, CPUGNT#, GNT0#/SIOREQ#, GNT1#/RESUME, MEMREQ# and FLSHREQ#.
3. Does not include PCICLK.

6.3 A.C. Characteristics

6.3.1 ISA AND UTILITY BUS AC SPECIFICATIONS

Type: M = Memory Cycle, I/O = I/O Cycle
Size: 8 = 8-Bit Transfer, 16 = 16-Bit Transfer

Table 6-3. ISA and Utility Bus A.C. Characteristics (V_{DD} = 5V ±5%, T_{case} = 0 to 85 °C)

Sym	Parameter	8.33 MHz		Units	Type	Size	Notes	Fig.
		Min	Max					
ISA CLOCK TIMINGS								
	SYSCLK							
t1a	Period (0.8V to 0.8V)	120		ns				6-2
t1b	High time (2.0V)	56		ns				6-2
t1c	Low time (0.8V)	56		ns				6-2
t1d	Rise/Fall time (0.8V <=> 2.0V)		4	ns				6-2
t1e	PCICLK to SYSCLK valid delay		25	ns				6-3

Table 6-3. ISA and Utility Bus A.C. Characteristics (V_{DD} = 5V ±5%, T_{case} = 0 to 85 °C) (Continued)

Sym	Parameter	8.33 MHz		Units	Type	Size	Notes	Fig.
		Min	Max					
	ISA CLOCK TIMINGS (Continued)							
	OSC							
t3a	Period (0.8V to 0.8V)	65	70	ns				6-2
t3b	High time (2.0V)	20		ns				6-2
t3c	Low time (0.8V <=> 2.0V)	20		ns				6-2
	SIO AS MASTER TIMINGS							
	BALE							
t4a	BALE pulse width	52		ns	M,I/O	8,16		6-4,5,6,7
t4b	BALE driven active from MEMx#, IOx# inactive	44		ns	M,I/O	8,16		6-4,5,6,7
	LA[23:17]							
t5a	LA[23:17] valid setup to BALE inactive	104		ns	M	8,16		6-4,5
t5b	LA[23:17] valid hold from BALE inactive	26		ns	M	8,16		6-4,5
t5c	LA[23:17] valid setup to MEMx# active	113		ns	M	16		6-5
t5d	LA[23:17] valid setup to MEMx# active	173		ns	M	8		6-4
t5e	LA[23:17] invalid from MEMx# active	39		ns	M	16		6-5
t5f	LA[23:17] invalid from MEMx# active	39		ns	M	8		6-4
	SA[19:0],SBHE#							
t6a	SA[19:0],SBHE# valid setup to MEMx# active	34		ns	M	16		6-5
t6b	SA[19:0],SBHE# valid setup to IOx# active	100		ns	I/O	16		6-7
t6c	SA[19:0],SBHE# setup to MEMx#, IOx# active	100		ns	M,I/O	8		6-4,6
t6d	SA[19:0],SBHE# valid setup to BALE inactive	37		ns	M,I/O	8,16		6-4,5,6,7
t6e	SA[19:0],SBHE# valid hold from MEMx#, IOx# inactive	51		ns	M,I/O	8,16		6-4,5,6,7
	MEMR#, MEMW#, IOR# AND IOW#							
t7a	MEMx# active pulse width (std)	225		ns	M	16		6-5
t7b	IOx# active pulse width (std)	160		ns	I/O	16		6-7
t7c	MEMx# active pulse width (nws)	105		ns	M	16	1	6-5
t7d	MEMx# or IOx# active pulse width (std)	520		ns	M,I/O	8		6-4,6
t7e	MEMx# or IOx# active pulse width (nws)	160		ns	M,I/O	8	1	6-4,6
t7f	MEMx# inactive pulse width	103		ns	M	16		6-5

Table 6-3. ISA and Utility Bus A.C. Characteristics (V_{DD} = 5V ±5%, T_{case} = 0 to 85 °C) (Continued)

Sym	Parameter	8.33 MHz		Units	Type	Size	Notes	Fig.
		Min	Max					
	SIO AS MASTER TIMINGS (Continued)							
t7g	MEMx# inactive pulse width	163		ns	M	8		6-4
t7h	IOx# inactive pulse width	163		ns	I/O	8,16		6-6,7
t7i	MEMx#, IOx# driven inactive from IOCHRDY active	120		ns	M,I/O	8,16		6-4,5,6,7
	SMEMR# and SMEMW#							
t8a	SMEMR# & SMEMW# propagation delay from MEMR# and MEMW#		5	ns	M	8,16		6-4,5
	READ DATA							
t9a	Read data driven from MEMR#, IOR# active	0		ns	M,I/O	8,16		6-4,5,6,7
t9b	Read data valid setup to MEMR#, IOR#	20		ns	M,I/O	8,16		6-4,5,6,7
t9c	Read data valid hold from MEMR#, IOR# inactive	0		ns	M,I/O	8,16		6-4,5,6,7
t9d	Read data tri-stated from MEMR# and IOR# inactive		41	ns	M,I/O	8,16		6-4,5,6,7
	WRITE DATA							
t10a	Write data valid setup to MEMW#, IOW# active	26		ns	M,I/O	8,16		6-4,5,6,7
t10b	Write data valid hold from MEMW#, IOW# inactive	45		ns	M,I/O	8,16		6-4,5,6,7
t10c	Write data tri-stated from MEMW#, IOW# inactive		75	ns	M,I/O	8,16		6-4,5,6,7
t10d	Write data driven valid after read MEMR#, IOR# inactive	41		ns	M,I/O	8,16		6-4,5,6,7
	MEMCS16#							
t11a	MEMCS16# driven active from LA[23:17] valid		94	ns	M	16		6-5
t11b	MEMCS16# inactive from LA[23:17] valid		91	ns	M	8		6-4
t11c	MEMCS16# valid hold from LA[23:17] invalid	0		ns	M	16		6-5
t11d	MEMCS16# driven active from SA[19:2] valid		35	ns	M	16		6-5
	IOCS16#							
t12a	IOCS16# driven active from valid SA[19:0]		117	ns	I/O	16		6-7
t12b	IOCS16# inactive from valid SA[19:0]		91	ns	I/O	8		6-6
t12c	IOCS16# valid hold from SA[19:0] invalid	0		ns	I/O	16		6-7
t12d	IOCS16# driven active from IOx active		75	ns	I/O	16		6-7

Table 6-3. ISA and Utility Bus A.C. Characteristics (V_{DD} = 5V ±5%, T_{case} = 0 to 85 °C) (Continued)

Sym	Parameter	8.33 MHz		Units	Type	Size	Notes	Fig.
		Min	Max					
	SIO AS MASTER TIMINGS (Continued)							
	ZEROWS#							
t13a	ZEROWS# driven active from MEMx# active		27	ns	M	16		6-5
t13b	ZEROWS# driven active from MEMx#, IOx# active		80	ns	M,I/O	8		6-4,6
t13c	ZEROWS# driven active from LA[23:17] valid		180	ns	M	16		6-5
t13d	ZEROWS# driven active from LA[23:17] valid		300	ns	M	8		6-4
t13e	ZEROWS# driven active from SA[19:0], SBHE# valid		80	ns	M	16		6-5
t13f	ZEROWS# driven active from SA[19:0], SBHE# valid		200	ns	M,I/O	8		6-4,6
t13g	ZEROWS# valid hold from the falling edge of SYSCLK		22	ns	M,I/O	8		6-4,6
	AEN							
t14a	AEN valid setup to IOx# driven active	111		ns	I/O	8,16		6-6,7
t14b	AEN valid setup to BALE driven inactive	111		ns	I/O	8,16		6-6,7
t14c	AEN valid hold from IOx# driven inactive	41		ns	I/O	8,16		6-6,7
	IOCHRDY							
t15a	IOCHRDY driven inactive from MEMx#, IOx# active		76	ns	M,I/O	16		6-5,7
t15b	IOCHRDY driven inactive from MEMx#, IOx# active		366	ns	M,I/O	8		6-4,6
t15c	IOCHRDY valid from MEMx#, IOx# active		76	ns	M,I/O	16		6-5,7
t15d	IOCHRDY valid from MEMx#, IOx# active		366	ns	M,I/O	8		6-4,6
t15e	IOCHRDY inactive pulse width	120	15.6	µs	M,I/O	8,16		6-4,5,6,7
	SIO AS SLAVE TIMINGS							
	LA[23:17]							
t16a	LA[23:17] valid setup to MEMx# active	102		ns	M	16		6-8
t16b	LA[23:17] invalid from MEMx# active	28		ns	M	16		6-8
	SA[19:0],SBHE#							
t17a	SA[19:0],SBHE# setup to MEMx# active	23		ns	M	16		6-8
t17b	SA[19:0],SBHE# setup to IOx# active	89		ns	I/O	8		6-9
t17c	SA[19:0],SBHE# valid hold from MEMx#, IOx# inactive	40		ns	M,I/O	8,16		6-8,9
	MEMR#, MEMW#, IOR#, IOW#							
t18a	MEMx# active pulse width	214		ns	M	16		6-8
t18b	IOx# active pulse width	509		ns	I/O	8		6-9
t18c	MEMx# inactive pulse width	92		ns	M	16		6-8
t18d	IOx# inactive pulse width	152		ns	I/O	8		6-9

Table 6-3. ISA and Utility Bus A.C. Characteristics (V_{DD} = 5V ±5%, T_{case} = 0 to 85 °C) (Continued)

Sym	Parameter	8.33 MHz		Units	Type	Size	Notes	Fig.
		Min	Max					
	SIO AS SLAVE TIMINGS (Continued)							
	READ DATA							
t19a	Read data valid from IOCHRDY active		69	ns	M,I/O	8,16		6-8,9
t19b	Read data valid from IOR# active.		69	ns	I/O	8	11	6-9
t19c	Read data valid hold from MEMR#, IOR# inactive	0		ns	M,I/O	8,16		6-8,9
t19d	Read data tri-state from MEMR#, IOR# inactive		30	ns	M,I/O	8,16		6-8,9
	WRITE DATA							
t20a	Write data valid setup to MEMW#, IOW# active	−54		ns	M,I/O	8,16		6-8,9
t20b	Write data valid hold from MEMW#, IOW# inactive	14		ns	M,I/O	8,16		6-8,9
	MEMCS16#							
t21a	MEMCS16# driven active from valid LA[23:17]		72	ns	M	16		6-8
t21b	MEMCS16# float from valid LA[23:17]		40	ns	M	16		6-8
t21c	MEMCS16# valid hold from LA[23:17] invalid	0		ns	M	16		6-8
	IOCHRDY							
t22a	IOCHRDY inactive from MEMx#, IOx# active		25	ns	M,I/O	8,16		6-8,9
t22b	IOCHRDY float from IOCHRDY rising.		70	ns	M,I/O	8,16	4	6-8,9
t22c	IOCHRDY inactive pulse width	120	2.5	µs	M,I/O	8,16		6-8,9
	INTERRUPT AND NMI TIMING							
	NMI Timing							
t23a	SERR#, IOCHK# active to NMI driven active		200	ns				6-10
	Interrupt Timing							
t24a	IRQ inactive pulse width	100		ns				6-11
t24b	INT output delay from IRQ# active		100	ns				6-11
	ISA BUS MASTER TIMINGS							
	AEN							
t25a	AEN inactive from MASTER# active	0	49	ns				6-12
t25b	AEN active from MASTER# inactive	71		ns				6-12
	DACK#							
t26a	DACK# inactive from DREQ inactive	240		ns				6-12
	Tri-Stating and Driving the Bus							
t27a	SIO tri-states address, data, and control signals from DACK# active	0	30	ns				6-12
t27b	SIO drives address, data, and control signals from DACK# inactive	71		ns				6-12
	SMEMR# and SMEMW#							
t28a	SMEMR# & SMEMW# valid from MEMR# and MEMW# valid		20	ns				6-12

Table 6-3. ISA and Utility Bus A.C. Characteristics (V_{DD} = 5V ±5%, T_{case} = 0 to 85 °C) (Continued)

Sym	Parameter	8.33 MHz		Units	Type	Size	Notes	Fig.
		Min	Max					
	ISA BUS MASTER TIMINGS (Continued)							
	DATA SWAP LOGIC TIMING (ISA Master to ISA slave)							
t29a	SD[7:0] to SD[15:8] Propagation Delay		15	ns				6-13
t29b	SD[15:8] to SD[7:0] Propagation Delay		15	ns				6-13
t29c	SIO drives data bus from IOR#, IOW#, MEMR# or MEMW# active		20	ns			2	6-13
t29d	SIO tri-states bus from IOR#, MEMR#, or SMEMR# inactive	5	20	ns			2,3	6-13
t29e	SIO tri-states bus from IOW#, MEMW#, or SMEMW# inactive	15	60	ns			2,3	6-13
	DMA COMPATIBLE TIMINGS							
	DREQ							
t30a	DREQ active hold from IOR# active		555	ns			5	6-15
t30b	DREQ active hold from IOW# active		315	ns			5	6-14
	DACK#							
t31a	DACK# active to IOR# active	73		ns				6-15
t31b	DACK# active to IOW# active	312		ns				6-14
t31c	DACK# active hold from IOR# inactive	105		ns				6-15
t31d	DACK# active hold from IOW# inactive	161		ns				6-14
	AEN and BALE							
t32a	AEN active to IOx# active	111		ns				6-14,15
t32b	AEN and BALE inactive from IOx# inactive	41		ns				6-14,15
	LA[23:19], SA[19:0], SBHE#							
t33a	LA[23:19],SA[19:0], SBHE# valid setup to MEMx# active	99		ns				6-14,15
t33b	LA[23:19],SA[19:0], SBHE# valid hold from MEMx# inactive	51		ns				6-14,15
	MEMR#, MEMW#, IOR#, IOW#							
t34a	IOW# and MEMW# active pulse width	474		ns				6-14,15
t34b	MEMR# active pulse width	520		ns				6-14
t34c	IOR# active pulse width	769		ns				6-15
t34d	IOW# inactive pulse width (continuous)	469		ns				6-14
t34e	IOR# inactive pulse width (continuous)	167		ns				6-15
t34f	IOR# active to MEMW# active	235		ns				6-15
t34g	MEMR# active to IOW# active	0		ns				6-15
t34h	MEMR# active hold from IOW# inactive	50		ns				6-14
t34i	IOR# active hold from MEMW# inactive	50		ns				6-15
t34j	MEMx# active hold from IOCHRDY active	120		ns				6-14,15

Table 6-3. ISA and Utility Bus A.C. Characteristics (V_{DD} = 5V ±5%, T_{case} = 0 to 85 °C) (Continued)

Sym	Parameter	8.33 MHz		Units	Type	Size	Notes	Fig.
		Min	Max					
	DMA COMPATIBLE TIMINGS (Continued)							
	SMEMR# & SMEMW#							
t35a	SMEMR# & SMEMW# valid from MEMR# and MEMW# valid		5	ns				6-14,15
	READ DATA							
t36a	Read data valid from IOR# active		237	ns				6-15
t36b	Read data valid hold from IOR# inactive	0		ns				6-15
t36c	Read Data float from IOR# inactive		61	ns				6-15
	WRITE DATA							
t37a	Write data valid setup to IOW# inactive	252		ns				6-14
t37b	Write data valid hold from IOW# inactive	36		ns				6-14
	DATA SWAP LOGIC TIMING (ISA to ISA Transaction)							
t38a	SD[7:0] to SD[15:8] Propagation Delay		15	ns				6-16
t38b	SD[15:8] to SD[7:0] Propagation Delay		15	ns				6-16
t38c	SIO drives data bus from IOR# or MEMR# active		20	ns			2	6-16
t38d	SIO tri-states bus from IOR# or MEMR# inactive		20	ns			2	6-16
	EOP							
t39a	EOP active setup to IOx# inactive	511		ns			6	6-14,15
t39b	EOP active hold from IOx# inactive	71		ns			6	6-14,15
t39c	EOP active delay from IOW# active		315	ns			7	6-14
t39d	EOP active delay from IOR# active		555	ns			7	6-15
t39e	EOP inactive delay from IOW# inactive		112	ns			7	6-14
t39f	EOP inactive delay from IOR# inactive		102	ns			7	6-15
t39g	EOP enable/disable from DACK# inactive		51	ns			7	6-14,15
	IOCHRDY							
t40a	IOCHRDY inactive from MEMx# active		315	ns				6-14,15
t40b	IOCHRDY valid from MEMx# active		315	ns				6-14,15
t40c	IOCHRDY inactive pulse width	125		ns				6-14,15
	DMA TYPE "A" TIMINGS							
	DREQ							
t41a	DREQ active hold from IOR# active		320	ns			5	6-17
t41b	DREQ active hold from IOW# active		200	ns			5	6-17
	DACK#							
t42a	DACK# active to IOR# active	73		ns				6-17
t42b	DACK# active to IOW# active	197		ns				6-17

Table 6-3. ISA and Utility Bus A.C. Characteristics (V_{DD} = 5V ±5%, T_{case} = 0 to 85 °C) (Continued)

Sym	Parameter	8.33 MHz		Units	Type	Size	Notes	Fig.
		Min	Max					
	DMA TYPE "A" TIMINGS (Continued)							
t42c	DACK# active hold from IOR# inactive	105		ns				6-17
t42d	DACK# active hold from IOW# inactive	161		ns				6-17
	AEN and BALE							
t43a	AEN active to IOx# active	111		ns				6-17
t43b	AEN and BALE inactive from IOx# inactive	41		ns				6-17
	IOR# and IOW#							
t44a	IOR# active pulse width	400		ns				6-17
t44b	IOW# active pulse width	340		ns				6-17
t44c	IOR# inactive pulse width (continuous)	167		ns				6-17
t44d	IOW# inactive pulse width (continuous)	345		ns				6-17
	READ DATA							
t45a	Read data valid from IOR# active		302	ns				6-17
t45b	Read data valid hold from IOR# inactive	2		ns				6-17
t45c	Read data float from IOR# inactive		61	ns				6-17
	WRITE DATA							
t46a	Write data valid setup to IOW# inactive	251		ns				6-17
t46b	Write data valid hold from IOW# inactive	36		ns				6-17
	EOP							
t47a	EOP active setup to IOR# inactive	311		ns			6	6-17
t47b	EOP active setup to IOW# inactive	251		ns			6	6-17
t47c	EOP active hold from IOx# inactive	71		ns			6	6-17
t47d	EOP active delay from IOW# active		222	ns			7	6-17
t47e	EOP active delay from IOR# active		342	ns			7	6-17
t47f	EOP inactive delay from IOW# inactive		112	ns			7	6-17
t47g	EOP inactive delay from IOR# inactive		102	ns			7	6-17
t47h	EOP enable/disable from DACK# inactive		51	ns			7	6-17
	DMA TYPE "B" TIMINGS							
	DREQ							
t48a	DREQ active hold from IOR# active		200	ns			5	6-18
t48b	DREQ active hold from IOW# active		80	ns			5	6-18
	DACK#							
t49a	DACK# active to IOR# active	73		ns				6-18
t49b	DACK# active to IOW# active	197		ns				6-18
t49c	DACK# active hold from IOR# inactive	46		ns				6-18
t49d	DACK# active hold from IOW# inactive	140		ns				6-18

Table 6-3. ISA and Utility Bus A.C. Characteristics (V_{DD} = 5V ±5%, T_{case} = 0 to 85 °C) (Continued)

Sym	Parameter	8.33 MHz		Units	Type	Size	Notes	Fig.
		Min	Max					
	DMA TYPE "B" TIMINGS (Continued)							
	AEN and BALE							
t50a	AEN active to IOx# active	111		ns				6-18
t50b	AEN and BALE inactive from IOx# inactive	41		ns				6-18
	IOR# and IOW#							
t51a	IOR# active pulse width	280		ns				6-18
t51b	IOW# active pulse width	220		ns				6-18
t51c	IOR# inactive pulse width (continuous)	55		ns				6-18
t51d	IOW# inactive pulse width (continuous)	225		ns				6-18
	READ DATA							
t52a	Read data valid from IOR# active		182	ns				6-18
t52b	Read data valid hold from IOR# inactive	2		ns				6-18
t52c	Read data float from IOR# inactive		61	ns				6-18
	WRITE DATA							
t53a	Write data valid setup to IOW# inactive	141		ns				6-18
t53b	Write data valid hold from IOW# inactive	36		ns				6-18
	EOP							
t54a	EOP active setup to IOR# inactive	221		ns			6	6-18
t54b	EOP active setup to IOW# inactive	191		ns			6	6-18
t54c	EOP active hold from IOx# inactive	−25		ns			6	6-18
t54d	EOP active delay from IOW# active		82	ns			7	6-18
t54e	EOP active delay from IOR# active		202	ns			7	6-18
t54f	EOP inactive delay from IOW# inactive		102	ns			7	6-18
t54g	EOP inactive delay from IOR# inactive		40	ns			7	6-18
t54h	EOP enable/disable from DACK# inactive		51	ns			7	6-18
	DMA TYPE "F" TIMINGS							
	DREQ							
t55a	DREQ active hold from IOR# active		82	ns			5	6-19
t55b	DREQ active hold from IOW# active		82	ns			5	6-19
	DACK#							
t56a	DACK# active to IOR# active	77		ns				6-19
t56b	DACK# active to IOW# active	77		ns				6-19
t56c	DACK# active hold from IOR# inactive	30		ns				6-19
t56d	DACK# active hold from IOW# inactive	30		ns				6-19
	AEN and BALE							
t57a	AEN active to IOx# active	111		ns				6-19
t57b	AEN and BALE inactive from IOx# inactive	41		ns				6-19

Table 6-3. ISA and Utility Bus A.C. Characteristics (V_{DD} = 5V ±5%, T_{case} = 0 to 85 °C) (Continued)

Sym	Parameter	8.33 MHz		Units	Type	Size	Notes	Fig.
		Min	Max					
	DMA TYPE "F" TIMINGS (Continued)							
	IOR# and IOW#							
t58a	IOR# active pulse width	115		ns				6-19
t58b	IOW# active pulse width	115		ns				6-19
t58c	IOR# inactive pulse width (continuous)	115		ns				6-19
t58d	IOW# inactive pulse width (continuous)	115		ns				6-19
	READ DATA							
t59a	Read data valid from IOR# active		96	ns				6-19
t59b	Read data valid hold from IOR# inactive	2		ns				6-19
t59c	Read data float from IOR# inactive		61	ns				6-19
	WRITE DATA							
t60a	Write data valid setup to IOW# inactive	75		ns				6-19
t60b	Write data valid hold from IOW# inactive	31		ns				6-19
	EOP							
t61a	EOP active setup to IOR# inactive	40		ns			6	6-19
t61b	EOP active setup to IOW# inactive	40		ns			6	6-19
t61c	EOP active hold from IOx# inactive	0		ns			6	6-19
t61d	EOP active delay from IOW# active		70	ns			7	6-19
t61e	EOP active delay from IOR# active		70	ns			7	6-19
t61f	EOP inactive delay from IOW# inactive		40	ns			7	6-19
t61g	EOP inactive delay from IOR# inactive		40	ns			7	6-19
t61h	EOP enable/disable from DACK# inactive		51	ns			7	6-19
	ISA REFRESH TIMINGS							
	REFRESH#							
t62a	REFRESH# active setup to MEMR# active	120		ns				6-20,21
t62b	REFRESH# active hold from MEMR# inactive	31	218	ns				6-20,21
t62c	REFRESH# driven active to SA[15:0] valid	11		ns				6-20,21
t62d	REFRESH# active hold from SA[15:0] invalid	11		ns				6-20,21
	AEN							
t63a	AEN driven active to MEMR# inactive	11		ns				6-20,21
t63b	AEN hold from MEMR# inactive	11		ns				6-20,21
	SA[7:0]							
t64a	SA[15:0] valid setup to MEMR# active	81		ns				6-20,21
t64b	SA[15:0] valid hold from MEMR# inactive	36		ns				6-20,21
t64c	SA[15:0] valid float from MEMR# inactive	45	120	ns			8	6-21

Table 6-3. ISA and Utility Bus A.C. Characteristics (V_{DD} = 5V ±5%, T_{case} = 0 to 85 °C) (Continued)

Sym	Parameter	8.33 MHz		Units	Type	Size	Notes	Fig.
		Min	Max					
	ISA REFRESH TIMINGS (Continued)							
	MEMR#, SMEMR#							
t65a	MEMR# active pulse width	220		ns				6-20,21
t65b	MEMR# tristate from MEMR# inactive	45	120	ns			8	6-20,21
t65c	MEMR# driven inactive from IOCHRDY active	120		ns				6-20,21
t65d	SMEMR# propagation delay from MEMR#		5	ns				6-20,21
	IOCHRDY							
t66a	IOCHRDY inactive from MEMR# active		76	ns				6-20,21
t66b	IOCHRDY valid from MEMR# active		76	ns				6-20,21
t66c	IOCHRDY active to inactive	120		ns				6-20,21
	SIO Driving bus from REFRESH#							
t67a	SIO drives control and address from REFRESH# active	5		ns			8	6-21
	PCI and ISA Master Accesses to the Utility Bus							
	ECSADDR[2:0] and ECSEN#							
t68a	ECSADDR[2:0], ECSEN# valid from SA[19:0], LA[23:17] valid		16	ns				6-22
t68b	ECSADDR[2:0], ECSEN# invalid from SA[16:0], LA[23:17] invalid		25	ns				6-22
	UBUSTR and UBUSOE#							
t69a	UBUSTR active from IOR#, MEMR# active		17	ns				6-22
t69b	UBUSOE# active from IOx#, MEMx# active		29	ns				6-22
t69c	UBUSTR active setup to UBUSOE# active	3	12	ns				6-22
t69d	UBUSOE# inactive from IOx#, MEMx# inactive	50	73	ns			9	6-22
t69e	UBUSTR inactive from IOR#, MEMR# inactive	60	113	ns			9	6-22
t69f	UBUSOE# setup to UBUSTR inactive	10	40	ns			9	6-22
t69g	UBUSOE# inactive from SA[16:0] and LA[23:17]		17	ns			10	6-22
t69h	UBUSTR inactive from IOR#, MEMR# inactive	15	60	ns			10	6-22
	DMA Accesses To The Utility Bus							
	UBUSTR and UBUSOE#							
t70a	UBUSTR active from DACKx# active		25	ns			12,14	6-23
t70b	UBUSOE# active from DACK2# active		37	ns				6-23
t70c	UBUSTR setup to UBUSOE# active	3	12	ns			13	6-23
t70d	UBUSOE# inactive from DACK2# inactive		25	ns				6-23
t70e	UBUSTR inactive from DACKx# inactive	10	65	ns			12	6-23
t70f	UBUSOE# inactive setup to UBUSTR inactive	10	40	ns			13	6-23

Table 6-3. ISA and Utility Bus A.C. Characteristics (V_{DD} = 5V ±5%, T_{case} = 0 to 85 °C) (Continued)

Sym	Parameter	8.33 MHz		Units	Type	Size	Notes	Fig.
		Min	Max					
	Miscellaneous Timings							
	Mouse Timing Support							
t71a	IRQ 12/M minimum active pulse width (for mouse function)	180		ns				6-24
	Port 92h Accesses							
t72a	ALT_RST# active from IOW# active		480	ns				6-24
t72b	ALT_RST# active pulse width	420	540	ns				6-24
t72c	ALT_A20 driven active/inactive from IOW# active		420	ns				6-24
	Coprocessor Error Support							
t73a	IGNNE# active from IOW# active from port F0H access		220	ns				6-25
t73b	IGNNE# inactive from FERR# inactive		150	ns				6-25
	DSKCHG Timing							
t74a	DSKCHG valid to SD7 read data valid		25	ns				6-26
t74b	IOR# active to SD7 driven		25	ns				6-26
t74c	IOR# inactive to SD7 float		25	ns				6-26
	Real Time Clock Timing (RTCALE)							
t75a	RTCALE pulse width	200	300	ns				6-27
t75b	RTCALE active from IOW# active		70	ns				6-27
	Speaker Timing							
t76a	SPKR valid delay from OSC rising		200	ns				6-28

NOTES:
1. No-wait-state (ZEROWS#) asserted
2. This applies to the byte lane that the data has been swapped to.
3. Data is tri-stated from the standard memory commands (SMEMR# or SMEMW#), when they are generated.
4. IOCHRDY is driven active for a maximum of 70ns before floating. The 70ns includes both the drive time and the time it takes the SIO float IOCHRDY.
5. This applies to the last cycle of a demand mode DMA transfer.
6. Output from SIO.
7. Input to SIO.
8. This applies to ISA Master initiated refresh only.
9. SIO as a master cycles only.
10. ISA master cycles only.
11. This applies to the SIO cycles that IOCHRDY is not driven low.
12. This applies to all DACK# signals.
13. This applies to DACK2# only.
14. Utility Bus Read

6.3.2 ISA AND UTILITY BUS AC TEST LOADS

Table 6-5. ISA and Utility Bus AC Test Loads

Capacative Load	Pin
240pf	REFRESH#, EOP, SD[15:0], SA[19:0], SBHE#, LA[23:17], I0CS16#, MASTER#, MEMCS16#, MEMR#, MEMW#, SMEMR#, SMEMW#, IOR#, IOW#, AEN, BALE, IOCHRDY, ZEROWS#, RSTDRV, SYSCLK
120pf	DACK#[7:5,3:0]
50pf	SPKR, INT, NMI, ECSADDR[2:0], ECSEN#, UBUSTR, UBUSOE#, ALT__A20, ALT__RST#, IGNNE#

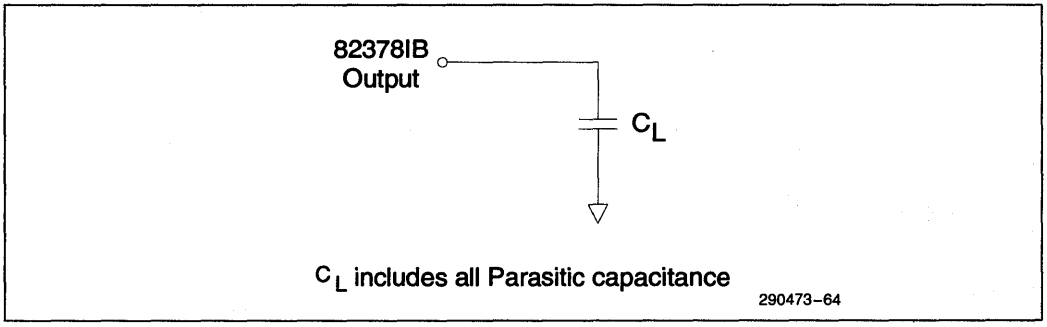


Figure 6-1. Test Load

6.3.3 ISA AND UTILITY BUS AC TIMING WAVE FORMS

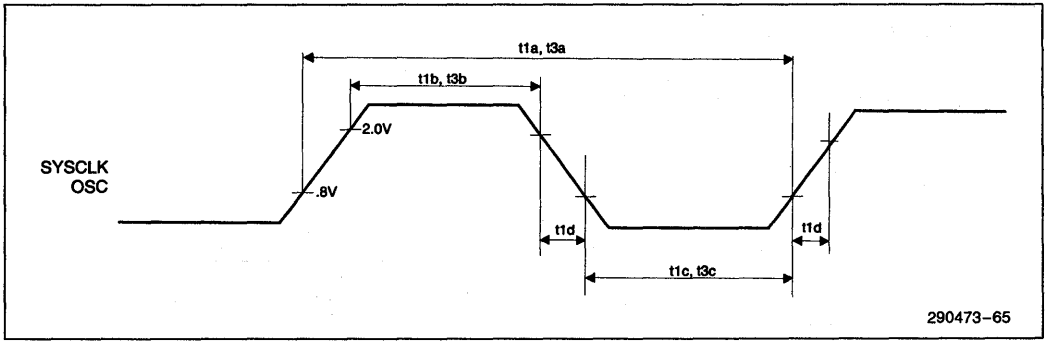


Figure 6-2. SYSCLK and OSC Timing

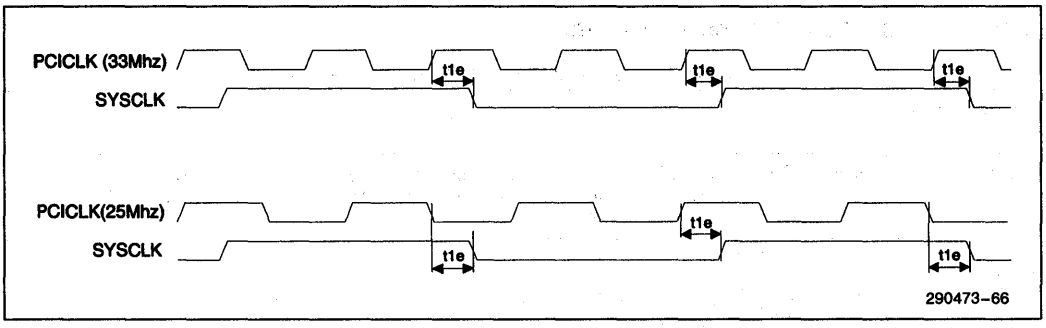


Figure 6-3. SYSCLK Valid Delay Timing

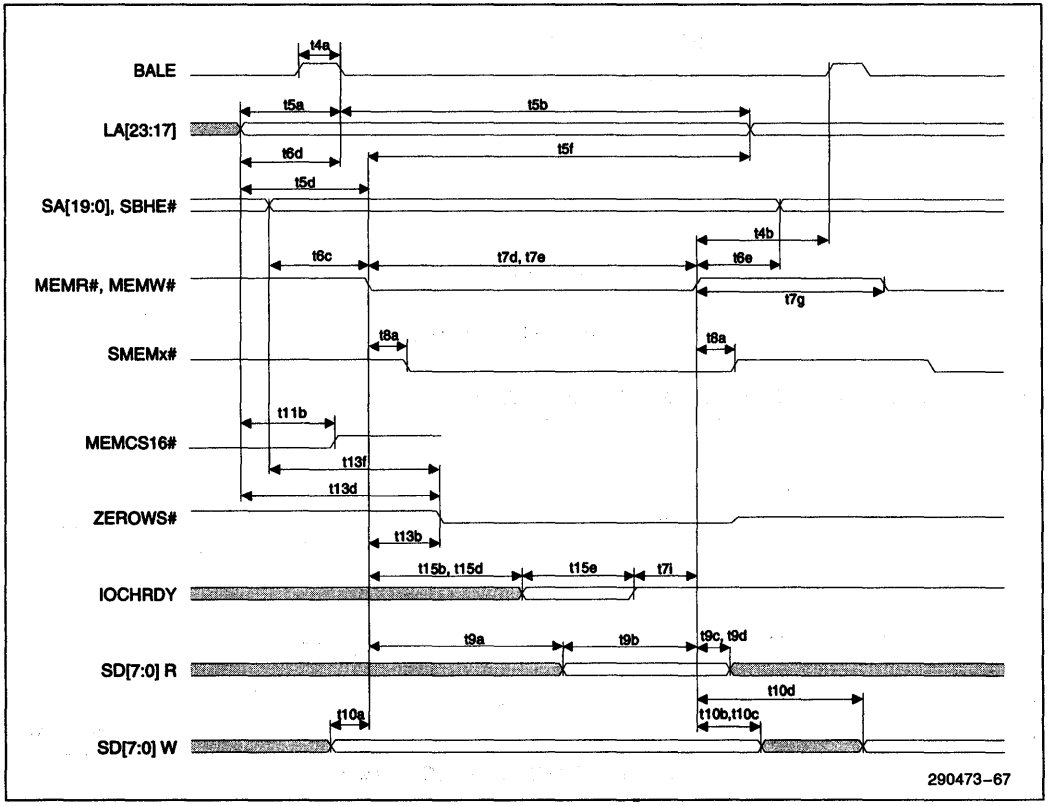


Figure 6-4. 8-Bit ISA Memory Slave Timing (SIO as Master)

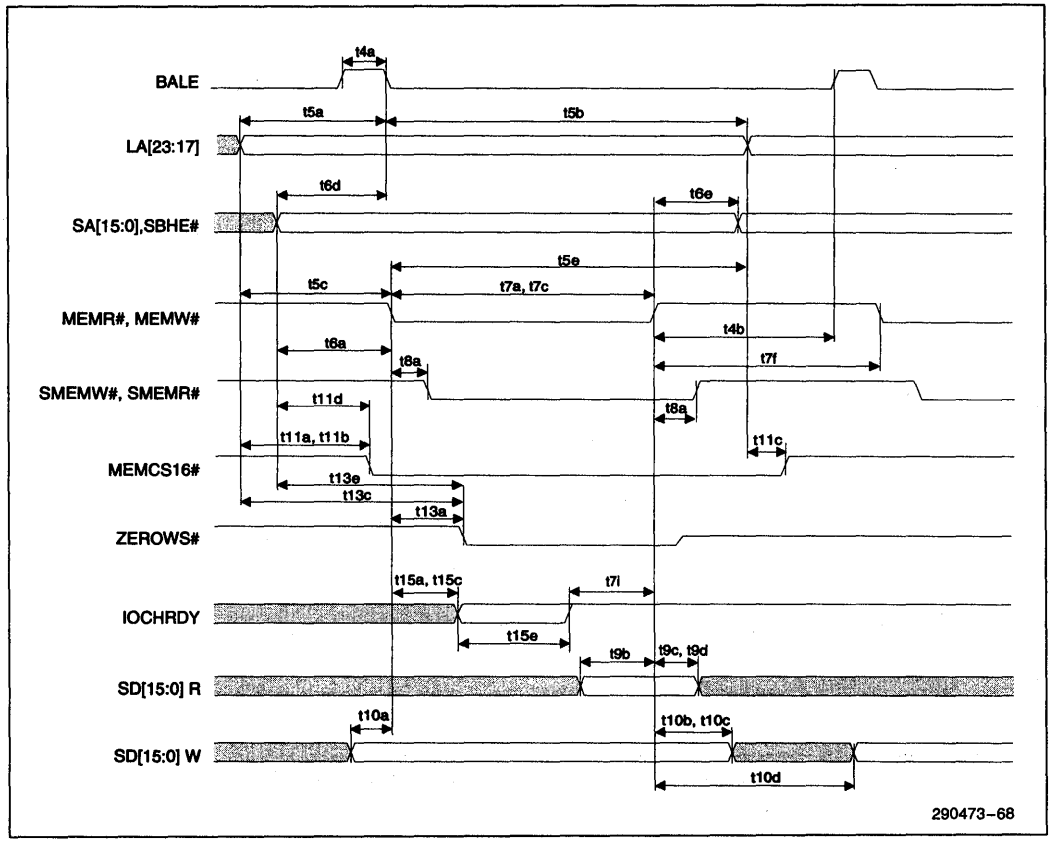


Figure 6-5. 16-Bit ISA Memory Slave Timing (SIO as Master)

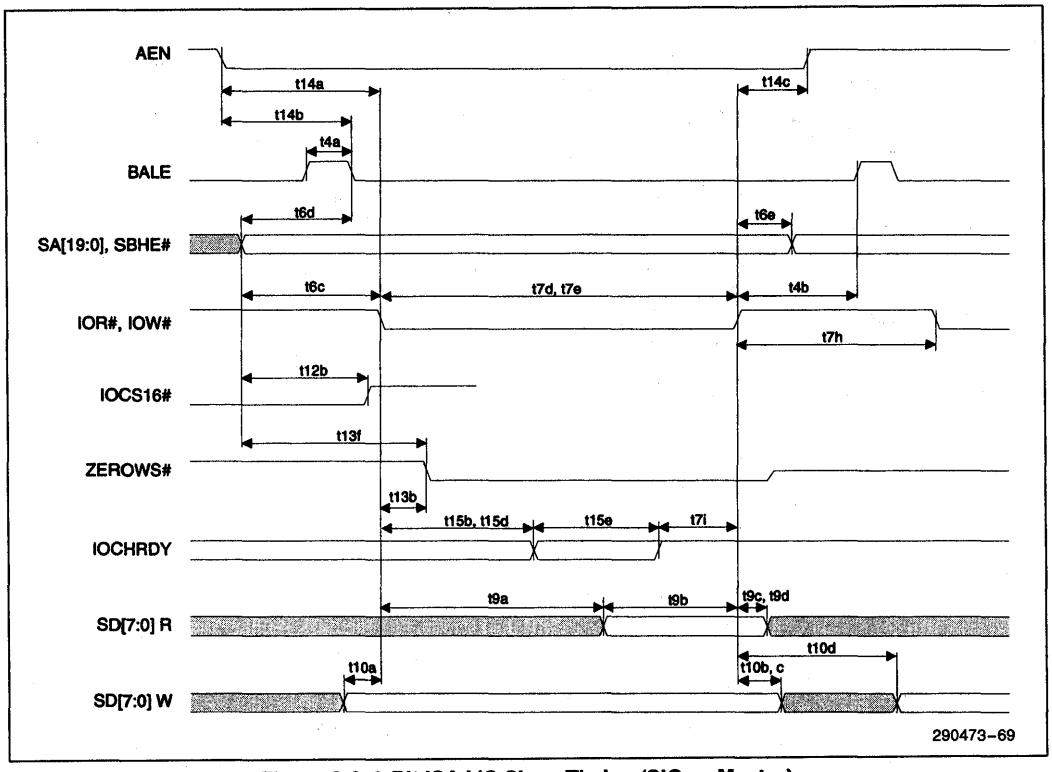


Figure 6-6. 8-Bit ISA I/O Slave Timing (SIO as Master)

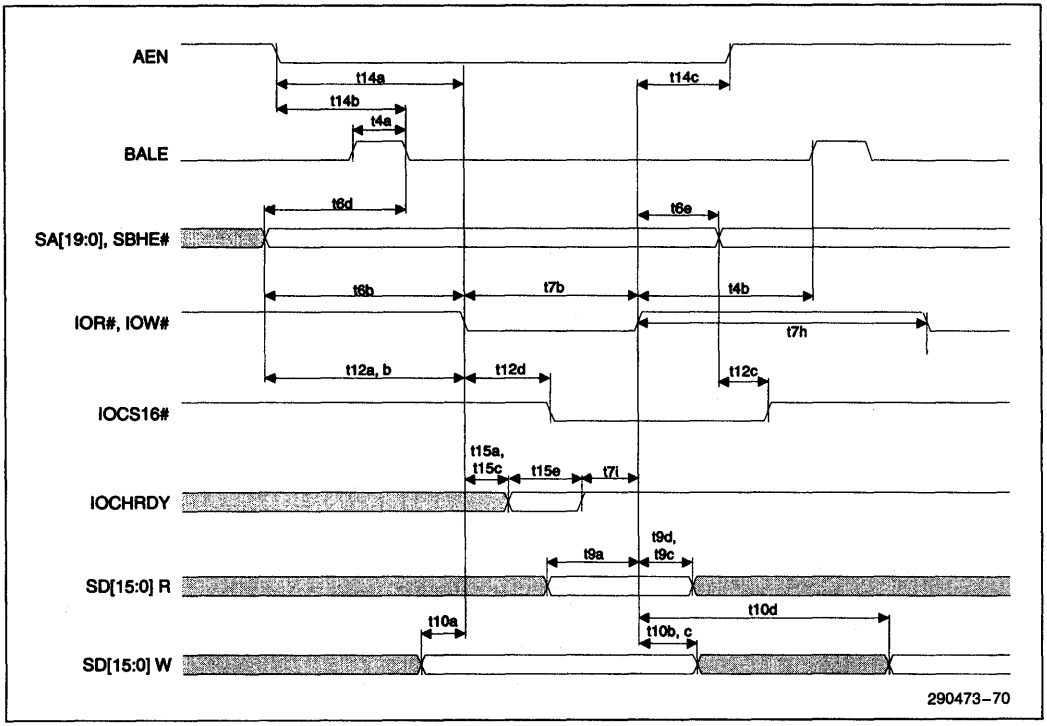


Figure 6-7. 16-Bit I/O Slave Timing (SIO as Master)

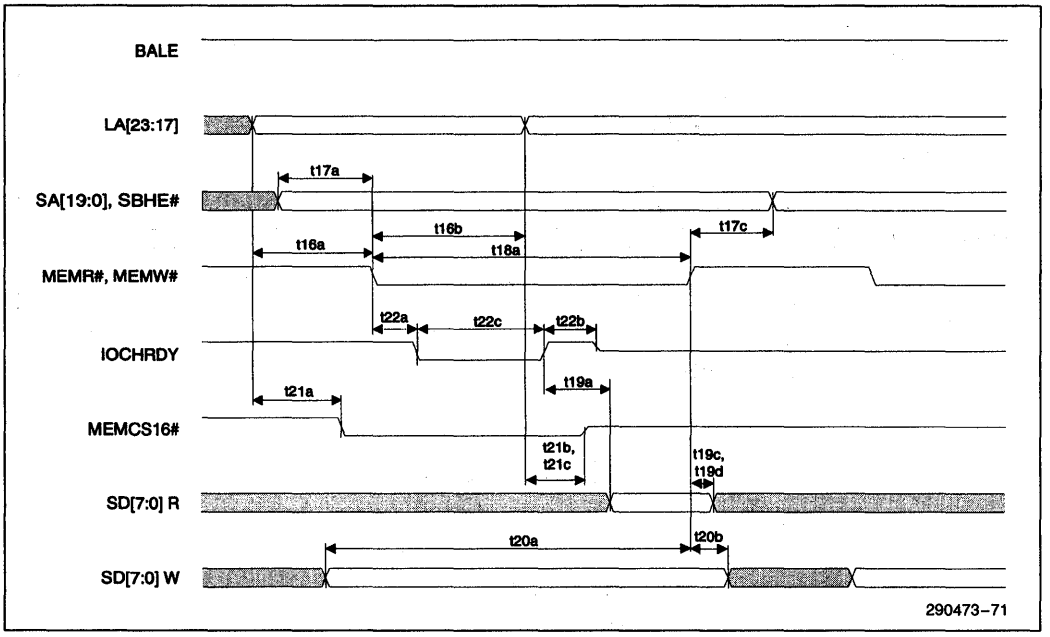


Figure 6-8. ISA Master Accessing PCI Memory Timing

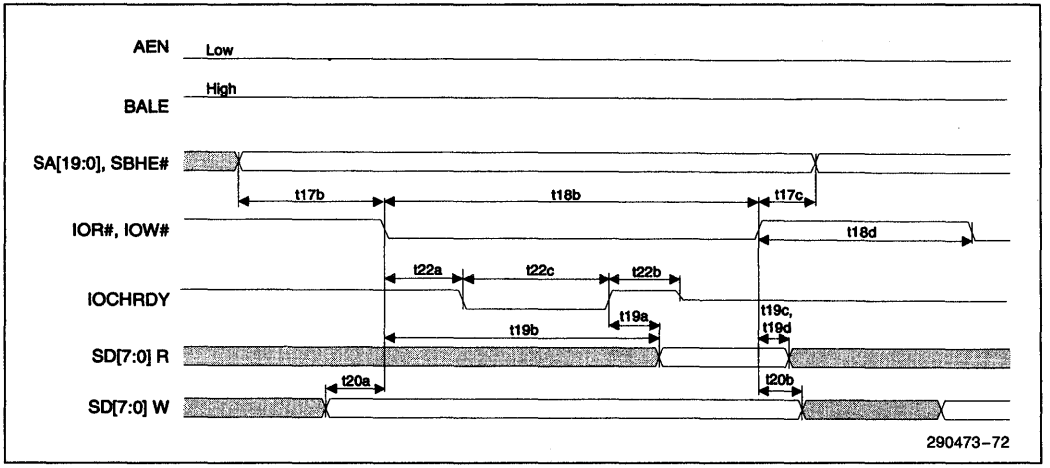


Figure 6-9. ISA Master Accessing SIO Register Timing

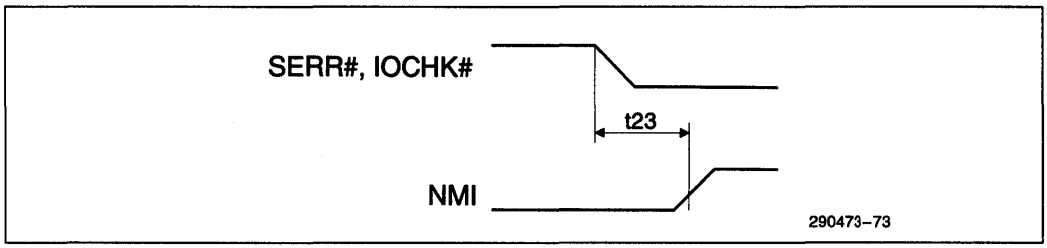


Figure 6-10. NMI Timing

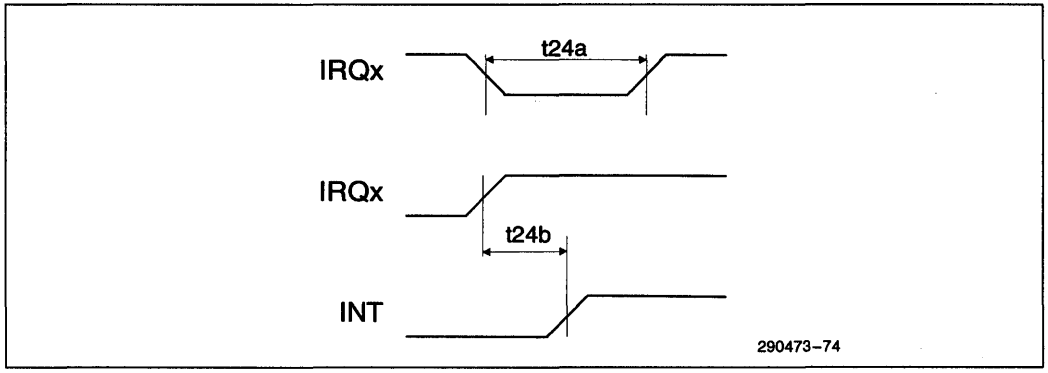


Figure 6-11. Interrupt Timing

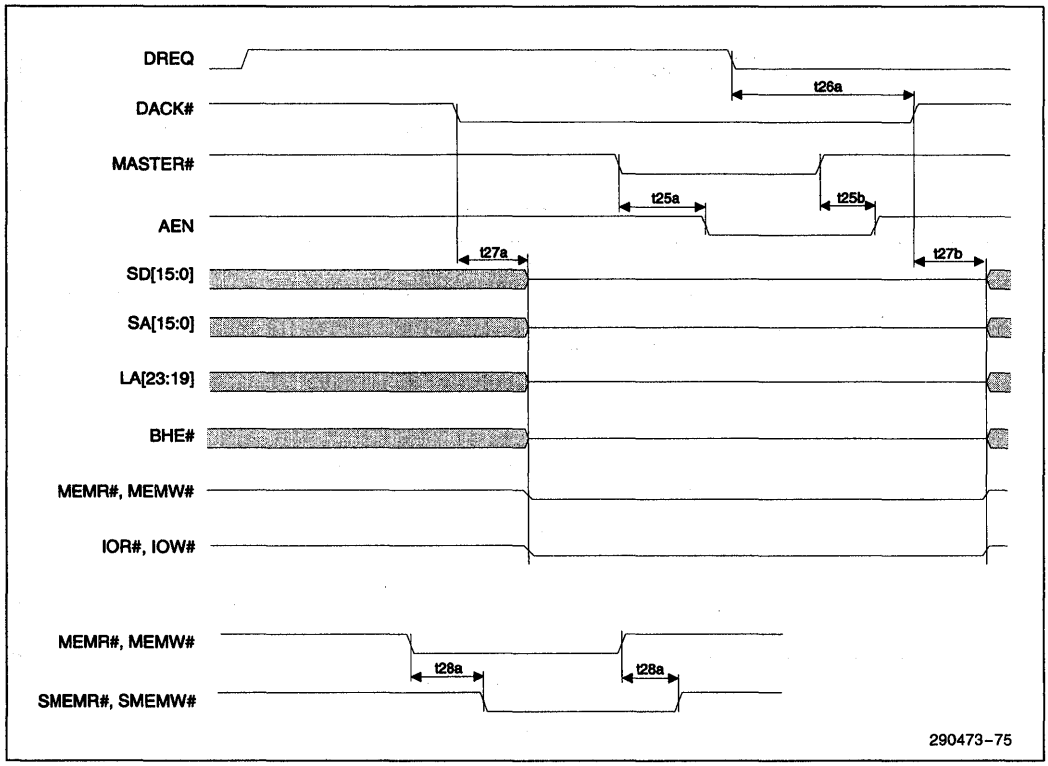


Figure 6-12. ISA Master Miscellaneous Timing

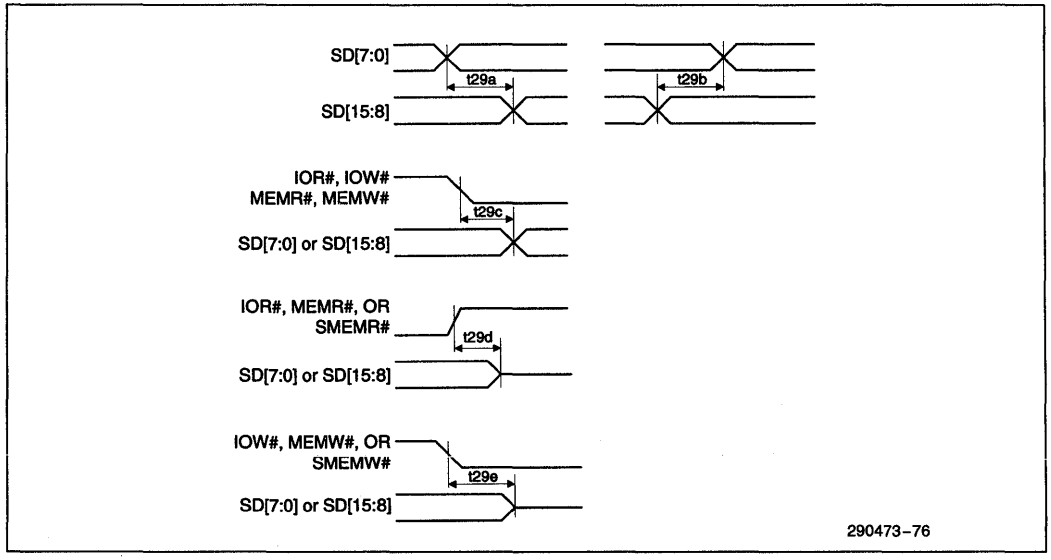


Figure 6-13. ISA Master Data Swap Timing

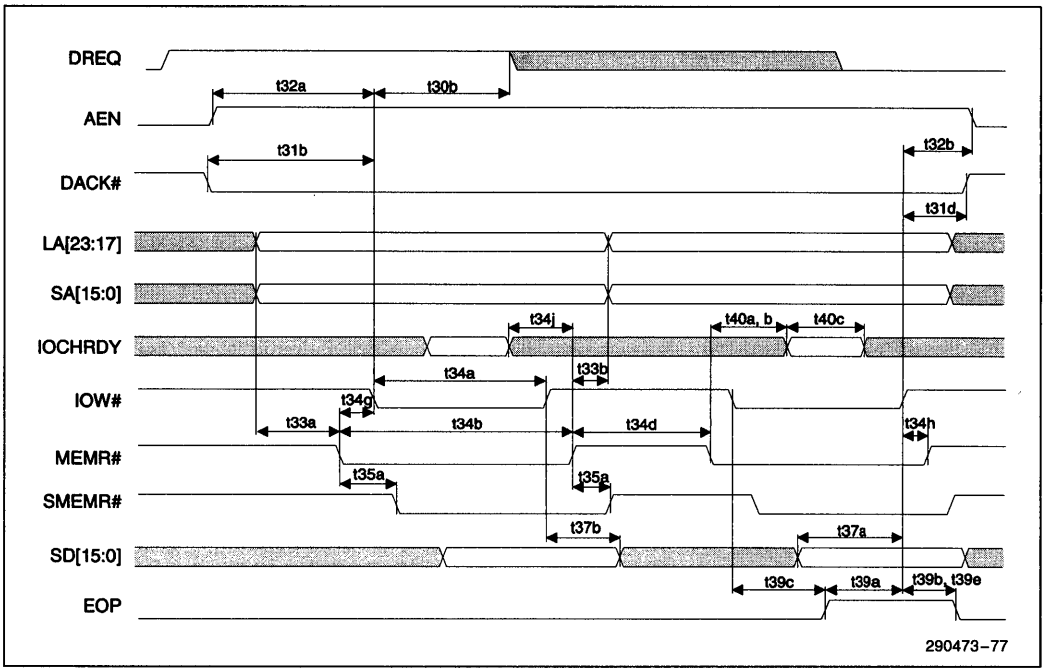


Figure 6-14. DMA Compatible Timing (Memory Read)

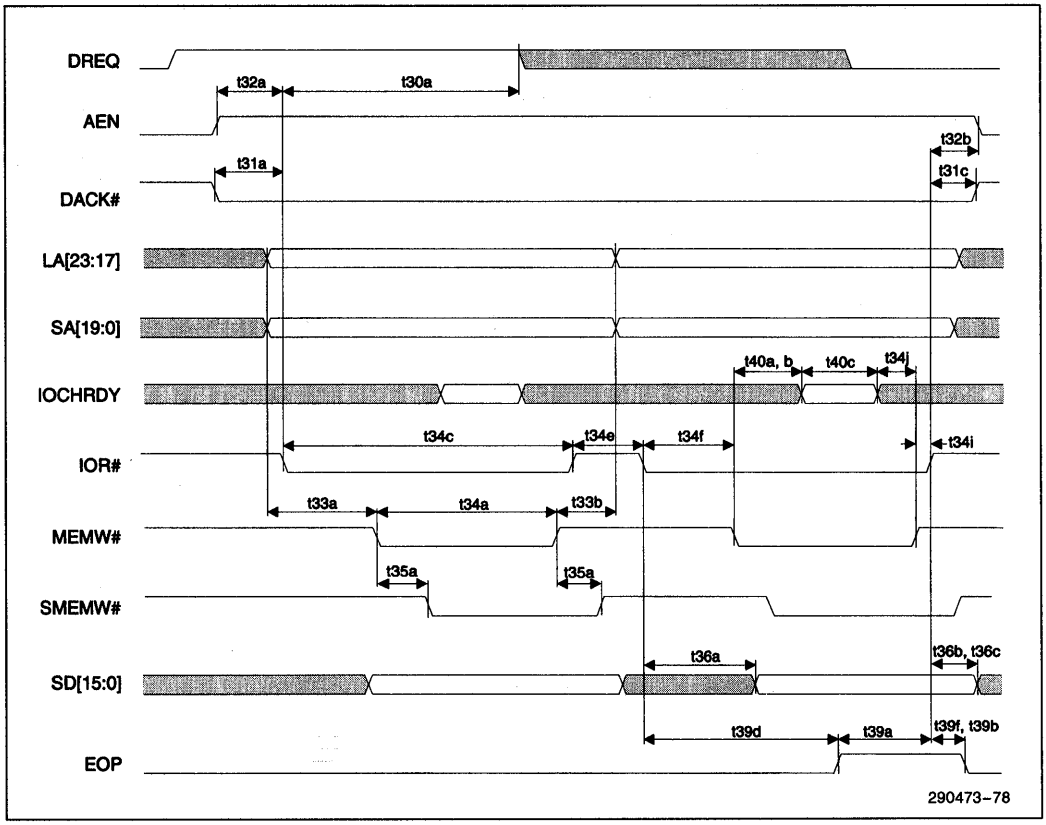


Figure 6-15. DMA Compatible Timing (Memory Write)

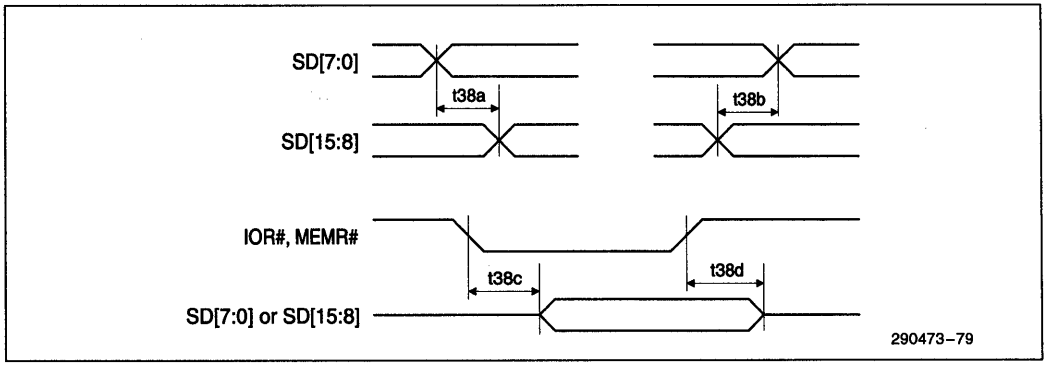


Figure 6-16. DMA Compatible Timing (Data Swap)

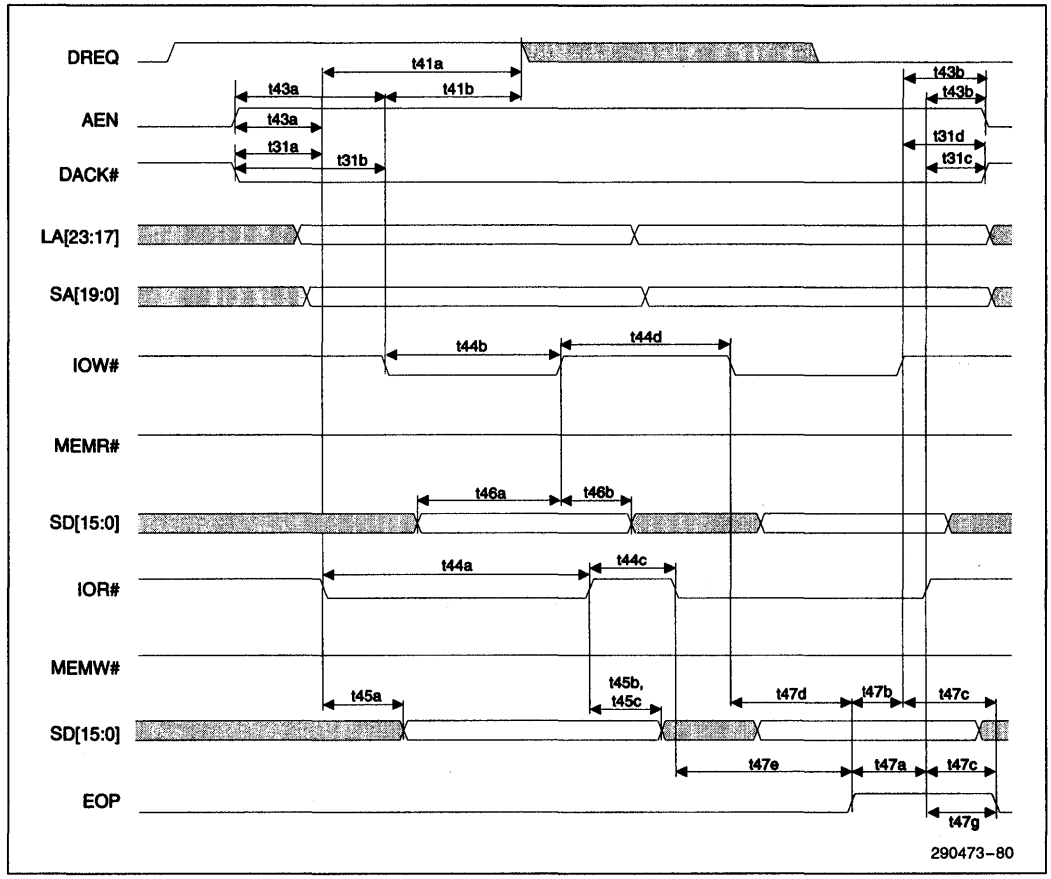


Figure 6-17. DMA Type "A" Timing

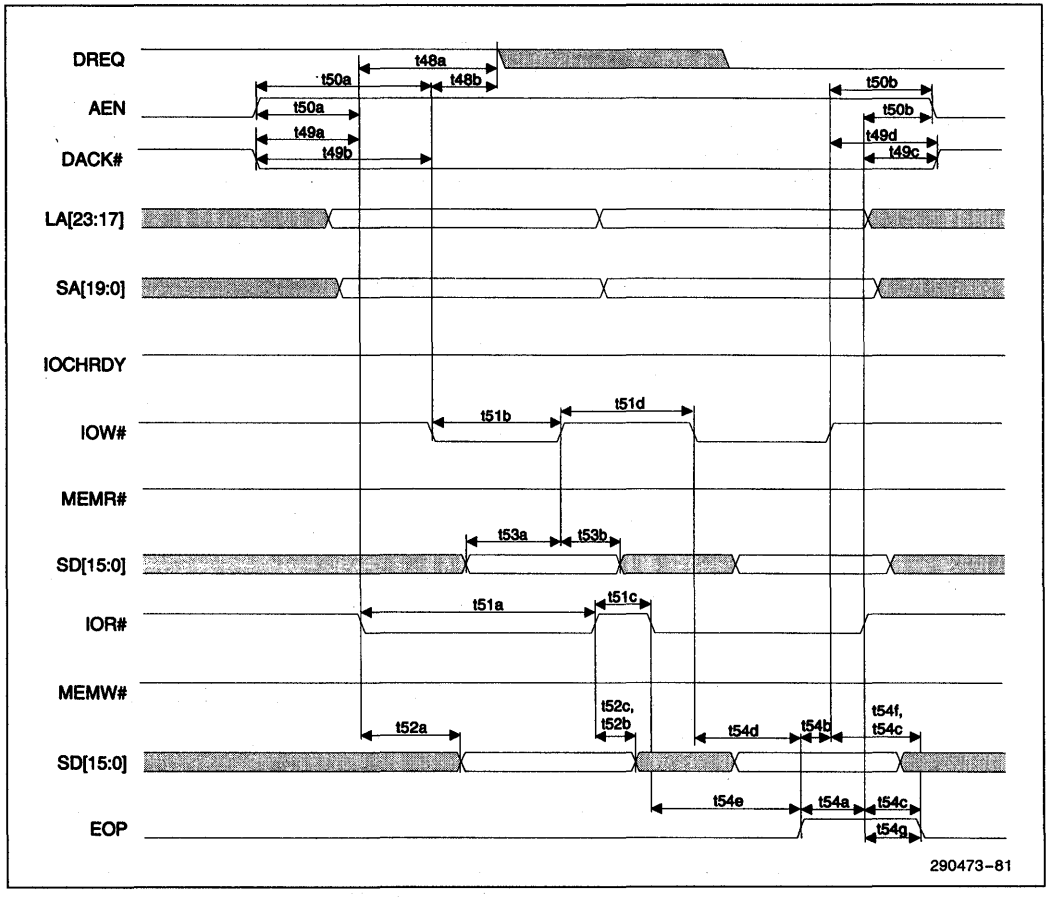


Figure 6-18. DMA Type "B" Timing

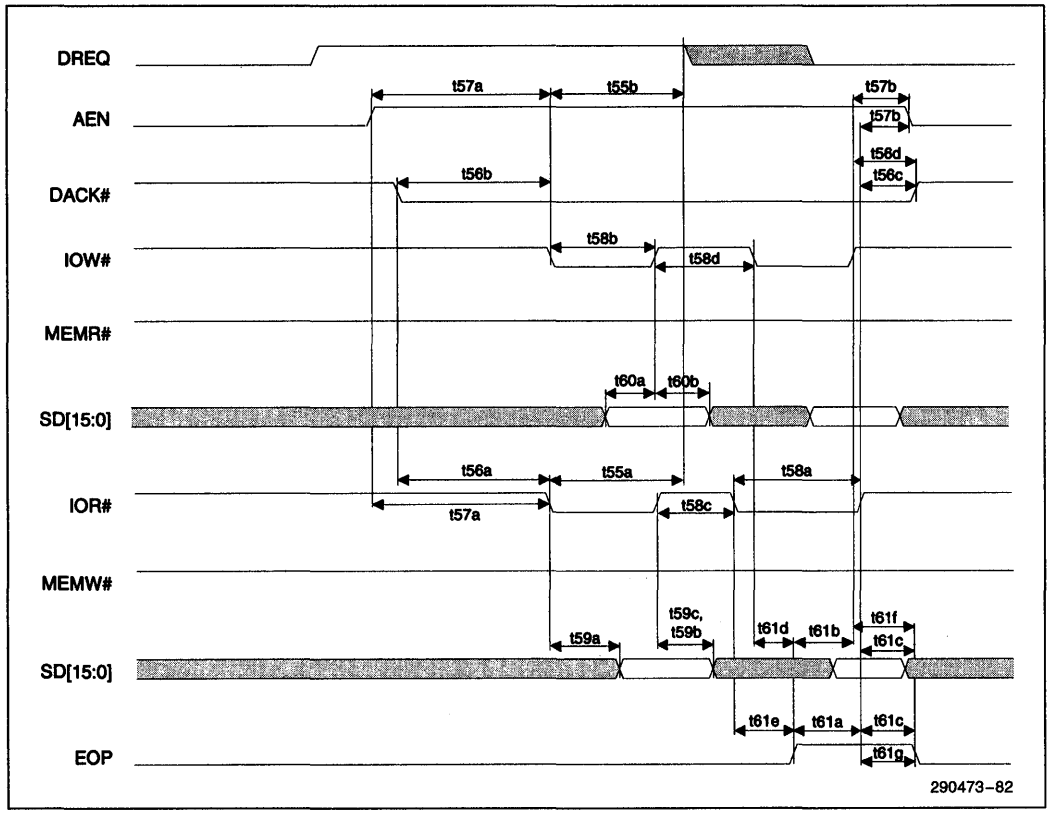


Figure 6-19. DMA Type "F" Timing

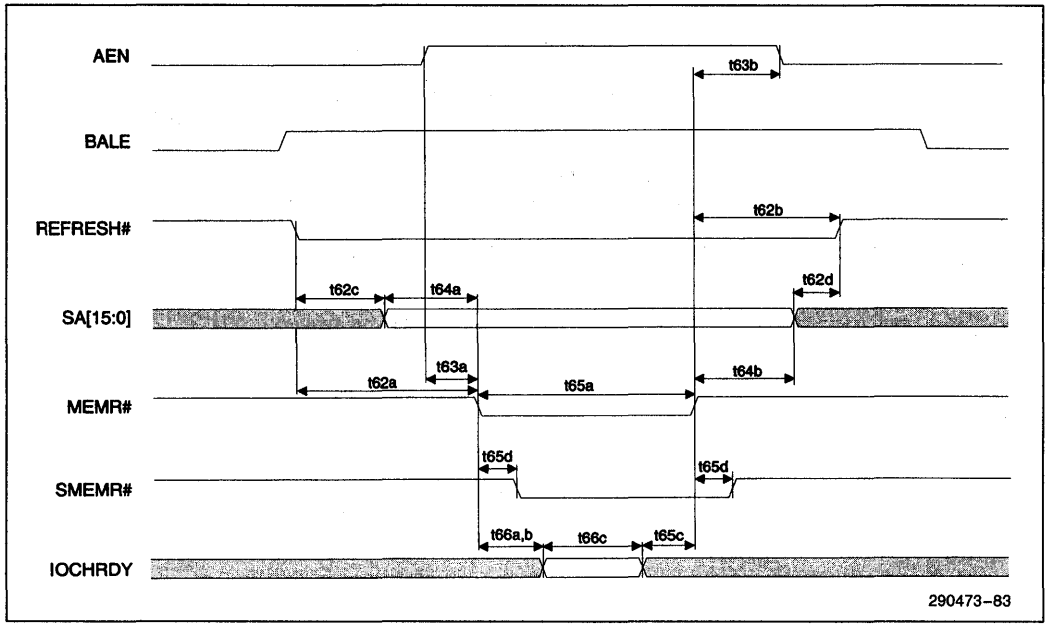


Figure 6-20. SIO-Initiated Refresh Timing

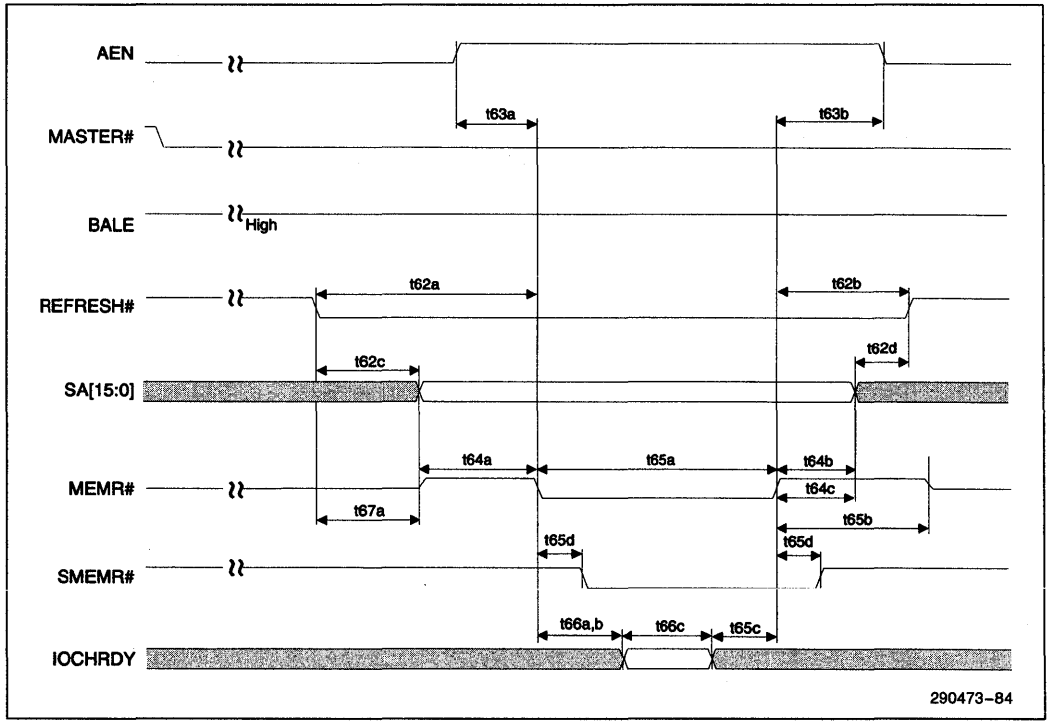


Figure 6-21. ISA Master-Initiated Refresh Timing

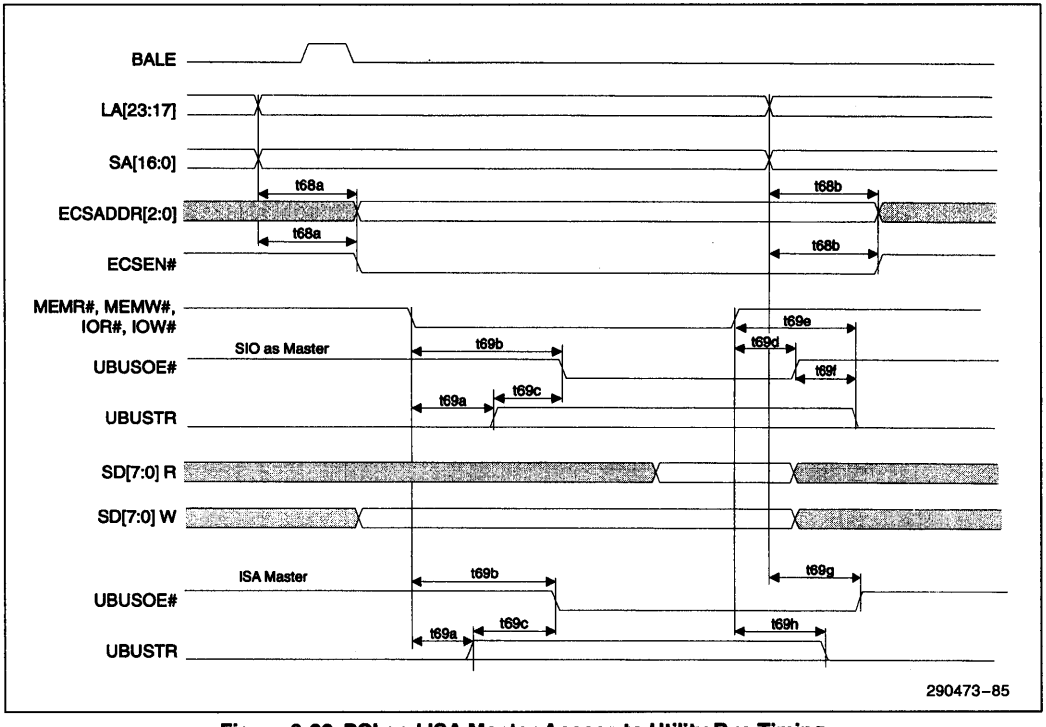


Figure 6-22. PCI and ISA Master Access to Utility Bus Timing

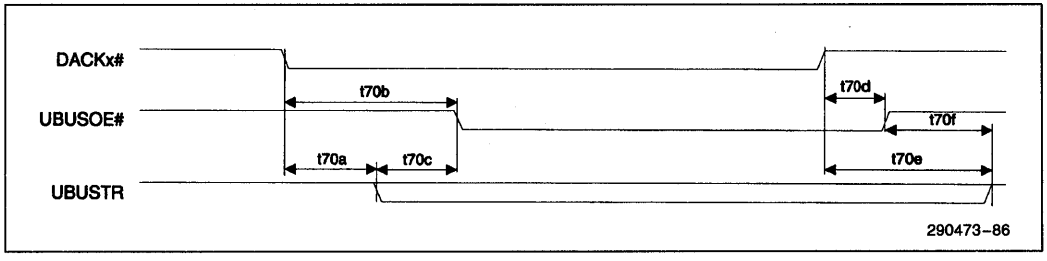


Figure 6-23. DMA Access to Utility Bus Timing

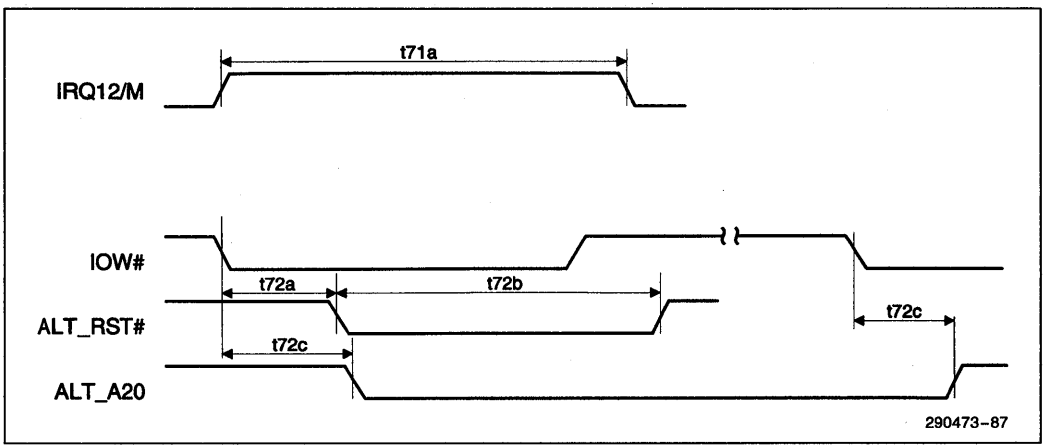


Figure 6-24. Mouse and Port 92 Timing

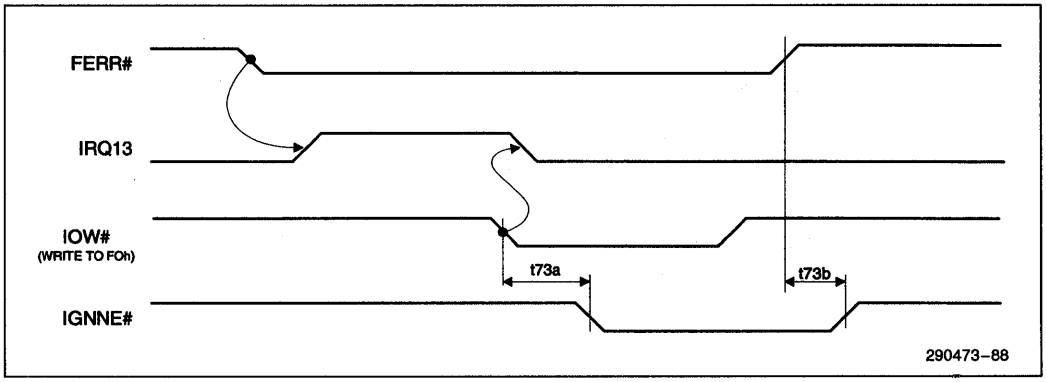


Figure 6-25. Coprocessor Error Support Timing

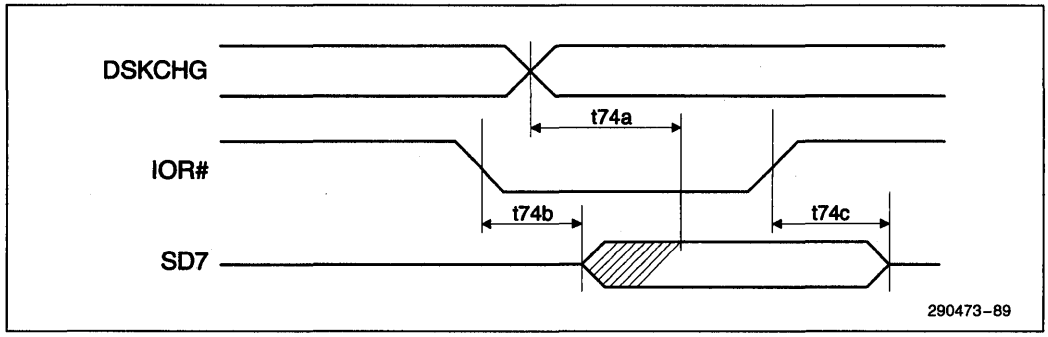


Figure 6-26. DSKCHG Timing

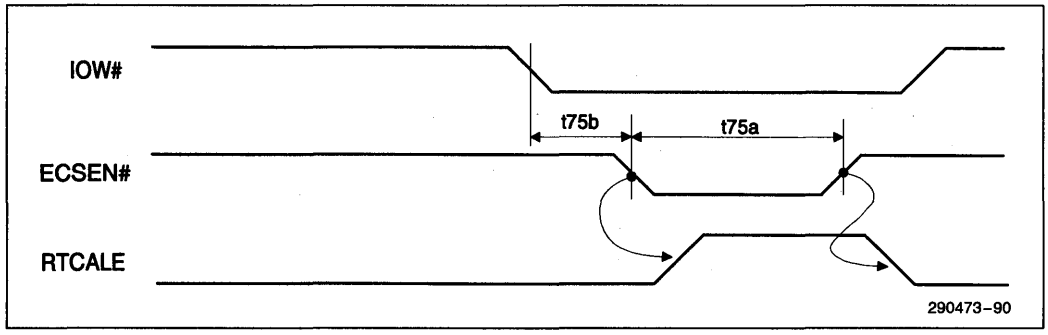


Figure 6-27. Real Time Clock Timing (RTCALE Generation)

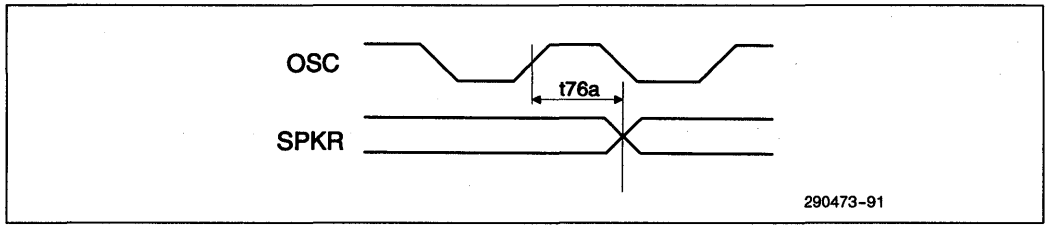


Figure 6-28. Speaker Timing

6.3.4 PCI BUS AC SPECIFICATIONS

Table 6-6. PCI System Signals A.C. Characteristics (V_{DD} = 5V ±5%, T_{case} = 0 to 85 °C)

Symbol	Parameter	Min	Typical	Max	Units	Notes	Figure
t77a	**PCICLK** Cycle Time	30		62.5	ns		
t77b	**PCICLK** High Time	40% * T_{CVC}			ns	at 2.0V	
t77c	**PCICLK** Low Time	40% * T_{CVC}			ns	at 0.8V	
t77d	**PCICLK** Rise Time			3	ns	0.8v to 2.0V	
t77e	**PCICLK** Fall Time			3	ns	2.0V to 0.8V	
t78	**PCIRST#** Pulse Width	1	100		ms	2	
t79	**PCICLK** active time at end of **PCIRST#**	100			μS		

NOTE:
1. System Signals are PCIRST#, PCICLK
2. PCICLK must be active for the last 100 μs (min) of any PCIRST# assertion.

Table 6-7. PCI Shared Signals A.C. Characteristics (V_{DD} = 5V ±5%, T_{case} = 0 to 85 °C)

Symbol	Parameter	Min	Max	Units	Notes
t80a	**PCICLK** to signal Valid Delay		11	ns	C_L = 50pF
t80b	**PCICLK** to Signal Invalid Delay	2		ns	
t80c	Hi-Z to Active delay from **PCICLK**	2		ns	
t80d	Active to Hi-Z delay from **PCICLK**		28	ns	Min = T_{VAL} min
t80e	Input Signal Valid Setup Time before **PCICLK**	7		ns	
t80f	Input Signal Hold Time from **PCICLK**	0		ns	

NOTE:
PCI Shared Signals are AD[31:0], C/BE[3:0]#, FRAME#, TRDY#, IRDY#, STOP#, LOCK#, IDSEL, DEVSEL#, PAR, SERR#.

Table 6-8. PCI Sideband Signals A.C. Characteristics (V_{DD} = 5V ±5%, T_{case} = 0 to 85 °C)

Symbol	Parameter	Min	Max	Units	Notes
t81a	**PCICLK** to Sideband Signal Valid Delay		12	ns	C_L = 50 pF
t81b	Sideband signal valid setup time before **PCICLK**	12		ns	2
t82	MEMACK# valid setup time before **PCICLK**	5		ns	
t83	SIOGNT# valid setup time before **PCICLK**	10		ns	

NOTES:
1. PCI Sideband Signals are MEMREQ#, MEMACK#, FLSHREQ, MEMCS#, GNT1#/RESUME#, GNT0#/SIOREQ#, CPUGNT#, REQ1#, REQ0#/SIOGNT#, CPUREQ#.
2. Excluding MEMACK#, SIOGNT#.
3. All other AC Timings are identical to the Shared Signal Timings.

6.3.5 PCI BUS A.C. TIMING WAVE FORMS

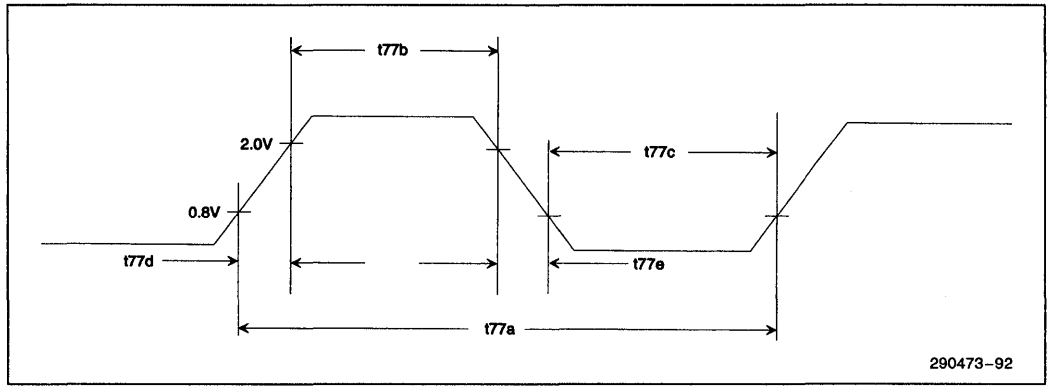


Figure 6-29. PCICLK A.C. Timing Wave Form

Since the PCIRST# signal is asynchronous, it can occur any time relative to PCICLK. However, it must be active for at least 100 μs while PCICLK is active as shown. If the component relies on synchronous PCIRST# behavior, it must synchronize PCIRST# internally.

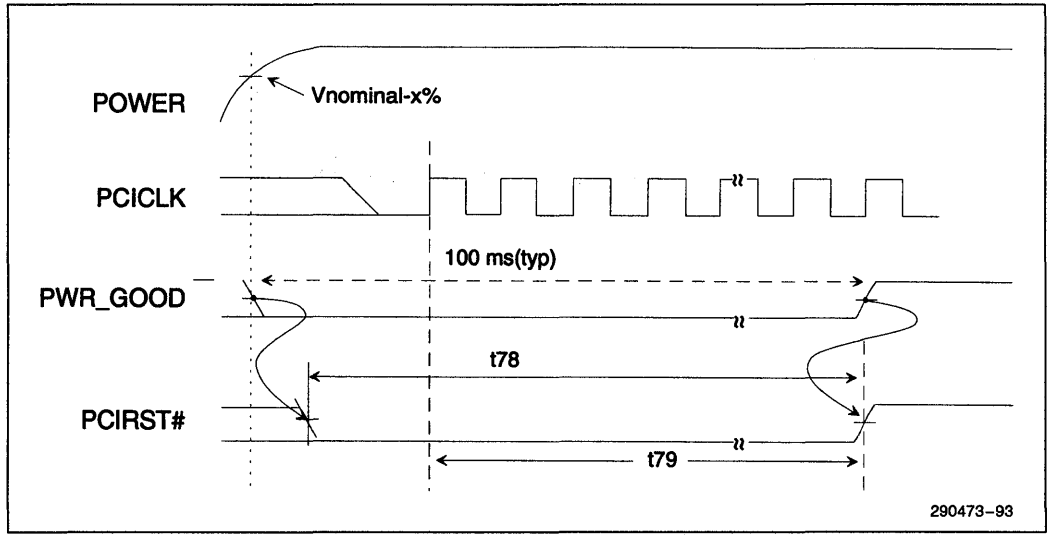


Figure 6-30. PCIRST# A.C. Timing Wave Form

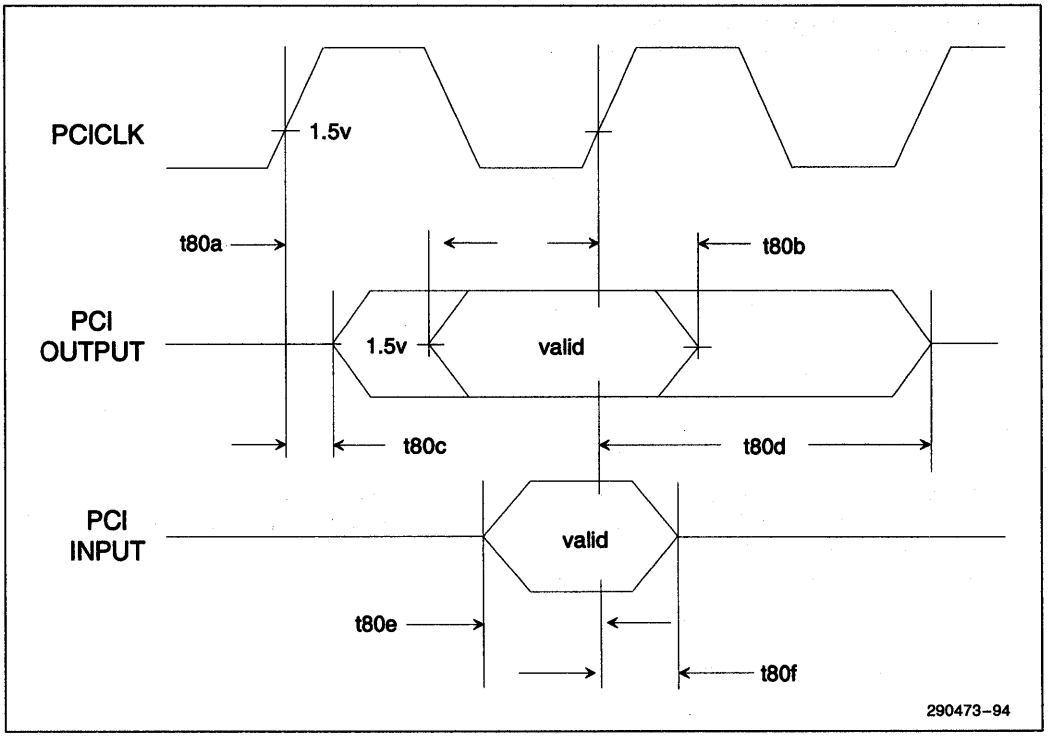


Figure 6-31. Shared Signals A.C. Timing Wave Form

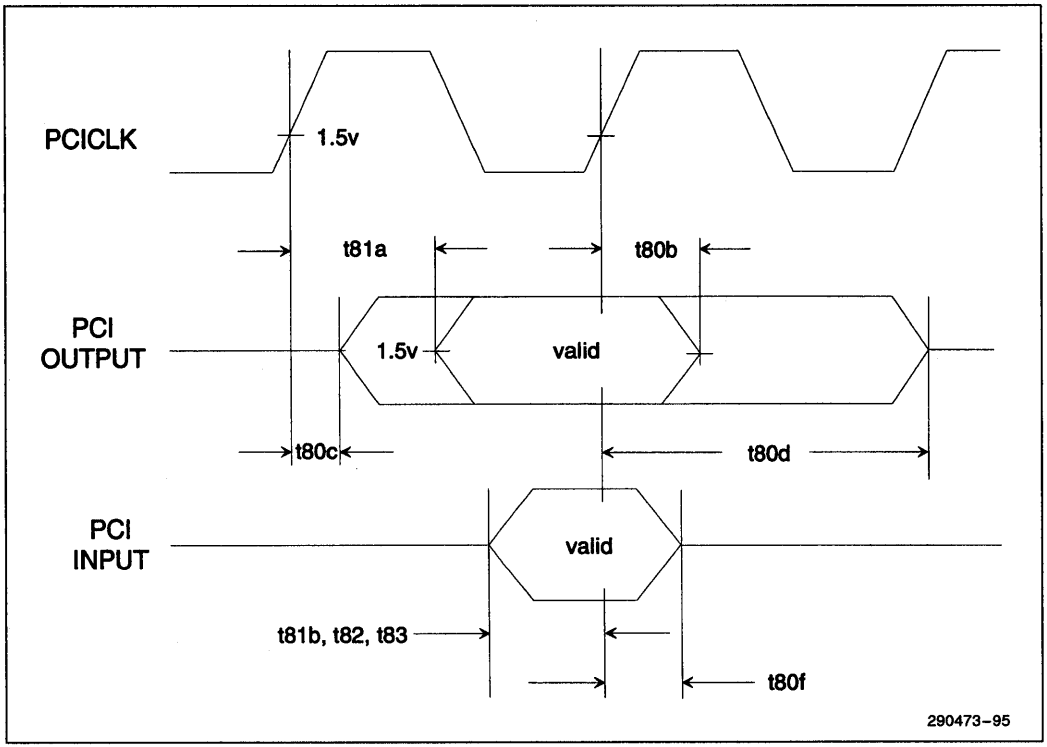


Figure 6-32. Sideband Signals A.C. Timing Wave Form

7.0 MECHANICAL DATA

7.1 Package Diagram

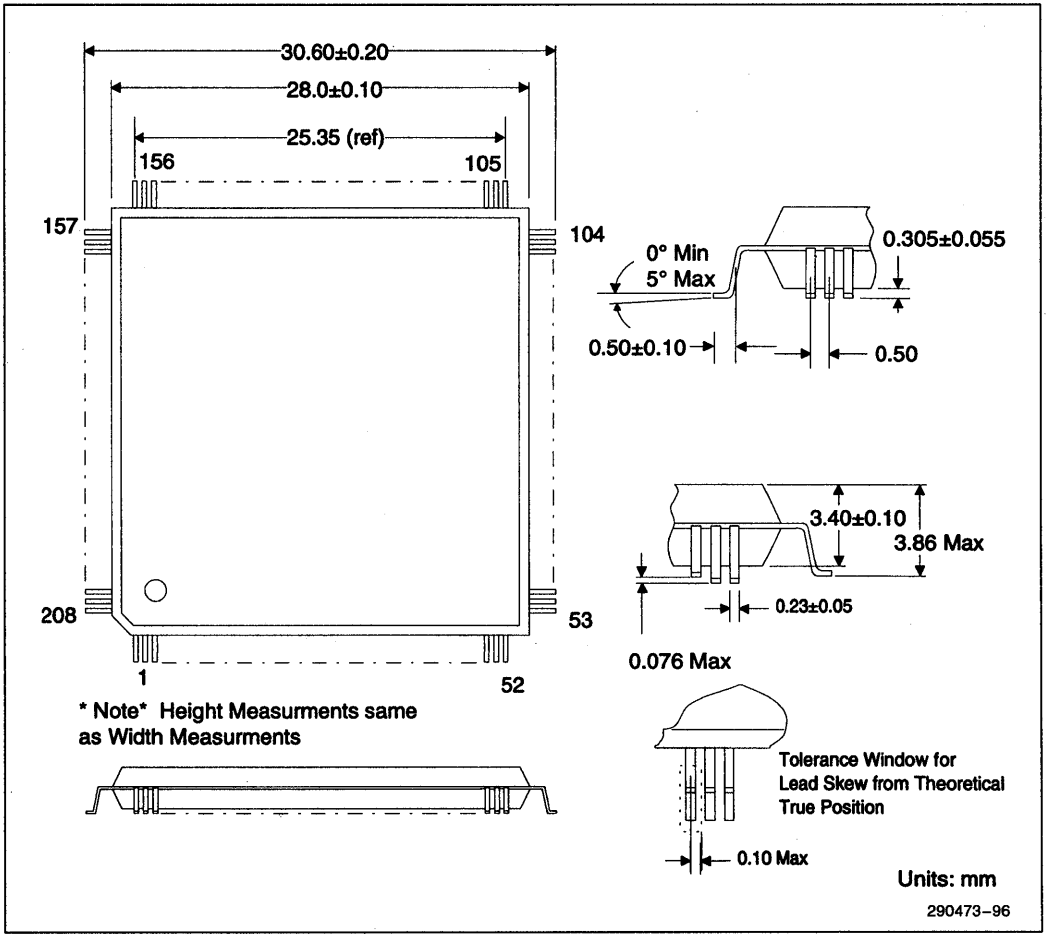


Figure 7-1. Packaging Dimension Information

7.2 Thermal Specifications

Table 7-1. 82378IB PQFP Package Thermal Characteristics

Thermal Resistance - °C/Watt			
	Air Flow Rate (Ft./Min)		
Parameter	**0**	**200**	**400**
θ Junction to Case	6.6	6.6	6.6
θ Case to Ambient	36.6	27.4	24

8.0 TESTABILITY

The TEST and TESTO pins are used to test the SIO. During normal operations, the TEST pin must be grounded. These test output TESTO may be left as a no-connect (NC).

8.1 Global Tri-State

The TEST pin and IRQ3 are used to provide a high-impedance tri-state test mode. When the following input combination occurs, all outputs and bi-directional pins are tri-stated, with the exception of TESTO:

TEST = '1'
IRQ3 = '1'

The SIO must be reset after the bi-directional and output pins have been tri-stated in this matter.

8.2 NAND Tree

A NAND Tree is provided primarily for VIL/VIH testing. The NAND Tree is also useful for **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 IRQ5 or IRQ6, activates the NAND Tree. All bi-directional pins, and certain pure output pins using bi-directional buffers for performance reasons, are tri-stated when the following input combinations occur:

TEST = '1'
IRQ5 = '1'

- or -

TEST = '1'
IRQ6 = '0'

The output pulse train is observed at the TESTO test output. IRQ4 must be held at logic '0' as the tree is toggled. IRQ4 may be pulsed at the end of the NAND test sequence. A pulse will appear at the TESTO output for an indeterminate period of time, thus verifying IRQ4 connectivity. Pure output pins are not included directly in the NAND Tree. As noted in Section 8.3, each output can be expected to toggle after the corresponding node noted next to the pin name toggles from a '1' to a '0'.

The sequence of the ATE test is as follows:

1. Drive TEST and IRQ5 high or TEST high and IRQ6 low.
2. Drive each input and bi-directional pin noted in Section 8.3 high.
3. Starting with the pin farthest from TESTO (SA8), individually drive each pin low. Except TESTO to toggle with each pin. Expect each pure output noted in Section 8.3 to toggle after each corresponding input pin has been driven low.
4. Turn off tester drivers before driving TEST low.
5. Reset the SIO prior to proceeding with further testing.

8.3 NAND Tree Cell Order

Table 8-1. NAND Tree Cell Order

Tree Output Number	Pin Number	Pin Name	Notes
	14	IRQ4	Process Monitor Control Pin, must be at logic '0' until the end of the test
	21	TESTO	Test Mode Output
1	11	IRQ5	Cell Closest to TESTO
2	10	SA9	
3	9	IRQ6	
4	8	SA10	
5	7	IRQ7	
6	6	SA11	
7	5	SA12	
8	4	REFRESH#	
9	3	SA13	
10	207	SA14	
1	206	MASTER#	
2	205	SA15	
3	204	MEMW#	
4	203	MEMR#	
15	202	SA16	
6	21	SA17	
7	200	IOR#	
8	199	SA18	
9	198	IOW#	
0	197	SA19	
1	196	SMEMR#	
2	193	AEN	
3	192	SMEMW#	
4	191	IOCHRDY	
5	190	SD0	
6	189	SD1	
7	188	ZEROWS#	
28	187	SD2	
29	186	SD3	
30	185	SD4	
31	184	IRQ9	
32	180	SD5	
33	179	SD6	

8.3 NAND Tree Cell Order (Continued)

Table 8-1. NAND Tree Cell Order (Continued)

Tree Output Number	Pin Number	Pin Name	Notes
34	178	SD7	
35	177	RSTDRV	
36	176	IOCHK#	
	175	ECSADDR0	NAND Tree Output of Tree Cell 37
	174	ECSADDR1	NAND Tree Output of Tree Cell 38
	173	ECSADDR2	NAND Tree Output of Tree Cell 41
37	172	IRQ8#	
	170	ECSEN#	NAND Tree Output of Tree Cell 42
	169	TEST	PI = > V_{CC}, TEST must be '1'
38	168	IRQ1	
39	166	SYSCLK	
	165	UBUSTR	NAND Tree Output of Tree Cell 43
	164	UBUSOE#	NAND Tree Output of Tree Cell 44
40	163	PCIRST#	
41	161	DSKCHG	
42	159	AD0	
43	155	AD1	
44	154	AD2	
45	153	AD3	
46	152	AD4	
47	151	AD5	
48	150	AD6	
49	149	AD7	
50	148	AD8	
51	147	C/BE0#	
52	146	AD9	
53	143	AD10	
54	142	AD11	
55	141	AD12	
56	140	AD13	
57	139	AD14	
58	138	AD15	
59	137	C/BE1#	
60	135	PAR	
61	134	SERR#	

8.3 NAND Tree Cell Order (Continued)

Table 8-1. NAND Tree Cell Order (Continued)

Tree Output Number	Pin Number	Pin Name	Notes
62	133	LOCK#	
63	132	STOP#	
64	128	DEVSEL#	
65	127	TRDY#	
66	126	IRDY#	
67	125	FRAME#	
68	124	C/BE2#	
69	123	AD16	
70	122	AD17	
71	121	AD18	
72	120	AD19	
73	119	AD20	
74	118	AD21	
75	115	AD22	
76	114	AD23	
77	113	C/BE3#	
78	112	AD24	
79	111	AD25	
80	110	AD26	
81	109	AD27	
82	108	AD28	
83	107	AD29	
84	106	AD30	
85	102	AD31	
86	101	IDSEL	
87	98	REQ1#	
88	96	CPUREQ#	
	95	CPUGNT#	NAND Tree Output of Tree Cell 89
	94	GNT1#	NAND Tree Output of Tree Cell 91
89	93	REQ0#	
	92	GNT0#	NAND Tree Output of Tree Cell 92
90	90	PCICLK	
	89	FLSHREQ#	NAND Tree Output of Tree Cell 94
91	88	MEMACK#	

8.3 NAND Tree Cell Order (Continued)

Table 8-1. NAND Tree Cell Order (Continued)

Tree Output Number	Pin Number	Pin Name	Notes
	87	MEMREQ#	NAND Tree Output of Tree Cell 95
	86	MEMCS#	NAND Tree Output of Tree Cell 96
	85	ALT_A20	NAND Tree Output of Tree Cell 97
92	80	OSC	
	76	ALT_RST#	NAND Tree Output of Tree Cell 98
	75	INT	NAND Tree Output of Tree Cell 99
	74	NMI	NAND Tree Output of Tree Cell 100
93	73	SPKR	
	72	IGNNE#	NAND Tree Output of Tree Cell 101
94	71	FERR#	
95	70	SD15	
96	69	SD14	
97	68	SD13	
98	67	SD12	
99	65	DREQ7	
100	64	SD11	
101	63	DACK7#	
102	62	SD10	
103	61	DREQ6	
104	60	SD9	
105	59	DACK6#	
106	58	DREQ3	
107	57	DREQ2	
108	56	DREQ1	
109	55	SD8	
110	51	DREQ5	
111	50	DACK5#	
112	49	DACK3#	
113	48	DACK1#	
114	47	DREQ0	
115	46	LA17	
116	45	DACK0#	
117	44	LA18	
118	43	IRQ14	
119	42	LA19	
120	41	IRQ15	
121	40	LA20	

8.3 NAND Tree Cell Order (Continued)

Table 8-1. NAND Tree Cell Order (Continued)

Tree Output Number	Pin Number	Pin Name	Notes
122	39	IRQ12/M	
123	38	LA21	
124	37	IRQ11	
125	36	LA22	
126	35	IRQ10	
127	34	LA23	
128	33	IOCS16#	
129	32	SBHE#	
130	31	MEMCS16#	
131	30	SA0	
132	29	SA1	
133	28	SA2	
134	24	SA3	
135	23	BALE	
136	22	SA4	
137	20	EOP	
138	19	SA5	
139	18	DACK2#	
140	17	SA6	
141	16	IRQ3	Output signals will transition from high-impedance state to driving state after this pin is driven low.
142	15	SA7	
143	13	SA8	Cell Furthest from TESTO Start of NAND Tree

8.4 NAND Tree Diagram

Figure 8-1 shows the NAND Tree Diagram. Note that the Pulse Generator shown is only used for testing the IRQ4 signal. It is enabled when IRQ4 is pulsed high.

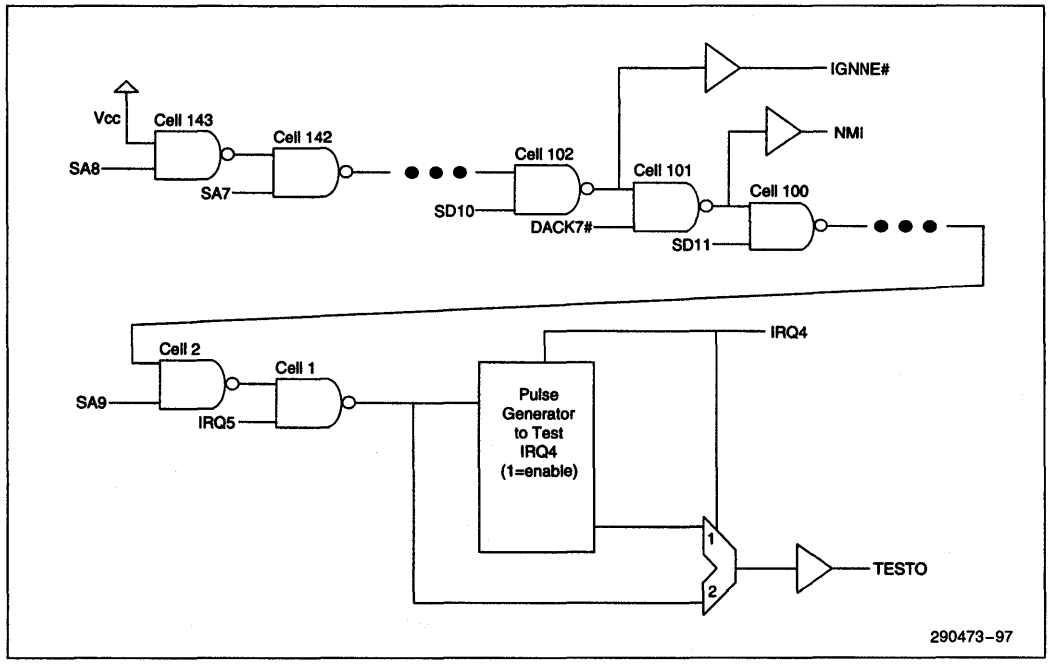


Figure 8-1. NAND Tree Diagram

ntel®

NORTH AMERICAN SALES OFFICES

.BAMA

l Corp.
Boulevard South
e 104-I
tsville 35802
(800) 628-8686
: (205) 883-3511

ZONA

el Corp.
North 44th Stree
e 500
enix 85008
(800) 628-8686
: (602) 244-0446

.IFORNIA

l Corp.
erra Gate Plaza
e 280C
eville 95678
(800) 628-8686
: (916) 782-8153

el Corp.
5 Chesapeake Dr.
e 325
Diego 92123
(800) 628-8686
: (619) 292-0628

l Corp.
1 Fox Drive
Jose 95131
(800) 628-8686
: (408) 441-9540

ntel Corp.
N. Tustin Avenue
te 450
nta Ana 92705
(800) 628-8686
X: 910-595-1114
: (714) 541-9157

el Corp.
60 Ventura Boulevard
te 360
rman Oaks 91403
(800) 628-8686
: (818) 995-6624

LORADO

ntel Corp.
S. Cherry St.
te 700
nver 80222
: (800) 628-8686
X: 910-931-2289
: (303) 322-8670

NNECTICUT

tel Corp.
Mill Plain Road
nbury 06811
: (800) 628-8686
: (203) 794-0339

ORIDA

tel Corp.
Fairway Drive
te 160
erfield Beach 33441
: (800) 628-8686
: (305) 421-2444

el Corp.
50 Lucien Way
te 100-8, 27
itland 32751
: (800) 628-8686
X: (407) 660-1283

GEORGIA

†Intel Corp.
20 Technology Parkway
Suite 150
Norcross 30092
Tel: (800) 628-8686
FAX: (404) 605-9762

ILLINOIS

*†Intel Corp.
Woodfield Corp. Center III
300 N. Martingale Road
Suite 400
Schaumburg 60173
Tel: (800) 628-8686
FAX: (708) 706-9762

INDIANA

†Intel Corp.
8910 Purdue Road
Suite 350
Indianapolis 46268
Tel: (800) 628-8686
FAX: (317) 875-8938

MARYLAND

*†Intel Corp.
10010 Junction Dr.
Suite 200
Annapolis Junction 20701
Tel: (800) 628-8686
FAX: (410) 206-3678

MASSACHUSETTS

*†Intel Corp.
Westford Corp. Center
5 Carlisle Road
2nd Floor
Westford 01886
Tel: (800) 628-8686
TWX: 710-343-6333
FAX: (508) 692-7867

MICHIGAN

†Intel Corp.
7071 Orchard Lake Road
Suite 100
West Bloomfield 48322
Tel: (800) 628-8686
FAX: (313) 851-8770

MINNESOTA

†Intel Corp.
3500 W. 80th St.
Suite 360
Bloomington 55431
Tel: (800) 628-8686
TWX: 910-576-2867
FAX: (612) 831-6497

NEW JERSEY

*†Intel Corp.
Lincroft Office Center
125 Half Mile Road
Red Bank 07701
Tel: (800) 628-8686
FAX: (908) 747-0983

NEW YORK

*Intel Corp.
850 Crosskeys Office Park
Fairport 14450
Tel: (800) 628-8686
TWX: 510-253-7391
FAX: (716) 223-2561

†Intel Corp.
300 Westage Business Center
Suite 230
Fishkill 12524
Tel: (800) 628-8686
FAX: (914) 897-3125

*†Intel Corp.
2950 Express Dr., South
Suite 130
Islandia 11722
Tel: (800) 628-8686
TWX: 510-227-6236
FAX: (516) 348-7939

OHIO

*Intel Corp.
Four Commerce Park Square
23200 Chagrin Blvd., Suite 600
Beachwood 44122
Tel: (800) 628-8686
FAX: (216) 464-2270

*†Intel Corp.
3401 Park Center Drive
Suite 220
Dayton 45414
Tel: (800) 628-8686
TWX: 810-450-2528
FAX: (513) 890-8658

OKLAHOMA

Intel Corp.
6801 N. Broadway
Suite 115
Oklahoma City 73162
Tel: (800) 628-8686
FAX: (405) 840-9819

OREGON

†Intel Corp.
15254 N.W. Greenbrier Pkwy.
Building B
Beaverton 97006
Tel: (800) 628-8686
TWX: 910-467-8741
FAX: (503) 645-8181

PENNSYLVANIA

*†Intel Corp.
925 Harvest Drive
Suite 200
Blue Bell 19422
Tel: (800) 628-8686
FAX: (215) 641-0785

SOUTH CAROLINA

Intel Corp.
7403 Parklane Rd., Suite 3
Columbia 29223
Tel: (800) 628-8686
FAX: (803) 788-7999

Intel Corp.
100 Executive Center Drive
Suite 109, B183
Greenville 29615
Tel: (800) 628-8686
FAX: (803) 297-3401

TEXAS

†Intel Corp.
8911 N. Capital of Texas Hwy.
Suite 4230
Austin 78759
Tel: (800) 628-8686
FAX: (512) 338-9335

*†Intel Corp.
5000 Quorum Drive
Suite 750
Dallas 75240
Tel: (800) 628-8686

*†Intel Corp.
20515 SH 249
Suite 401
Houston 77070
Tel: (800) 628-8686
TWX: 910-881-2490
FAX: (713) 988-3660

UTAH

†Intel Corp.
428 East 6400 South
Suite 135
Murray 84107
Tel: (800) 628-8686
FAX: (801) 268-1457

WASHINGTON

†Intel Corp.
2800 156th Avenue S.E.
Suite 105
Bellevue 98007
Tel: (800) 628-8686
FAX: (206) 746-4495

Intel Corp.
408 N. Mullan Road
Suite 105
Spokane 99206
Tel: (800) 628-8686
FAX: (509) 928-9467

WISCONSIN

Intel Corp.
400 N. Executive Dr.
Suite 401
Brookfield 53005
Tel: (800) 628-8686
FAX: (414) 789-2746

CANADA

BRITISH COLUMBIA

Intel Semiconductor of
Canada, Ltd.
999 Canada Place
Suite 404, #11
Vancouver V6C 3E2
Tel: (800) 628-8686
FAX: (604) 844-2813

ONTARIO

†Intel Semiconductor of
Canada, Ltd.
2650 Queensview Drive
Suite 250
Ottawa K2B 8H6
Tel: (800) 628-8686
FAX: (613) 820-5936

†Intel Semiconductor of
Canada, Ltd.
190 Attwell Drive
Suite 500
Rexdale M9W 6H8
Tel: (800) 628-8686
FAX: (416) 675-2438

QUEBEC

†Intel Semiconductor of
Canada, Ltd.
1 Rue Holiday
Suite 115
Tour East
Pt. Claire H9R 5N3
Tel: (800) 628-8686
FAX: 514-694-0064

ales and Service Office
eld Application Location

NORTH AMERICAN DISTRIBUTORS

ALABAMA

Arrow/Schweber Electronics
1015 Henderson Road
Huntsville 35816
Tel: (205) 837-6955
FAX: (205) 895-0126

Hamilton/Avnet
4960 Corporate Drive, #135
Huntsville 35805
Tel: (205) 837-7210
FAX: (205) 830-8404

MTI Systems Sales
4950 Corporate Dr., #120
Huntsville 35805
Tel: (205) 830-9526
FAX: (205) 830-9557

Pioneer Technologies Group
4835 University Square, #5
Huntsville 35816
Tel: (205) 837-9300
FAX: (205) 837-9358

ARIZONA

Arrow/Schweber Electronics
2415 W. Erie Drive
Tempe 85282
Tel: (602) 431-0030
FAX: (602) 431-9555

Avnet Computer
1626 S. Edwards Drive
Tempe 85281
Tel: (602) 902-4642
FAX: (602) 902-4646

Hamilton/Avnet
1626 S. Edwards Drive
Tempe 85281
Tel: (602) 902-4700
FAX: (602) 902-4747

Wyle Laboratories
4141 E. Raymond
Phoenix 85040
Tel: (602) 437-2088
FAX: (602) 437-2124

CALIFORNIA

Arrow Commercial Systems Group
1502 Crocker Avenue
Hayward 94544
Tel: (510) 489-5371
FAX: (510) 391-1742

Arrow Commercial Systems Group
14242 Chambers Road
Tustin 92680
Tel: (714) 544-0200
FAX: (714) 454-4203

Arrow/Schweber Electronics
26707 W. Agoura Road
Calabasas 91302
Tel: (818) 880-9686
FAX: (818) 880-4687

Arrow/Schweber Electronics
9511 Ridgehaven Court
San Diego 92123
Tel: (619) 565-4800
FAX: (619) 279-0862

Arrow/Schweber Electronics
1180 Murphy Avenue
San Jose 95131
Tel: (408) 441-9700
FAX: (408) 453-4810

Arrow/Schweber Electronics
48834 Kato Rd., Suite 103
Fremont 94538
Tel: (510) 440-2681
FAX: (510) 490-1084

Arrow/Schweber Electronics
6 Cromwell, Suite 100
Irvine 92718
Tel: (714) 838-5422
FAX: (714) 454-4203

Avnet Computer
3170 Pullman Street
Costa Mesa 92626
Tel: (714) 641-4179
FAX: (714) 641-4170

Avnet Computer
1361 B West 190th Street
Gardena 90248
Tel: (310) 217-6830
FAX: (310) 327-5389

Avnet Computer
1175 Bordeaux Drive
Sunnyvale 94089
Tel: (408) 743-3454
FAX: (408) 743-3348

Hamilton/Avnet
3170 Pullman Street
Costa Mesa 92626
Tel: (714) 641-4182
FAX: (714) 641-4149

Hamilton/Avnet
1175 Bordeaux Drive
Sunnyvale 94089
Tel: (408) 743-3300
FAX: (408) 745-6679

Hamilton/Avnet
4545 Viewridge Avenue
San Diego 92123
Tel: (619) 571-7540
FAX: (619) 277-6136

Hamilton/Avnet
21150 Califa St.
Woodland Hills 91367
Tel: (818) 594-0404
FAX: (818) 594-8233

Hamilton/Avnet
755 Sunrise Avenue, #150
Roseville 95661
Tel: (916) 925-2216
FAX: (916) 925-3478

Pioneer Technologies Group
134 Rio Robles
San Jose 95134
Tel: (408) 954-9100
FAX: (408) 954-9113

Pioneer Standard
217 Technology Dr., #110
Irvine 92718
Tel: (714) 753-5090
FAX: (714) 753-5074

Pioneer Standard
5850 Canoga Ave., #400
Woodland Hills 91367
Tel: (818) 883-4640
FAX: (818) 883-9721

Wyle Laboratories
2951 Sunrise Blvd., #175
Rancho Cordova 95742
Tel: (916) 638-5282
FAX: (916) 638-1491

Wyle Laboratories
9525 Chesapeake Drive
San Diego 92123
Tel: (619) 565-9171
FAX: (619) 365-0512

Wyle Laboratories
3000 Bowers Avenue
Santa Clara 95051
Tel: (408) 727-2500
FAX: (408) 727-7359

Wyle Laboratories
17872 Cowan Avenue
Irvine 92714
Tel: (714) 863-9953
FAX: (714) 251-0365

Wyle Laboratories
26010 Mureau Road, #150
Calabasas 91302
Tel: (818) 880-9000
FAX: (818) 880-5510

COLORADO

Arrow/Schweber Electronics
61 Inverness Dr. East, #105
Englewood 80112
Tel: (303) 799-0258
FAX: (303) 799-4303

Hamilton/Avnet
9605 Maroon Circle, #200
Englewood 80112
Tel: (303) 799-7800
FAX: (303) 799-7801

Wyle Laboratories
451 E. 124th Avenue
Thornton 80241
Tel: (303) 457-9953
FAX: (303) 457-4831

CONNECTICUT

Arrow/Schweber Electronics
12 Beaumont Road
Wallingford 06492
Tel: (203) 265-7741
FAX: (203) 265-7988

Avnet Computer
55 Federal Road, #103
Danbury 06810
Tel: (203) 797-2880
FAX: (203) 791-2896

Hamilton/Avnet
55 Federal Road, #103
Danbury 06810
Tel: (203) 743-9799
FAX: (203) 797-0373

Pioneer-Standard
2 Trap Falls Rd., #101
Shelton 06484
Tel: (203) 929-5600
FAX: (203) 929-9791

FLORIDA

Arrow/Schweber Electronics
400 Fairway Drive, #102
Deerfield Beach 33441
Tel: (305) 429-8200
FAX: (305) 428-3991

Arrow/Schweber Electronics
37 Skyline Drive, #3101
Lake Mary 32746
Tel: (407) 333-9300
FAX: (407) 333-9320

Avnet Computer
541 S. Orlando Ave., #203
Maitlan 32751
Tel: (407) 539-2888
FAX: (407) 539-2085

Hamilton/Avnet
5371 N.W. 33rd Ave., #204
Ft. Lauderdale 33309
Tel: (305) 733-6300
FAX: (305) 484-8369

Hamilton/Avnet
3247 Tech Drive
St. Petersburg 35805
Tel: (813) 573-4346
FAX: (813) 572-0833

Hamilton/Avnet
7079 University Boulevard
Winter Park 32792
Tel: (407) 657-3300
FAX: (407) 678-4414

Pioneer Technologies Group
337 Northlake Blvd., #1000
Alta Monte Springs 32701
Tel: (407) 834-9090
FAX: (407) 834-0865

Pioneer Technologies Group
674 S. Military Trail
Deerfield Beach 33442
Tel: (305) 428-8877
FAX: (305) 481-2950

GEORGIA

Arrow Commercial Systems Group
3400 C. Corporate Way
Duluth 30136
Tel: (404) 623-8825
FAX: (404) 623-8802

Arrow/Schweber Electronics
4250 E. Rivergreen Pkwy., #E
Duluth 30136
Tel: (404) 497-1300
FAX: (404) 476-1493

Avnet Computer
3425 Corporate Way, #G
Duluth 30136
Tel: (404) 623-5400
FAX: (404) 476-0125

Hamilton/Avnet
3425 Corporate Way, #G
Duluth 30136
Tel: (404) 623-5475
FAX: (404) 623-5490

Pioneer Technologies Group
4250 C. Rivergreen Parkway
Duluth 30136
Tel: (404) 623-1003
FAX: (404) 623-0665

ILLINOIS

Arrow/Schweber Electronics
1140 W. Thorndale Rd.
Itasca 60143
Tel: (708) 250-0500
FAX: 708-250-0916

Avnet Computer
1124 Thorndale Avenue
Bensenville 60106
Tel: (708) 860-8573
FAX: (708) 773-7978

Hamilton/Avnet
1130 Thorndale Avenue
Bensenville 60106
Tel: (708) 860-7700
FAX: (708) 860-8532

MTI Systems
1140 W. Thorndale Avenue
Itasca 60143
Tel: (708) 250-8222
FAX: (708) 250-8275

Pioneer-Standard
2171 Executive Dr., #200
Addison 60101
Tel: (708) 495-9680
FAX: (708) 495-9831

INDIANA

Arrow/Schweber Electronics
7108 Lakeview Parkway West Dr.
Indianapolis 46268
Tel: (317) 299-2071
FAX: (317) 299-2379

Avnet Computer
655 W. Carmel Dr., #120
Carmel 46032
Tel: (317) 575-8029
FAX: (317) 844-4964

Hamilton/Avnet
485 Gradle Drive
Carmel 46032
Tel: (317) 844-9533
FAX: (317) 844-5921

Pioneer-Standard
9350 Priority Way West Dr.
Indianapolis 46250
Tel: (317) 573-0880
FAX: (317) 573-0979

IOWA

Hamilton/Avnet
2335A Blairsferry Rd., N.E.
Cedar Rapids 52402
Tel: (319) 362-4757
FAX: (319) 393-7050

KANSAS

Arrow/Schweber Electronics
9801 Legler Road
Lenexa 66219
Tel: (913) 541-9542
FAX: (913) 752-2612

Avnet Computer
15313 W. 95th Street
Lenexa 61219
Tel: (913) 541-7989
FAX: (913) 541-7904

Hamilton/Avnet
15313 W. 95th Street
Overland Park 66215
Tel: (913) 888-1055
FAX: (913) 541-7951

ntel®

NORTH AMERICAN DISTRIBUTORS (Contd.)

NTUCKY

nilton/Avnet
7 Mercer Rd., #G
ington 40511
(606) 288-4911
(606) 288-4936

RYLAND

ow/Schweber Electronics
0J Patuxent Woods Dr.
umbia 21046
(301) 596-7800
(301) 596-7821

ow Commercial Systems Group
Perry Parkway
thersburg 20877
(301) 670-1600
(301) 670-0188

net Computer
2 Columbia Gateway Dr.
umbia 21046
(301) 995-3571
(301) 995-3515

nilton/Avnet
2 Columbia Gateway Dr., #F
umbia 21046
(301) 995-3554
(301) 995-3553

rth Atlantic Industries
tems Division
5 River Wood Dr.
umbia 21046
(301) 312-5800
(301) 312-5850

neer Technologies Group
0 Gaither Road
thersburg 20877
(301) 921-0660
(301) 921-4255

SSACHUSETTS

ow Commercial Systems Group
Upton Drive
mington 01887
(508) 658-7100
(508) 658-0977

ow/Schweber Electronics
Upton Dr.
mington 01887
(508) 658-0900
(508) 694-1754

net Computer
D Centennial Drive
abody 01960
(508) 532-9822
(508) 532-9887

milton/Avnet
D Centennial Drive
abody 01960
(508) 531-7430
(508) 532-9802

neer-Standard
Hartwell Avenue
dington 02173
(617) 861-9200
(617) 863-1547

le Laboratories
Third Avenue
rlington 01803
(617) 272-7300
(617) 272-6809

CHIGAN

ow/Schweber Electronics
880 Haggerty Road
onia 48152
(800) 231-7902
(313) 462-2686

net Computer
650 Garden Brook Rd. #120
vi 48375
(313) 347-4067
(313) 347-1820

milton/Avnet
76 28th Street, S.W., #5
andville 49418
(616) 531-0345
(616) 531-0059

Hamilton/Avnet
41650 Garden Brook Rd., #100
Novi 48375
Tel: (313) 347-4270
FAX: (313) 347-4021

Pioneer-Standard
13485 Stamford Ct.
Livonia 48150
Tel: (313) 525-1800
FAX: (313) 427-3720

MINNESOTA

Arrow/Schweber Electronics
10100 Viking Drive, #100
Eden Prairie 55344
Tel: (612) 941-5280
FAX: (612) 829-8007

Avnet Computer
9800 Bren Road, East
Minnetonka 55343
Tel: (612) 829-0025
FAX: (612) 944-0638

Hamilton/Avnet
9800 Bren Road, East, #410
Minnetonka 55343
Tel: (612) 932-0600
FAX: (612) 932-0613

Pioneer-Standard
7625 Golden Triange Dr., #G
Eden Prairie 55344
Tel: (612) 944-3355
FAX: (612) 944-3794

MISSOURI

Arrow/Schweber Electronics
2380 Schuetz Road
St. Louis 63146
Tel: (314) 567-6888
FAX: (314) 567-1164

Avnet Computer
741 Goddard Avenue
Chesterfield 63005
Tel: (314) 537-2725
FAX: (314) 537-4248

Hamilton/Avnet
741 Goddard
Chesterfield 63005
Tel: (314) 537-4265
FAX: (314) 537-4248

NEW HAMPSHIRE

Avnet Computer
2 Executive Park Drive
Bedford 03102
Tel: (800) 442-8638
FAX: (603) 624-2402

NEW JERSEY

Arrow/Schweber Electronics
4 East Stow Rd., Unit 11
Marlton 08053
Tel: (609) 596-8000
FAX: (609) 596-9632

Arrow/Schweber Electronics
43 Route 46 East
Pine Brook 07058
Tel: (201) 227-7880
FAX: (201) 227-2064

Avnet Computer
1B Keystone Ave., Bldg. 36
Cherry Hill 08003
Tel: (609) 424-8962
FAX: (609) 751-2502

Hamilton/Avnet
1 Keystone Ave., Bldg. 36
Cherry Hill 08003
Tel: (609) 424-0110
FAX: (609) 751-2611

Hamilton/Avnet
10 Lanidex Plaza West
Parsippany 07054
Tel: (201) 515-5300
FAX: (201) 515-1600

MTI Systems Sales
43 US Rt. 46
Pinebrook 07058
Tel: (201) 882-8780
FAX: (201) 882-8901

Pioneer-Standard
14A Madison Rd.
Fairfield 07004
Tel: (201) 575-3510
FAX: (201) 575-3454

NEW MEXICO

Alliance Electronics, Inc.
10510 Research Ave., SE
Albuquerque 87123
Tel: (505) 292-3360
FAX: (505) 275-6392

Avnet Computer
7801 Academy Rd., SE
Bldg. 1, Suite 204
Albuquerque 87109
Tel: (505) 828-9722
FAX: (505) 828-0364

Hamilton/Avnet
7801 Academy Rd., NE
Bldg. 1, Suite 204
Albuquerque 87108
Tel: (505) 828-1058
FAX: (505) 828-0360

NEW YORK

Arrow/Schweber Electronics
3375 Brighton Henrietta Townline Rd.
Rochester 14623
Tel: (716) 427-0300
FAX: (716) 427-0735

Arrow/Schweber Electronics
20 Oser Avenue
Hauppauge 11788
Tel: (516) 231-1000
FAX: (516) 231-1072

Avnet Computer
933 Motor Parkway
Hauppauge 11788
Tel: (516) 434-7443
FAX: (516) 434-7459

Avnet Computer
2060 Townline Rd.
Rochester 14623
Tel: (716) 272-9110
FAX: (716) 272-9685

Hamilton/Avnet
933 Motor Parkway
Hauppauge 11788
Tel: (516) 231-9800
FAX: (516) 434-7426

Arrow Commercial Systems Group
120 Commerce
Hauppauge 11788
Tel: (516) 231-1175
FAX: (516) 435-2389

Hamilton/Avnet
2060 Townline Rd.
Rochester 14623
Tel: (716) 475-9130
FAX: (716) 475-9119

Hamilton/Avnet
103 Twin Oaks Drive
Syracuse 13120
Tel: (315) 453-4000
FAX: (315) 453-4010

MTI Systems
1 Penn Plaza
250 W. 34th Street
New York 10119
Tel: (212) 643-1280
FAX: (212) 643-1288

Pioneer-Standard
68 Corporate Drive
Binghamton 13904
Tel: (607) 722-9300
FAX: (607) 722-9562

Pioneer-Standard
60 Crossway Park West
Woodbury, Long Island 11797
Tel: (516) 921-8700
FAX: (516) 921-2143

Pioneer-Standard
840 Fairport Park
Fairport 14450
Tel: (716) 381-7070
FAX: (716) 381-8774

NORTH CAROLINA

Arrow/Schweber Electronics
5240 Greensdairy Road
Raleigh 27604
Tel: (919) 876-3132
FAX: (919) 878-9517

Avnet Computer
2725 Millbrook Rd., #123
Raleigh 27604
Tel: (919) 790-1735
FAX: (919) 872-4972

Hamilton/Avnet
5250-77 Center Dr. #350
Charlotte 28217
Tel: (704) 527-2485
FAX: (704) 527-8058

Hamilton/Avnet
3510 Spring Forest Drive
Raleigh 27604
Tel: (919) 878-0819
FAX: (919) 954-0940

Pioneer Technologies Group
9401 L-Southern Pine Blvd.
Charlotte 28273
Tel: (704) 527-8188
FAX: (704) 522-8564

Pioneer Technologies Group
2810 Meridian Parkway, #148
Durham 27713
Tel: (919) 544-5400
FAX: (919) 544-5885

OHIO

Arrow Commercial Systems Group
284 Cramer Creek Court
Dublin 43017
Tel: (614) 889-9347
FAX: (614) 889-9680

Arrow/Schweber Electronics
6573 Cochran Road, #E
Solon 44139
Tel: (216) 248-3990
FAX: (216) 248-1106

Arrow/Schweber Electronics
8200 Washington Village Dr.
Centerville 45458
Tel: (513) 435-5563
FAX: (513) 435-2049

Avnet Computer
7764 Washington Village Dr.
Dayton 45459
Tel: (513) 439-6756
FAX: (513) 439-6719

Avnet Computer
2 Summit Park Dr., #520
Independence 44131
Tel: (216) 573-7400
FAX: (216) 573-7404

Hamilton/Avnet
7760 Washington Village Dr.
Dayton 45459
Tel: (513) 439-6633
FAX: (513) 439-6711

Hamilton/Avnet
2 Summit Park Dr., #520
Independence 44131
Tel: (216) 573-7400
FAX: (216) 573-7404

MTI Systems Sales
23404 Commerce Park Rd.
Beachwood 44122
Tel: (216) 464-6688
FAX: (216) 464-3564

Pioneer-Standard
4433 Interpoint Boulevard
Dayton 45424
Tel: (513) 236-9900
FAX: (513) 236-8133

Pioneer-Standard
4800 E. 131st Street
Cleveland 44105
Tel: (216) 587-3600
FAX: (216) 663-3906

elf Certified Small Business per Federal Acquisition Regulations

NORTH AMERICAN DISTRIBUTORS (Contd.)

OKLAHOMA

Arrow/Schweber Electronics
12111 East 51st Street, #101
Tulsa 74146
Tel: (918) 252-7537
FAX: (918) 254-0917

Hamilton/Avnet
12121 E. 51st St., #102A
Tulsa 74146
Tel: (918) 252-7297
FAX: (918) 250-8763

OREGON

Almac/Arrow Electronics
1885 N.W. 169th Place, #106
Beaverton 97006
Tel: (503) 629-8090
FAX: (503) 645-0611

Arrow Commercial Systems Group
1885 N.W. 169th Place
Beaverton 97006-7312
Tel: (503) 629-8090
FAX: (503) 645-0611

Avnet Computer
9150 Southwest Nimbus Ave.
Beaverton 97005
Tel: (503) 627-0900
FAX: (503) 526-6242

Hamilton/Avnet
9409 Southwest Nimbus Ave.
Beaverton 97005
Tel: (503) 627-0201
FAX: (503) 641-4012

Wyle Laboratories
9640 Sunshine Court
Bldg. G, Suite 200
Beaverton 97005
Tel: (503) 643-7900
FAX: (503) 646-5466

PENNSYLVANIA

Avnet Computer
213 Executive Drive, #320
Mars 16046
Tel: (412) 772-1888
FAX: (412) 772-1890

Hamilton/Avnet
213 Executive, #320
Mars 16046
Tel: (412) 772-1881
FAX: (412) 772-1890

Pioneer-Standard
259 Kappa Drive
Pittsburgh 15238
Tel: (412) 782-2300
FAX: (412) 963-8255

Pioneer Technologies Group
500 Enterprise Road
Keith Valley Business Center
Horsham 19044
Tel: (215) 674-4000
FAX: (215) 674-3107

TEXAS

Arrow/Schweber Electronics
3220 Commander Drive
Carrollton 75006
Tel: (214) 380-6464
FAX: (214) 248-7208

Arrow/Schweber Electronics
10899 Kinghurst Dr., #100
Houston 77099
Tel: (713) 530-4700
FAX: (713) 568-8518

Avnet Computer
4004 Beltline, Suite 200
Dallas 75244
Tel: (214) 308-8168
FAX: (214) 308-8129

Avnet Computer
1235 North Loop West, #525
Houston 77008
Tel: (713) 867-7580
FAX: (713) 861-6851

Hamilton/Avnet
1826-F Kramer Lane
Austin 78758
Tel: (512) 832-4306
FAX: (512) 832-4315

Hamilton/Avnet
4004 Beltline, Suite 200
Dallas 75244
Tel: (214) 308-8105
FAX: (214) 308-8141

Hamilton/Avnet
1235 North Loop West, #521
Houston 77008
Tel: (713) 861-8517
FAX: (713) 861-6541

Pioneer-Standard
1826D Kramer Lane
Austin 78758
Tel: (512) 835-4000
FAX: (512) 835-9829

Pioneer-Standard
13765 Beta Road
Dallas 75244
Tel: (214) 263-3168
FAX: (214) 490-6419

Pioneer-Standard
10530 Rockley Road, #100
Houston 77099
Tel: (713) 495-4700
FAX: (713) 495-5642

Wyle Laboratories
1810 Greenville Avenue
Richardson 75081
Tel: (214) 235-9953
FAX: (214) 644-5064

Wyle Laboratories
4030 West Braker Lane, #420
Austin 78759
Tel: (512) 345-8853
FAX: (512) 345-9330

Wyle Laboratories
11001 South Wilcrest, #100
Houston 77099
Tel: (713) 879-9953
FAX: (713) 879-4069

UTAH

Arrow/Schweber Electronics
1946 W. Parkway Blvd.
Salt Lake City 84119
Tel: (801) 973-6913
FAX: (801) 972-0200

Avnet Computer
1100 E. 6600 South, #150
Salt Lake City 84121
Tel: (801) 266-1115
FAX: (801) 266-0362

Hamilton/Avnet
1100 East 6600 South, #120
Salt Lake City 84121
Tel: (801) 972-2800
FAX: (801) 263-0104

Wyle Laboratories
1325 West 2200 South, #E
West Valley 84119
Tel: (801) 974-9953
FAX: (801) 972-2524

WASHINGTON

Almac/Arrow Electronics
14360 S.E. Eastgate Way
Bellevue 98007
Tel: (206) 643-9992
FAX: (206) 643-9709

Arrow Commercial Systems Group
14360 S.E. Eastgate Way
Bellevue 98007
Tel: (206) 643-9992
FAX: (206) 643-9709

Hamilton/Avnet
17761 N.E. 78th Place, #C
Redmond 98052
Tel: (206) 241-8555
FAX: (206) 241-5472

Avnet Computer
17761 N.E. 78th Place
Redmond 98052
Tel: (206) 867-0160
FAX: (206) 867-0161

Wyle Laboratories
15385 N.E. 90th Street
Redmond 98052
Tel: (206) 881-1150
FAX: (206) 881-1567

WISCONSIN

Arrow/Schweber Electronics
200 N. Patrick Blvd., #100
Brookfield 53045
Tel: (414) 792-0150
FAX: (414) 792-0156

Avnet Computer
20875 Crossroads Circle, #400
Waukesha 53186
Tel: (414) 784-8205
FAX: (414) 784-6006

Hamilton/Avnet
28875 Crossroads Circle, #400
Waukesha 53186
Tel: (414) 784-4511
FAX: (414) 784-9509

Pioneer-Standard
120 Bishop Way #163
Brookfield 53005
Tel: (414) 784-3480
FAX: (414) 784-8207

ALASKA

Avnet Computer
1400 West Benson Blvd., #400
Anchorage 99503
Tel: (907) 274-9899
FAX: (907) 277-2639

CANADA

ALBERTA

Avnet Computer
108 1144 29th Ave., NE
Calgary T2E 7P1
Tel: (403) 291-3284
FAX: (403) 250-1591

Zentronics
6815 8th Street N.E., #100
Calgary T2E 7H7
Tel: (403) 295-8838
FAX: (403) 295-8714

BRITISH COLUMBIA

Almac-Arrow Electronics
8544 Baxter Place
Burnaby V5A 4T8
Tel: (604) 421-2333
FAX: (604) 421-5030

Hamilton/Avnet
8610 Commerce Court
Burnaby V5A 4N6
Tel: (604) 420-4101
FAX: (604) 420-5376

Zentronics
11400 Bridgeport Rd., #108
Richmond V6X 1T2
Tel: (604) 273-5575
FAX: (604) 273-2413

ONTARIO

Arrow Commercial Systems Group
1093 Meyerside Dr., Unit 2
Mississauga, Ontario
L5T 1M4
Tel: (416) 670-7784
FAX: (416) 670-7781

Arrow/Schweber Electronics
36 Antares Dr., Unit 100
Nepean K2E 7W5
Tel: (613) 226-6903
FAX: (613) 723-2018

Arrow/Schweber Electronics
1093 Meyerside, Unit 2
Mississauga L5T 1M4
Tel: (416) 670-7769
FAX: (416) 670-7781

Avnet Computer
151 Superior Blvd.
Mississauga L5T 2L1
Tel: (416) 795-3895
FAX: (416) 795-3855

Avnet Computer
190 Colonnade Road
Nepean K2E 7J5
Tel: (613) 727-2000
FAX: (613) 727-2020

Hamilton/Avnet
151 Superior Blvd.
Mississauga L5T 2L1
Tel: (416) 795-3835
FAX: (416) 564-6036

Hamilton/Avnet
190 Colonnade Road
Nepean K2E 7J5
Tel: (613) 226-1700
FAX: (613) 226-1184

Zentronics
1355 Meyerside Drive
Mississauga L5T 1C9
Tel: (416) 564-9600
FAX: (416) 564-8320

Zentronics
155 Colonnade Rd., South
Unit 17/18
Nepean K2E 7K1
Tel: (613) 226-8840
FAX: (613) 226-6352

QUEBEC

Arrow/Schweber Electronics
1100 St. Regis Blvd.
Dorval H9P 2T5
Tel: (514) 421-7411
FAX: (514) 421-7430

Arrow Commercial Systems Group
500 Ave Street Jean Baptiste
Quebec City 2GE 5R9
Tel: (418) 871-7500
FAX: (418) 871-6816

Avnet Computer
2795 Rue Halpern
St. Laurent H4S 1P8
Tel: (514) 335-2483
FAX: (514) 335-2490

Hamilton/Avnet
2795 Rue Halpern
St. Laurent H4S 1P8
Tel: (514) 335-1000
FAX: (514) 335-2481

Zentronics
520 McCaffrey Street
St. Laurent H4T 1N1
Tel: (514) 737-9700
FAX: (514) 737-5212

EUROPEAN SALES OFFICES

ILAND

ıl Finland OY
›silantie 2
90 Helsinki
(358) 0 544 644
C: (358) 0 544 030

ANCE

ıl Corporation S.A.R.L.
tue Edison-BP 303
54 St. Quentin-en-Yvelines
dex
(33) (1) 30 57 70 00
C: (33) (1) 30 64 60 32

GERMANY

Intel GmbH
Dornacher Strasse 1
8016 Feldkirchen bei Muenchen
Tel: (49) 089/90992-0
FAX: (49) 089/9043948

ISRAEL

Intel Semiconductor Ltd.
Atidim Industrial Park-Neve Sharet
P.O. Box 43202
Tel-Aviv 61430
Tel: (972) 03 498080
FAX: (972) 03 491870

ITALY

Intel Corporation Italia S.p.A.
Milanofiori Palazzo E
20094 Assago
Milano
Tel: (39) (2) 575441
FAX: (39) (2) 3498464

NETHERLANDS

Intel Semiconductor B.V.
Postbus 84130
3009 CC Rotterdam
Tel: (31) 10 407 11 11
FAX: (31) 10 455 4688

RUSSIA

Intel Technologies, Inc.
Krementshugskaya 6/7
121357 Moscow
Tel: 007-095-4439785
FAX: 007-095-4459420
TLX: 612092 smail su.

SPAIN

Intel Iberia S.A.
Zubaran, 28
28010 Madrid
Tel: (34) (1) 308 2552
FAX: (34) (1) 410 7570

SWEDEN

Intel Sweden A.B.
Dalvagen 24
171 36 Solna
Tel: (46) 8 705 5600
FAX: (46) 8 278085

UNITED KINGDOM

Intel Corporation (U.K.) Ltd.
Pipers Way
Swindon, Wiltshire SN3 1RJ
Tel: (44) (0793) 696000
FAX: (44) (0793) 641440

EUROPEAN DISTRIBUTORS/REPRESENTATIVES

STRIA

acher Electronics GmbH
ergasse 6
231 Wien
(43) 1816020
C: (43) 181602400

.GIUM

ıelco Distribution
nue des Croix de Guerre 94
0 Bruxelles
(32) 2 244 2811
C: (32) 2 216 3304

›de Belgium
›berg II, Minervastraat, 14/B2
0 Zaventem
(32) 2 725 46 60
C: (32) 2 725 45 11

NMARK

net Nortec A/S
›nsformervej 17
2730 Herlev
(45) 4284 2000
C: (45) 4492 1552

T Multikomponent AS
›erland 29
2600 Glostrup
(45) 4245 6645
C: (45) 4245 7624

LAND

Y Fintronic AB
kkilantie 2a
00210 Helsinki
(358) 0 682 791
C: (358) 0 682 1251

ANCE

›ow Electronique
79 Rue des Solets
: 585
63 Rungis Cedex
(33) (1) 4978 4978
C: (33) (1) 4978 0596

ıtrologie
r d'Asnieres
avenue Laurent Cely
06 Asnieres Cedex
(33) (1) 4080 9000
C: (33) (1) 4791 0561

kelec
ı des Bruyeres
lue Carle Vernet-BP 2
10 Sevres
(33) (1) 4623 2425
C: (33) (1) 4507 2191

GERMANY

*Electronic 2000
Bauelemente GmbH
Stahlgruberring 12
8000 Muenchen 82
Tel: (49) 89 42110-01
FAX: (49) 89 42110209

*Jermyn GmbH
Im Dachsstueck 9
6250 Limburg
Tel: (49) 6431 5080
FAX: (49) 6431 508289

†Metrologie GmbH
Steinerstrasse 15
8000 Muenchen 70
Tel: (49) 89 724470
FAX: (49) 89 72447111

*Proelectron Vertriebs GmbH
Max-Planck-Strasse 1-3
6072 Dreieich
Tel: (49) 6103 304343
FAX: (49) 6103 304425

†Rein Elektronik GmbH
Loetscher Weg 66
4054 Nettetal 1
Tel: (49) 2153 7330
FAX: (49) 2153 733513

GREECE

†Ergodata
Aigiroupoleos 2A
176 76 Kalithea
Tel: (30) 1 95 10 922
FAX: (30) 1 95 93 160

*Pouliadis Associates Corp.
Koumbari Street 5
Kolonaki Square
106 74 Athens
Tel: (30) 1 36 03 741
FAX: (30) 1 36 07 501

IRELAND

†*Micro Marketing
Taney Hall
Eglinton Terrace
Dundrum
Dublin 14
Tel: (353) (1) 298 9400
FAX: (353) (1) 298 9828

ISRAEL

†*Eastronics Limited
Rozanis 11
P.O.B. 39300
Tel Baruch
Tel-Aviv 61392
Tel: (972) 3 6458 777
FAX: (972) 3 6458 666

ITALY

*Intesi Div. Della Deutsche
Divisione ITT Industries GmbH
P.I. 06550110156
Milanofiori Palazzo e5
20094 Assago (Milano)
Tel: (39) 2 824701
FAX: (39) 2 8242631

*Lasi Elettronica
P.I. 00839000155
Viale Fulvio Testi, N.280
20126 Milano
Tel: (39) 2 661431
FAX: (39) 2 66101385

†Omnilogic Telcom
Via Lorenteggio 270/A
20152 Milano
Tel: (39) 2 48302640
FAX: (39) 2 43802010

NETHERLANDS

†Datelcom
Computerweg 10-16
3600 BD Maarsen
Tel: (31) 3465 95222
FAX: (31) 3465 71245

*Diode Components
Coltbaan 17
3439 NG Nieuwegein
Tel: (31) 3402 9 12 34
FAX: (31) 3402 3 59 24

†*Koning en Hartman
Energieweg 1
2627 AP Delft
Tel: (31) 15 609 906
FAX: (31) 15 619 194

NORWAY

*Avnet Nortec A/S
Postboks 123
N-1364 Hvalstad
Tel: (47) 284 6210
FAX: (47) 284 6545

†Computer System Integration A/S
Postbox 198
N-2013 Skjetten
Tel: (47) 6 84 54 11
FAX: (47) 6 84 53 10

PORTUGAL

*ATD Electronica LDA
Rua dr. Faria de Vasconcelos, 3a
1900 Lisboa
Tel: (351) (1) 847 2200
FAX: (351) (1) 847 2197

†Metrologia Iberica Portugal
Rua Dr. Faria de Vasconcelos 3A
1900 Lisboa
Tel: (351) (1) 847 2202
FAX: (351) (1) 847 2197

SOUTH AFRICA

†*EBE
PO Box 912-1222
Silverton 0127
178 Erasmus Street
Meyerspark
Pretoria 0184
Tel: (27) 12 803 7680-93
FAX: (27) 12 803 8294

SPAIN

*ATD Electronica
Avenue de la Industria, 32, 2B
28100 Alcobendas
Madrid
Tel: (34) (1) 661 6551
FAX: (34) (1) 661 6300

†Metrologia Iberica
Avda. Industria, 32-2
28100 Alcobendas
Madrid
Tel: (34) (1) 661 1142
FAX: (34) (1) 661 5755

SWEDEN

†Avnet Computer AB
Box 184
S-123 23 Farsta
Tel: (46) 8 705 18 00
FAX: (46) 8 735 2373

*Avnet Nortec AB
Box 1830
S-171 27 Solna
Tel: (46) 8705 1800
FAX: (46) 883 6918

*ITT Multikomponent AB
Ankdammsgatan 32
Box 1330
S-171 26 Solna
Tel: (46) 8 830020
FAX: (46) 8 27 13 03

SWITZERLAND

†IMIC Microcomputer
Zurichstrasse
CH-8185 Winkel-Ruti
Tel: (41) (1) 8620055
FAX: (41) (1) 8620266

†*Industrade AG
Hertistrasse 31
CH-8304 Wallisellen
Tel: (41) (1) 8328111
FAX: (41) (1) 8307550

TURKEY

*Empa Electronic
34630 Besyol Londra Asfalti
Florya Is Merkezi Sefakoy
Istanbul
Tel: (90) (1) 599 3050
FAX: (90) (1) 598 5353

UNITED KINGDOM

*Arrow Electronics
St. Martins Business Centre
Cambridge Road
Bedford - MK42 0LF
Tel: (44) 234 270272
FAX: (44) 234 211434

*Avnet EMG Ltd.
Jubilee House
Jubilee Road
Letchworth
Hertsfordshire - SG6 1QH
Tel: (44) 462 488 500
FAX: (44) 462 488 567

*Bytech Components
12a Cedarwood
Chineham Business Park
4 Crockford Lane
Basingstoke
Hants RG12 1RW
Tel: (44) 256 707 107
FAX: (44) 256 707 162

†Bytech Systems
5 The Sterling Centre
Eastern Road
Bracknell
Berks - RG12 2PW
Tel: (44) 344 55 333
FAX: (44) 344 867 270

*Jermyn Electronics
Vestry Estate
Otford Road
Sevenoaks
Kent TN14 5EU
Tel: (44) 732 743 743
FAX: (44) 732 451 251

†Metrologie VA
Rapid House
Oxford Road
High Wycombe
Bucks - HP11 2E
Tel: (44) 494 526 271
FAX: (44) 494 452 144

*MMD/Rapid Ltd.
Rapid Silicon
3 Bennet Court
Bennet Road
Reading
Berks - RG2 0QX
Tel: (44) 734 750 697
FAX: (44) 734 313 255

chnical Distributor
D

INTERNATIONAL SALES OFFICES

AUSTRALIA

Intel Australia Pty. Ltd.
Unit 13
Allambie Grove Business Park
25 Frenchs Forest Road East
Frenchs Forest, NSW, 2086
Sydney
Tel: 61-2-975-3300
FAX: 61-2-975-3375

Intel Australia Pty. Ltd.
711 High Street
1st Floor
East Kw. Vic., 3102
Melbourne
Tel: 61-3-810-2141
FAX: 61-3-819 7200

BRAZIL

Intel Semiconductores do Brazil LTDA
Avenida Paulista, 1159-CJS 404/405
CEP 01311 - Sao Paulo - S.P.
Tel: 55-11-287-5899
FAX: 55-11-287-5119

CHINA/HONG KONG

Intel PRC Corporation
15/F, Office 1, Citic Bldg.
Jian Guo Men Wai Street
Beijing, PRC
Tel: (1) 500-4850
TLX: 22947 INTEL CN
FAX: (1) 500-2953

Intel Semiconductor Ltd.*
32/F Two Pacific Place
88 Queensway
Central
Hong Kong
Tel: (852) 844-4555
FAX: (852) 868-1989

INDIA

Intel Asia Electronics, Inc.
4/2, Samrah Plaza
St. Mark's Road
Bangalore 560001
Tel: 91-812-215065
TLX: 953-845-2646 INTEL IN
FAX: 091-812-215067

JAPAN

Intel Japan K.K.
5-6 Tokodai, Tsukuba-shi
Ibaraki, 300-26
Tel: 0298-47-8511
FAX: 0298-47-8450

Intel Japan K.K.*
Hachioji ON Bldg.
4-7-14 Myojin-machi
Hachioji-shi, Tokyo 192
Tel: 0426-48-8770
FAX: 0426-48-8775

Intel Japan K.K.*
Kawa-asa Bldg.
2-11-5 Shin-Yokohama
Kohoku-ku, Yokohama-shi
Kanagawa, 222
Tel: 045-474-7660
FAX: 045-471-4394

Intel Japan K.K.*
Ryokuchi-Eki Bldg.
2-4-1 Terauchi
Toyonaka-shi, Osaka 560
Tel: 06-863-1091
FAX: 06-863-1084

Intel Japan K.K.
Shinmaru Bldg.
1-5-1 Marunouchi
Chiyoda-ku, Tokyo 100
Tel: 03-3201-3621
FAX: 03-3201-6850

Intel Japan K.K.*
TK Gotanda Bldg. 9F
8-3-6 Nishi Gotanda
Shinagawa, Tokyo 141
Tel: 03-3493-6081
FAX: 03-3493-5951

KOREA

Intel Korea, Ltd.
16th Floor, Life Bldg.
61 Yoido-dong, Youngdeungpo-Ku
Seoul 150-010
Tel: (2) 784-8186
FAX: (2) 784-8096

MEXICO

Intel Tecnologia de Mexico
S.A. de C.V.
Av. Mexico No. 2798-9B, S.H.
44620 Guadalajara, Jal.
Tel. & FAX: 523-640-1259

SINGAPORE

Intel Singapore Technology, Ltd.
101 Thomson Road #08-03/06
United Square
Singapore 1130
Tel: (65) 250-7811
FAX: (65) 250-9256

TAIWAN

Intel Technology Far East Ltd.
Taiwan Branch Office
8th Floor, No. 205
Bank Tower Bldg.
Tung Hua N. Road
Taipei
Tel: 886-2-5144202
FAX: 886-2-717-2455

INTERNATIONAL DISTRIBUTORS/REPRESENTATIVES

ARGENTINA

Dafsys S.R.L.
Chacabuco, 90-6 Piso
1069-Buenos Aires
Tel. & FAX: 54.1334.1871

AUSTRALIA

Email Electronics
15-17 Hume Street
Huntingdale, 3166
Tel: 011-61-3-544-8244
TLX: AA 30895
FAX: 011-61-3-543-8179

NSD-Australia
205 Middleborough Rd.
Box Hill, Victoria 3128
Tel: 03 8900970
FAX: 03 8990819

BRAZIL

Microlinear
Largo do Arouche, 24
01219 Sao Paulo, SP
Tel: 5511-220-2215
FAX: 5511-220-5750

CHILE

Sisteco
Vecinal 40 – Las Condes
Santiago
Tel: 562-234-1644
FAX: 562-233-9895

CHINA/HONG KONG

Novel Precision Machinery Co., Ltd.
Room 728 Trade Square
681 Cheung Sha Wan Road
Kowloon, Hong Kong
Tel: (852) 360-8999
TWX: 32032 NVTNL HX
FAX: (852) 725-3695

GUATEMALA

Abinitio
11 Calle 2 – Zona 9
Guatemala City
Tel: 5022-32-4104
FAX: 5022-32-4123

INDIA

Priya International Limited
D-6, II Floor
Devatha Plaza, 131/132 Residency Rd.
Bangalore 560 025
Tel: (91) 812-214027, 812-214395
FAX: (91) 812-214105

Priya International Limited
Podar Chambers, 4th Floor
109, S.A. Brelvi Road, Fort
Bombay 400 001
Tel: (91) 22-2863611, 22-2863676,
22-2863900, 22-2864026
FAX: (91) 22-2619935

Priya International Limited
Flat No. 8, 10th Floor
Akashdeep Building, Barakhamba Rd.
New Delhi 110 001
Tel: (91) 11-3314512, 11-3310413
FAX: (91) 11-3719107

Priya International Limited
5-J, Century Plaza
560-562 Mount Road, Teynampet
Madras 600 018
Tel: (91) 44-451031, 44-451597

Priya International Limited
No. 10, I Floor
Minerva House, 94 Sarojini Devi Rd.
Secunderabad 500 003
Tel: (91) 842-813549, 842-813120

SES Computers and Technologies Pvt. Ltd.
14, SNS Chambers
239 Palace Upper Orchards
Sankey Road, Sadashivanagar
Bangalore 560 080
Tel: 91-812-348481
FAX: 91-812-343685

SES Computers and Technologies Pvt. Ltd.
Western Express Highway, Andheri (East)
Bombay 400 069
Tel: 91-22-6341584, 91-22-6341667
FAX: 91-22-4937524

SES Computers and Technologies Pvt. Ltd.
605A, Ansal Chambers II
No. 6, Bhikaji Camaplace
New Delhi 110 066
Tel: 91-11-6881663
FAX: 91-11-6840471

JAMAICA

MC Systems
10-12 Grenada Crescent
Kingston 5
Tel: (809) 926-0104
FAX: (809) 929-5678

JAPAN

Asahi Electronics Co. Ltd.
KMM Bldg. 2-14-1 Asano
Kokurakita-ku
Kitakyushu-shi 802
Tel: 093-511-6471
FAX: 093-551-7861

CTC Components Systems Co., Ltd.
4-8-1 Dobashi, Miyamae-ku
Kawasaki-shi, Kanagawa 213
Tel: 044-852-5121
FAX: 044-877-4268

Dia Semicon Systems, Inc.
Flower Hill Shinmachi Higashi-kan
1-23 Shinmachi, Setagaya-ku
Tokyo 154
Tel: 03-3439-1600
FAX: 03-3439-1601

Okaya Koki
2-4-18 Sakae
Naka-ku, Nagoya-shi 460
Tel: 052-204-8315
FAX: 052-204-8380

Ryoyo Electro Corp.
Konwa Bldg.
1-12-22 Tsukiji
Chuo-ku, Tokyo 104
Tel: 03-3546-5011
FAX: 03-3546-5044

KOREA

J-Tek Corporation
Dong Sung Bldg. 9/F
158-24, Samsung-Dong, Kangnam-Ku
Seoul 135-090
Tel: (822) 557-8039
FAX: (822) 557-8304

Samsung Electronics
Samsung Main Bldg.
150 Taepyung-Ro-2KA, Chung-Ku
Seoul 100-102
C.P.O. Box 8780
Tel: (822) 751-3680
TWX: KORSST K 27970
FAX: (822) 753-9065

MEXICO

PSI S.A. de C.V.
Fco. Villa esq. Ajusco s/n
Cuernavaca, MOR 62130
Tel: 52-73-11-1994/5
FAX: 52-73-17-5333

NEW ZEALAND

Email Electronics
36 Olive Road
Penrose, Auckland
Tel: 011-64-9-591-155
FAX: 011-64-9-592-681

SAUDI ARABIA

AAE Systems, Inc.
642 N. Pastoria Ave.
Sunnyvale, CA 94086
U.S.A.
Tel: (408) 732-1710
FAX: (408) 732-3095
TLX: 494-3405 AAE SYS

SINGAPORE

Electronic Resources Pte, Ltd.
17 Harvey Road
#03-01 Singapore 1336
Tel: (65) 283-0888
TWX: RS 56541 ERS
FAX: (65) 289-5327

SOUTH AFRICA

Electronic Building Elements
178 Erasmus St. (off Watermeyet St.)
Meyerspark, Pretoria, 0184
Tel: 011-2712-803-7680
FAX: 011-2712-803-8294

TAIWAN

Micro Electronics Corporation
12th Floor, Section 3
285 Nanking East Road
Taipei, R.O.C.
Tel: (886) 2-7198419
FAX: (886) 2-7197916

Acer Sertek Inc.
15th Floor, Section 2
Chien Kuo North Rd.
Taipei 18479 R.O.C.
Tel: 886-2-501-0055
TWX: 23756 SERTEK
FAX: (886) 2-5012521

URUGUAY

Interfase
Blvr. Espana 2094
11200 Montevideo
Tel: 5982-49-4600
FAX: 5982-49-3040

VENEZUELA

Unixel C.A.
4 Transversal de Monte Cristo
Edf. AXXA, Piso 1, of. 1&2
Centro Empresarial Boleita
Caracas
Tel: 582-238-7749
FAX: 582-238-1816

*Field Application Location

ntel®

NORTH AMERICAN SERVICE OFFICES

PrimeService

Intel Corporation's North American Preferred Service Provider

Central Dispatch: 1-800-800-PRIM (1-800-800-7746)

ıBAMA

irmingham
untsville

ıSKA

nchorage

ZONA

hoenix*
ucson

‹ANSAS

ttle Rock

.IFORNIA

akersfield
rea
arson*
resno
ivermore
lar Del Rey
ntario*
range
acramento*
an Diego*
an Francisco*
anta Clara*
entura
unnyvale
/alnut Creek*
/oodland Hills*

LORADO

olorado Springs
enver
nglewood*

NNECTICUT

ilastonbury*

LAWARE

lew Castle

)RIDA

t. Lauderdale
leathrow
acksonville
lelbourne
ensacola
ampa
Vest Palm Beach

GEORGIA

Atlanta*
Savannah
West Robbins

HAWAII

Honolulu

ILLINOIS

Buffalo*
Calumer City
Chicago
Lansing
Oak Brook

INDIANA

Carmel*
Ft. Wayne

KANSAS

Overland Park*
Wichita

KENTUCKY

Lexington
Louisville
Madisonville

LOUISIANA

Baton Rouge
Metarie

MAINE

Brunswick

MARYLAND

Frederick
Linthicum*
Rockville*

MASSACHUSETTS

Boston*
Natick*
Norton*
Springfield

MICHIGAN

Ann Harbor
Benton Harbor
Flint
Grand Rapids*
Leslie
Livonia*
St. Joseph
Troy*

MINNESOTA

Bloomington*
Deluth

MISSOURI

Springfield
St. Louis*

NEVADA

Minden
Las Vegas
Reno

NEW HAMSHIRE

Manchester*

NEW JERSEY

Edison*
Hamton Town*
Parsippany*

NEW MEXICO

Albuquerque

NEW YORK

Albany*
Amherst*
Dewitt*
Fairport*
Farmingdale*
New York City*

NORTH CAROLINA

Brevard
Charlotte
Greensboro
Haveluch
Raleigh
Wilmington

NORTH DAKOTA

Bismark

OHIO

Cincinnati*
Columbus
Dayton
Independence*
Middle Heights*
Toledo*

OREGON

Beaverton*

PENNSYLVANIA

Bala Cynwyd*
Camp Hill*
East Erie
Pittsburgh*
Wayne*

SOUTH CAROLINA

Charleston
Cherry Point
Columbia
Fountain Inn

SOUTH DAKOTA

Sioux Falls

TENNESSEE

Bartlett
Chattanooga
Knoxville
Nashville

TEXAS

Austin
Bay City
Beaumont
Canyon
College Station
Houston*
Irving*
San Antonio
Tyler

UTAH

Salt Lake City*

VIRGINIA

Charlottesville
Glen Allen
Maclean*
Norfolk
Virginia Beach

WASHINGTON

Bellevue*
Olympia
Renton
Richland
Spokane
Verdale

WASHINGTON D.C.*

WEST VIRGINIA

St. Albans

WISCONSIN

Brookfield*
Green Bay
Madison
Wausau

CANADA

Calgary*
Edmonton
Halifax
London*
Montreal*
Ottawa
Toronto*
Vancouver, BC*
Winnipeg
Regina
St. John

CUSTOMER TRAINING CENTERS

IZONA

mputervision Customer
ucation
)1 W. Behrend Dr., Suite 17
)enix 85027
: 1-800-234-8806

ILLINOIS

Computervision Customer
Education
1 Oakbrook Terrace
Suite 600
Oakbrook 60181
Tel: 1-800-234-8806

MASSACHUSETTS

Computervision Customer
Education
11 Oak Park Drive
Bedford 01730
Tel: 1-800-234-8806

SYSTEMS ENGINEERING OFFICES

NNESOTA

)0 W. 80th Street
ite 360
)omington 55431
: (612) 835-6722

NEW YORK

2950 Expressway Dr., South
Islandia 11722
Tel: (506) 231-3300

arry-in locations

UNITED STATES
Intel Corporation
2200 Mission College Boulevard
P.O. Box 58119
Santa Clara, CA 95052-8119

JAPAN
Intel Japan K.K.
5-6 Tokodai, Tsukuba-shi
Ibaraki, 300-26

FRANCE
Intel Corporation S.A.R.L.
1, Rue Edison, BP 303
78054 Saint-Quentin-en-Yvelines Cedex

UNITED KINGDOM
Intel Corporation (U.K.) Ltd.
Pipers Way
Swindon
Wiltshire, England SN3 1RJ

GERMANY
Intel GmbH
Dornacher Strasse 1
8016 Feldkirchen bei Muenchen

HONG KONG
Intel Semiconductor Ltd.
10/F East Tower
Bond Center
Queensway, Central

CANADA
Intel Semiconductor of Canada, Ltd.
190 Attwell Drive, Suite 500
Rexdale, Ontario M9W 6H8

Order Number: 290483-001
Printed in USA/C92-4004/0393/LB RRD
Peripheral Components