ADVANCE INFORMATION

82374EB/82374SB EISA SYSTEM COMPONENT (ESC)

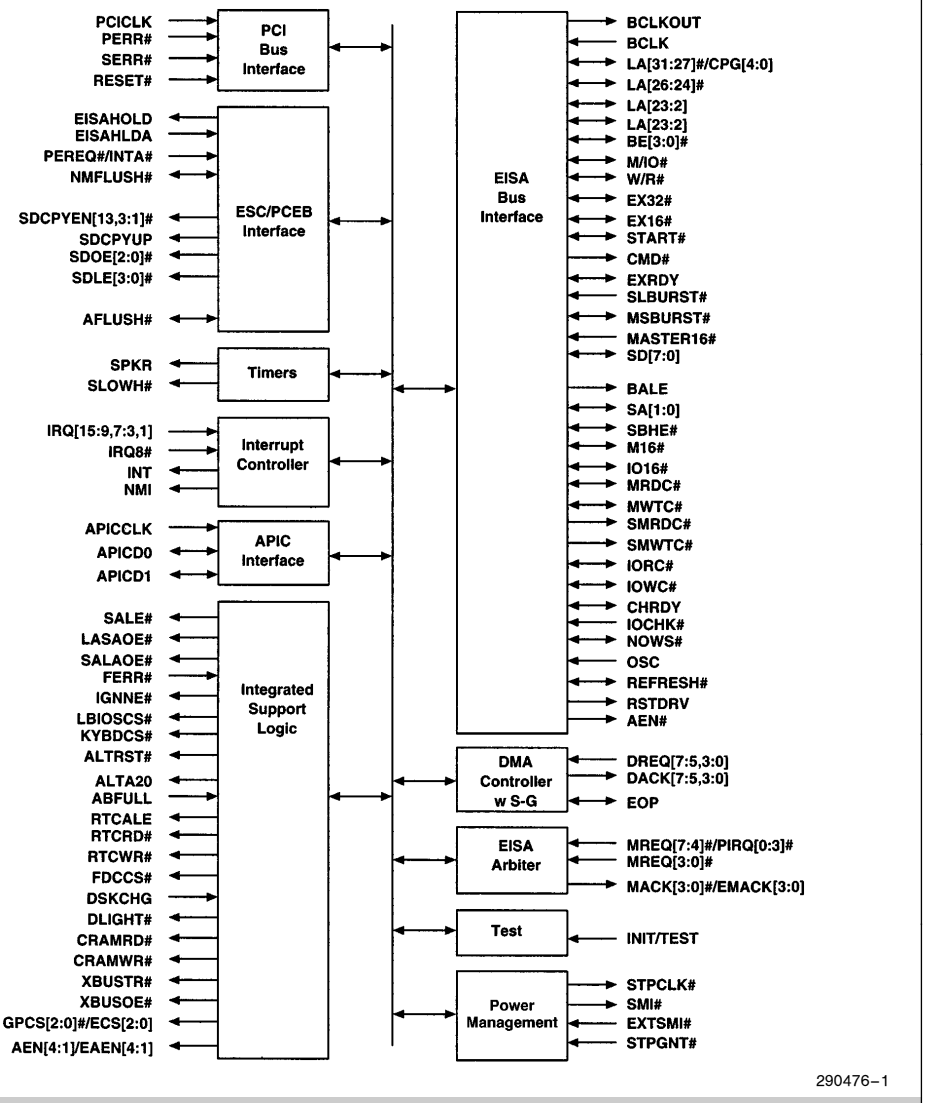
- Integrates EISA Compatible Bus Controller
  - Translates Cycles Between EISA and ISA Bus
  - Supports EISA Burst and Standard Cycles
  - Supports ISA Zero Wait-State Cycles
  - Supports Byte Assembly/ Disassembly for 8-, 16- and 32-Bit Transfers
  - Supports EISA Bus Frequency of up to 8.33 MHz
- Supports 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 Transfer
  - Provides Seven Independently Programmable Channels
  - Integrates Two 82C37A Compatible DMA Controllers
- 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 CPU Speed Control
- Advanced Programmable Interrupt Controller (APIC)
  - Multiprocessor Interrupt Management
  - Separate Bus For Interrupt Messages
- 5V CMOS Technology
- 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
  - 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
- 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 512K Bytes of Flash or EPROM BIOS on the X-Bus
  - Allows BIOS on PCI
  - Supports Integrated VGA BIOS
- 82374SB System Power Management (Intel SMM Support)
  - Fast On/Off Support via SMI GenerationHardware Events, Software Events, EXTSMI#, Fast Off Timer, System Events
  - Programmable CPU Clock Control
  - Enables Energy Efficient Desktop Systems
- Only Available as Part of a Supported Kit
- 208-Pin QFP Package

March 1996 Order Number: 290476-004

This document describes both the 82374EB and 82374SB components. Unshaded areas describe the 82374EB. Shaded areas, like this one, describe the 82374SB operations that differ from the 82374EB.

The 82374EB/SB 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 an EISA (master and slave) interface, EISA bus controller, enhanced seven channel DMA controller with scatter-gather support, EISA arbitration, 14 channel interrupt controller, Advanced Programmable Interrupt Controller (APIC), 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.

The 82374SB also contains support for SMM power management



NOTE:
SMM Interface and INIT are on the 82347SB only.

Simplified ESC Block Diagram

82374EB/82374SB EISA SYSTEM COMPONENT (ESC)

CONTENTS

PAGE

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

PAGE

1.0 ARCHITECTURAL OVERVIEW

The PCI-EISA bridge chip set provides an I/O subsystem core for the next generation of high-performance personal computers (e.g., those based on the Intel486™ or Pentium® processors). System designers can take advantage of the power of the PCI (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/SB PCI-EISA Bridge (PCEB) and the 82374EB/SB EISA System Component (ESC). These components work in tandem to provide an EISA I/O subsystem interface for personal computer platforms based on the PCI standard. This section provides an overview of the PCI and EISA Bus hierarchy followed by an overview of the PCEB and ESC components.

Bus Hierarchy—Concurrent Operations:

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

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

This bus hierarchy allows concurrency for simultaneous operations on all three bus environments. Data buffering permits concurrency for operations that cross over 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).

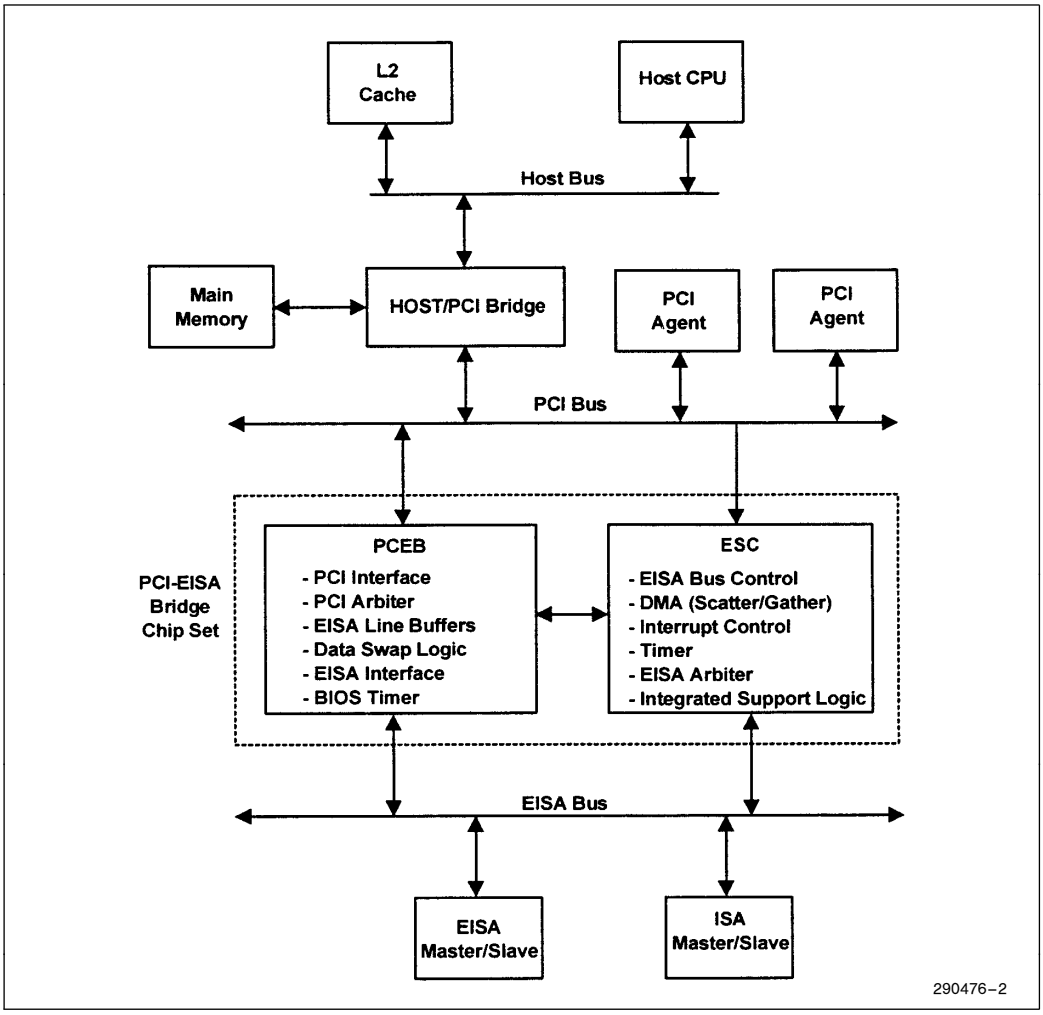


Figure 1. PCI-EISA Chip Set System Block Diagram

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 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 20–33 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 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 second level 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. 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 Bus, the functions include PCI 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 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 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-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 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 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 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 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.

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 $\pm$ 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 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.

Advanced Programmable Interrupt Controller (APIC)

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

Timer/Counter

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

Integrated Support Logic

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

Power Management (82374SB)

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

2.0 SIGNAL DESCRIPTION

This section provides a detailed description of each signal. The signals are arranged in a 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.

NOTE:
During a hard reset, INTR, NMI, IGNNE#, SMI# (on 82374SB), ALTRST#, STPCLK# (on 82374SB) and ALTA20 are driven low to prevent problems associated with 5V/3.3V power sequencing. Any outputs of the ESC that are directed to a 3.3V CPU must be driven through a 5V to 3.3V translator.

2.1 PCI Local Bus Interface Signals

Pin Name	Type	Description
PCICLK	in	**PCI CLOCK:** PCICLK provides timing for all transactions on the PCI 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**: The ESC uses 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.

Pin Name	Type	Description
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.

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#.

Pin Name	Type	Description
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#.
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 writes 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 a 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& 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.

Pin Name	Type	Description
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 pull-up 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 pull-up 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.
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 ignores CHRDY for ISA-Bus master accessing an ISA-Bus slave.
IOCHK#	in	**I/O 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.

Pin Name	Type	Description
NOWS#	o/d	**ZERO WAIT STATES:** NOWS# indicates that an peripheral device wishes to execute a zero wait-state 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 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	**OSCILLATOR:** 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 causes a hardware reset of the devices on the ISA Bus. RSTDRV is asserted whenever the RESET# input to the ESC is asserted.
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 tristate. 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 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 used 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	**SLOT SPECIFIC ADDRESS ENABLE/ENCODED SLOT SPECIFIC ADDRESS ENABLE:** These pins have a 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 slots, 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# indicate that a request for DMA service from the DMA subsystem has been recognized or that an 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 tristated 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>MASTER REQUEST/PCI INTERRUPT REQUEST: 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 behave in the same manner as MREQ[3:0]# signals.

PCI Interrupt Request: PIRQ# are used to 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>Register</th><th colspan="4">Pins</th></tr>
<tr><th>Bit[1:0]</th><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>
</table>

Pin Name	Type	Description
MACK[3:0]#/ EMACK[3:0]	out	**MASTER ACKNOWLEDGE:/ENCODED MASTER ACKNOWLEDGE:** 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.

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 low.
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. **Hardware Reset (Strapping Option)** During hardware reset this signal is an input and the level on the pin at the end of the reset sequence determines where BIOS resides. A high level indicates that BIOS resides on the X-Bus and a low level indicates that BIOS resides on the ISA Bus. The status is used by the ESC, to control the X-Bus transceivers during BIOS access. **NOTE:** For the 82374EB, this pin has an internal weak pull-up of approximately 8 KΩ. For proper configuration of the BIOS location during reset, a weak external pull-down resistor (approx. 500Ω) must be connected to this pin. An external pull-down resistor is not needed for the 82374SB.

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. IRQ8# requires an external pull-up resistor (8 KΩ–10 KΩ).
INTR	out	**CPU INTERRUPT:** INTR 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 APIC Bus Signals

Pin Name	Type	Description
APICCLK	in	**APIC BUS CLOCK:** APICCLK provides the timing reference for the APIC Bus. Changes on APICD[1:0]# are synchronous to the rising edge of APICCLK.
APICD[1:0]	od	**APIC DATA:** APICD1 and APICD0 are the APIC data bus signals. Interrupt messages are sent/received over this bus. APIC arbitration uses APICD1.

2.9 System Power Management Signals (82374SB Only)

Pin Name	Type	Description
STPCLK#	out	**STOP CLOCK:** STPCLK# is asserted by the ESC in response to one of many maskable hardware or software events. For 3.3V processors that are not 5V tolerant, STPCLK# is driven to the CPU STPCLK# pin through a 5V to 3.3V translator. When the CPU samples STPCLK# asserted it responds by stopping its internal clock. After a hard reset, this signal is negated.
SMI#	out	**SYSTEM MANAGEMENT INTERRUPT:** SMI# is asserted by the ESC in response to one of many maskable hardware or software events. For 3.3V processors that are not 5V tolerant, SMI# is driven to the CPU SMI# pin through a 5V to 3.3V translator The CPU recognizes the falling edge of SMI# as the highest priority interrupt in the system. The CPU responds by entering SMM (System Management Mode). SMI# is negated during and following reset. After a hard reset, this signal is negated.

Pin Name	Type	Description
EXTSMI#	in	**EXTERNAL SYSTEM MANAGEMENT INTERRUPT:** EXTSMI# is a falling edge triggered input to the ESC indicating that an external device is requesting the system to enter SMM mode. When enabled via the SMI Enable Register, a falling edge on EXTSMI# results in the assertion of the SMI# signal to the CPU. EXTSMI# is an asynchronous input to the ESC.
INIT/TEST	in	**INITIALIZE/TEST:** On the 82374SB, the function of this pin is selected by the value on the GPCS0# pin at reset. If GPCS0# is low, INIT is selected and if GPCS0# is high, TEST is selected. On the 82374EB, this pin only functions as the TEST pin. **INIT** INIT is connected to the INIT pin on the CPU and indicates to the ESC that a CPU soft reset is occuring. When asserted, the ESC ensures that STPCLK# is negated when the CPU comes out of the soft reset. The ESC also blocks SMI# generation when INIT is asserted. **TEST** For TEST signal description, see the TEST signal section.
STPGNT#	in	**STPCLK# GRANT:** When asserted, STPGNT# indicates to the ESC that a Stop grant PCI special cycle was recognized by the PCEB. The ESC may then negate the STPCLK# signal when the STPCLK# Timer expires.

2.10 ESC/PCEB Interface Signals

2.10.1 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, the PCEB. This signal is synchronous to PCICLK and is asserted when RESET# is asserted.
EISAHLDA	in	**EISA HOLD ACKNOWLEDGE:** EISAHLDA in 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 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 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 an PCI agent. This signal is synchronous to the PCICLK and it is driven inactive when RESET# is asserted.

2.10.2 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 the ESC samples the 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 is negated by the ESC at reset.
AFLUSH#	t/s	**APIC FLUSH:** AFLUSH# is bi-directional signal between the PCEB and ESC that controls system buffer flushing on behalf of the APIC. After a reset the ESC negates AFLUSH# until the APIC is initialized and the first interrupt request is recognized.
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# enable 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 signal to enable the SD data buffers as follows: SDOE0#: Enables byte lane 0 SD[7:0] SDOE1#: Enables byte lane 1 SD[15:8] SDOE2#: Enables byte lane 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]

2.11 Integrated Logic Signals

2.11.1 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.

2.11.2 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 INTR from the ESC will be asserted.
IGNNE#	out	**IGNORE NUMERIC 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. During reset, this signal is driven low.

2.11.3 BIOS INTERFACE

Pin Name	Type	Description
LBIOSCS#	out	**LATCHED BIOS CHIP-SELECT:** LBIOSCS# indicates the 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#.

2.11.4 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 and 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. During reset, this signal is driven 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. See the CLKDIV Register desciption for programming the ABFULL function. If this function is not used, ABFULL should be tied low through a 1K resistor.

2.11.5 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# / PIRQ3#	out	**REAL TIME CLOCK READ COMMAND/PCI INTERRUPT REQUEST 3:** This signal pin has two functions and the function is selected via the Mode Select Register. When functioning as RTCRD#, this signal 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. For details on PIRQ3#, see the Mode Select Register description. For the PIRQ3# function, an external pull-up resistor (10-20 K) must be added to this signal.

Pin Name	Type	Description
RTCWR#/ PIRQ2#	out	**REAL TIME CLOCK WRITE COMMAND/PCI INTERRUPT REQUEST 2:** This signal pin has two functions, and the function is selected via the Mode Select Register. When functioning as RTCWR#, this signal 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. For details on PIRQ2#, see the Mode Select Register description. For the PIRQ2# function, an external pull-up resistor (10K–20K) must be added to this signal.

2.11.6 FLOPPY DISK CONTROLLER INTERFACE

Pin Name	Type	Description
FDCCS#/ PIRQ1#	out	**FLOPPY DISK CONTROLLER CHIP SELECT/PCI INTERRUPT REQUEST 1:** This signal has two functions and the function is selected via the Mode Select Register. As FDCCS# is asserted for I/O cycles to the floppy drive controller. When functioning as FDCCS#, this signal is also asserted when IDECS1# is decoded. See the Mode Select Register description for details on the PIRQ1# function of this signal. Note that for the PIRQ1# function, an external pull-up resistor (10 KΩ–20 KΩ) must be added to this signal.
DSKCHG	in	**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 that the primary and secondary locations are programmed in the X-Bus Address Decode Enable/Disable Register ''A''. **FDCCS# Decode** / **IDECSx# Decode** / **State of SD7 (output)** / **State of XBUSOE#** Enabled / Enabled / Tri-stated / Enabled Enabled / Disabled / Driven via DSKCHG / Disabled Disabled / Enabled / Tri-stated / Disabled (note) Disabled / Disabled / Tri-stated / Disabled **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 enable for accesses to the floppy disk controller.

Pin Name	Type	Description
DLIGHT#/ PIRQ0#	out	**FIXED DISK ACTIVITY LIght/PCI INTERRUPT REQUEST 0:** This signal has two functions, depending on the programming of the Mode Select Register. As DLIGHT#, this signal controls 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. For the PIRQ0# function, see the Mode Select Register description. Note that for the PIRQ0# function, an external pull-up resistor (10 KΩ–20 KΩ) must be added to this signal.

2.11.7 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.

2.11.8 X-BUS CONTROL AND GENERAL PURPOSE DECODE

Pin Name	Type	Description
XBUST/R#	out	**X-BUS DATA TRANSMIT/RECEIVE:** XBUST/R# 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). XBUST/R# is driven high (transmit) during I/O and memory reads for EISA and ISA masters. For DMA cycles (channel 2 only), XBUST/R# is driven high for the following cases: 1. Memory read, I/O write cycles where LBIOSCS# is asserted. 2. I/O read, memory write cycles where Digital Output Register bit 3 is set to 1. XBUST/R# is driven low (receive) under all other conditions.

Pin Name	Type	Description
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). For EISA and ISA master memory read or write cycles, XBUSOE# is asserted when LBIOSCS# is asserted. Otherwise, XBUSOE# is not asserted. For EISA and ISA master I/O read or write cycles, SBUSOE# is asserted if an ISC supported X-Bus device has been decoded, and the decoding for that device has been enabled via the proper configuration registers. An exception to this is 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 present on the X-Bus). In this case, XBUSOE# is not asserted. XBUSOE# is also not asserted during an I/O access to the floppy controller if DSKCHG is sampled low at reset. XBUSOE# is not asserted during DMA cycles, except for channel 2 DMA. For channel 2 DMA, XBUSOE# is asserted.
GPCS[2:0]# / ECS[2:0]	out	**GENERAL PURPOSE CHIP SELECT/ENCODED CHIP SELECT:** 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. **Hardware Reset (Test Mode)** **82374SB:** During Reset, GPCS0/ECS0 is an input signal. The level of this signal is sampled at the end of the reset sequence to determine whether the TEST pin is used as the current TEST function (sampled ‘‘1’’) or as the INIT signal (sampled ‘‘0’’). After reset, the existing GPCS/ECS functionality on this pin is maintained. Note that an internal pull-up of approximately 8 KΩ is included on this pin. If the INIT mode on the TEST pin is to be selected, an external pull-down of approximately 500Ω should be connected to the pin.

2.12 Test Signal

Pin Name	Type	Description
INIT/TEST	in	**TEST:** On the 82374EB, this pin only functions as a TEST pin.
		On the 82374SB, the function of this pin is selected by the value on the GPCS0# pin at reset. If GPCS0# is low, INIT is selected and if GPCS0# is high, TEST is selected.
		INIT For INIT signal description, see the Power Management Signal section.
		TEST TEST is used to tri-state all of the outputs. For normal operations, this signal should be tied to V_{CC}. For test mode, this pin should be tied to ground.

3.0 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 accessible 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.

The following notation is used to describe register access attributes:

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

WO **Write Only**. If a register is write only, reads have no effect.

R/W **Read/Write**. A register with this attribute can be read and written. Note that individual bits in some read/write registers may be read only.

3.1 Configuration Registers

The ESC's 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 the Address Decode section for a summary of configuration register index addresses.

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

In addition to reserved bits within a register, the ESC's configuration space contains address locations that are marked ''Reserved'' (See Address Decode Section). The ESC responds to accesses to these address locations by completing the Host cycle. When a reserved register location is read, 0000h is returned. Writes to reserved registers have no effect on the ESC.

3.1.1 ESCID—ESC ID REGISTER

Address Offset: 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.

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

Address Offset: 08h
Default Value: 02h (82374EB: A-2 stepping)
03h (82374SB: B-0 stepping)
Attribute: Read only
Size: 8 Bits

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

Bit	Description
7:0	**Revision ID Byte:** These bits contain the stepping information about the device. The register is hardwired to the default value during manufacturing. The register is read only. Writes have no effect on the register value.

3.1.3 MS—MODE SELECT REGISTER

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

This register selects the various functional modes of the ESC.

Bit	Description
7	**Reserved**
6	**MREQ[7:4]#/PIRQ[3:0]# Enable:** This bit enables the selected (MREQ[7:4]#/PIRQ[3:0]# functionality. 1 = Enabled; 0 = Disabled
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 bit is 1.

Pin Name	Description
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 (0) or enable (1) the generation of NMI based on SERR# signal pulsing active. When this bit = 1 (and NMIs are enabled via the NMIERTC Register) and SERR# is asserted, the NMI signal is asserted. When this bit = 0, the NMI signal is negated and SERR# is disabled from generating an NMI. Note that other NMI sources are enabled/disabled via the NMISC Register.
2:0	**PIRQx Mux/Mapping Control:** These bits select muxing/mapping of PIRQ[3:0]# with MREQ[7:4] and group of X-Bus signals (DLIGHT#, RTCWR#, RTCRD#). Different bit combinations select the number of EISA slots or group of X-Bus signals which can be supported with the certain number of PIRQx# signals by determining the functionality of pins AEN[4:1]/EAEN[4:1], MACK[3:0]#/EMACK[3:0]#, MREQ[7:4]#/PIRQ[3:0]#, DLIGHT#/ PIRQ0#, FDCCS#/PIRQ1#, RTCWR#/PIRQ2#, RTCRD#/PIRQ3# as shown in Table 1.

Table 1. Mode Select Register

Bits [2:0]	Signal Function						
	AEN[4:1]/ EAEN[4:1]#	**MACK[3:0]#/ EMACK[3:0]#**	**MREQ[7:4#/ PIRQ[0:3]#**	**DLIGHT#/ PIRQ0#**	**FDDCS#/ PIRQ1#**	**RTCWR#/ PIRQ2#**	**RTCRD#/ PIRQ3#**
000	EAEN[4:1]#	EMACK[3:0]#	MREQ[7:4#	PIRQ0#	PIRQ1#	PIRQ2#	PIRQ3#
001	EAEN[4:1]#	EMACK[3:0]#	MREQ[7:4#	PIRQ0#	PIRQ1#	RTCWR#	RTCRD#
010	EAEN[4:1]#	EMACK[3:0]#	MREQ[7:4#	PIRQ0#	FDDCS#	RTCWR#	RTCRD#
011	EAEN[4:1]#	EMACK[3:0]#	MREQ[7:4#	DLIGHT#	FDDCS#	RTCWR#	RTCRD#
100	AEN[4:1]	MACK[3:0]#	PIRQ[0:3]#	DLIGHT#	FDDCS#	RTCWR#	RTCRD#
101	EAEN[4:1]#	EMACK[3:0]#	PIRQ0#, PIRQ1#, MREQ5#, MREQ4#	DLIGHT#	FDDCS#	RTCWR#	RTCRD#
110	EAEN[4:1]#	EMACK[3:0]#	PIRQ0#, MREQ6#, MREQ5#, MREQ4#	DLIGHT#	FDDCS#	RTCWR#	RTCRD#
111	EAEN[4:1]#	EMACK[3:0]#	MREQ[7:4#	DLIGHT#	FDDCS#	RTCWR#	RTCRD#

3.1.4 BIOSCSA—BIOS CHIP SELECT A REGISTER

Address Offset: 42h
Default Value: 10h
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.

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.
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 B REGISTER

Address Offset: 43h
Default Value: 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.

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

Address Offset: 4Dh
Default Value: xx001000b
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) and enable/disable the co-processor error support. In addition, for the 82374SB, the register controls the ABFULL and KBFULL functions.

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	**82374EB: Reserved** **82374SB: ABFULL (With IRQ12):** When bit 4 = 0, the internal IRQ12 is directed to the interrupt controller and transitions on ABFULL have no affect on this interrupt signal. When bit 4 = 1, the assertion of ABFULL is latched and directed to the internal IRQ12 signal in the following manner: • If the interrupt controller is programmed for edge detect mode on IRQ12, a low-to-high transition is generated on the internal IRQ12 signal. Transitions on the IRQ12 input pin are not reflected on the internal IRQ12 signal. • If the interrupt controller is programmed for level-sensitive mode, a high-to-low transition is generated on the internal IRQ12 signal. Transitions on the IRQ12 input pin are also reflected on the internal IRQ12 signal. The latching of the ABFULL signal is cleared by an I/O read of address 60h (no aliasing) or by a hard reset.
3	**82374EB: Reserved** **82374SB: Keyboard Full (KBFULL):** This bit selects the edge-detect KBFULL function on the IRQ1 input signal. When bit 3 = 0, IRQ1 is directed to the interrupt controller. When bit 3 = 1 (default), IRQ1 is latched and directed to the interrupt controller. The latched IRQ1 is cleared by an I/O read of address 60h (no aliasing) or by a hard reset.
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 of 4). **Bit[2:0] — Divisor — BCLK** 000 — 4 (33.33 MHz) — 8.33 MHz 001 — 3 (25 MHz) — 8.33 MHz 010 — Reserved 011 — Reserved 1xx — Reserved

3.1.7 PCSA—PERIPHERAL CHIP SELECT A REGISTER

Address Offset: 4Eh
Default Value: x0000111b
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. It also allows control of where the keyboard controller is physically located (X-Bus or elsewhere).This insures that there is no contention with the X-Bus transceiver driving the system data bus during read accesses to these devices.

Bit	Description
7	**Reserved**
6	**Keyboard Controller Mapping:** 0 = keyboard controller mapped to the X-Bus (default). 1 = keyboard controller not mapped to the X-Bus. When bit 6 = 0, the keyboard controller encoded chip select signal and the X-Bus transceiver enable (XBUSOE#) are generated for accesses to address locations 60h (82374EB/SB), 62h (82374EB only), 64h (82374EB/SB) and 66h (82374EB only). When bit 6 = 1, the keyboard controller chip select signals are generated for accesses to these address locations. However XBUSOE# is disabled. Bit 1 must be 1 for either value of this configuration bit to decode an access to locations 60h, 62h, 64h, or 66h.
5,3:2	**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:

Address	Bit 2	Bit 3	Bit 5	DSKCHG	FDCCS#
X	X	X	X	0	1
3F0h,3F1h	X	1	0	1	0
3F2h–3F7h	1	X	0	1	0 (Note)
370h,371h	X	1	1	1	0
372h–37Fh	1	X	1	1	0 (Note)

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.

Pin Name	Description
4	**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). **82374EB:** 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. **82374SB:** When this bit is set to 0, the IDE encoded chip select signals and the X-Bus transceiver signal (XBUSOE#) are not generated for addresses 1F0h–1F7h (primary) or 170h–177h (secondary) and 3F6h, or 376h. Note that read/write accesses to addresses 377h and 3F7h are not disabled and still generate XBUSOE#.
1	**KEYBOARD CONTROLLER DECODE:** Enables (1) or disables (0) the keyboard controller address locations 60h (82374EB/SB), 62h (82374EB only), 64h (82374EB/SB), and 66h (82374EB only). 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. Note that the value of this bit affects control function (keyboard controlling mapping) provided by bit 6 of this register.
0	**Bit 0: 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.

3.1.8 PCSB—PERIPHERAL CHIP SELECT B REGISTER

Address Offset: 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.

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. **Bits[5:4] Decode** 00 LPT1 (3BCh–3BFh) 01 LPT2 (378h–37Fh) 10 LPT3 (278h–27Fh) 11 Disabled

Pin Name	Description
3:2	**Serial Port B Address Decode:** If either COM1 or COM2 address ranges are selected, these bits default to disabled upon PCIRST. **Bits[3:2] Decode** 00 3F8h–3FFh (COM1) 01 2F8h–2FFh (COM2) 10 Reserved 11 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. **Bits[1:0] Decode** 00 3F8h–3FFh (COM1) 01 2F8h–2FFh (COM2) 10 Reserved 11 Port A disabled

3.1.9 EISAID[4:1]—EISA ID REGISTERS

Address Offset: 50h, 51h, 52h, 53h
Default Value: 00h, 00h, 00h, 00h
Attribute: Read/Write
Size: 8 Bits each

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.

Bit	Description
7:0	**EISA ID Byte:** These bits contain the EISA Motherboard ID information. On power up these bits default to 00h. These bit 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

Address Offset: 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.

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 APICBASE—APIC BASE ADDRESS RELOCATION

Address Offset: 59h
Default Value: 00h
Attribute: Read/Write
Size: 8 Bits

The APICBASE Register provides the modifier for the APIC base address.

82374EB:
APIC is mapped in the CPU memory space at the locations FEC0__x000h and FEC0__x010h (x = 0-Fh). The value of "x" is defined by bits[5:2]. Thus, the relocation register provides a 4 KByte address granularity. The default value of 00h provides APIC unit mapping at the addresses FEC00000h and FEC00010h.

82374SB:
APIC is mapped in the CPU memory space at the locations FEC0__xy00h and FEC0__xy10h (x = 0-Fh, y = 0,4,8,Ch). The value of "y" is defined by bits[1:0] and value of "x" is defined by bits[5:2]. Thus, the relocation register provides a 1 KByte address granularity (i.e. potentially up to 64 I/O APICs can be uniformly addresses in the memory space). The default value of 00h provides APIC unit mapping at the addresses FEC00000h and FEC00010h.

Bit	Description
7:6	**Reserved**
5:2	**X-Base Address—R/W:** Bits[5:2] are compared to host address bits A[15:12], respectively.
1:0	**82374EB: Reserved**
	82374SB: Y-Base Address—R/W: Bits[1:0] are compared to host address bits A[11:10], respectively.

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

Address Offset: 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.

Interrupt Steering Programming Considerations
When using the PCI programmable interrupt steering feature, the following programming considerations apply:

1. Any interrupt steered to by a PIRQx# must be programmed to level sensitive mode.
2. For an interrupt used as a PIRQx#, that IRQ pin is also level sensitive. It is not permissible to use an interrupt on the EISA/ISA Bus as edge triggered as well as on the PCI Bus as level sensitive.
3. Registers that must be programmed when using a PIRQx# include the Mode Select Registers, Edge Level Registers, PIRQ[3:0]# Route Control Registers, and the Interrupt Mask Registers (listed in suggested programming order)

Bit	Description
7	**Routing of Interrupts:** When enabled (0) 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. **Bits[6:0] IRQx# Bits[6:0] IRQx#** 0000000 Reserved 0001001 IRQ9 0000001 Reserved 0001010 IRQ10 0000010 Reserved 0001011 IRQ11 0000011 IRQ3 0001100 IRQ12 0000100 IRQ4 0001101 Reserved 0000101 IRQ5 0001110 IRQ14 0000110 IRQ6 0001111 IRQ15 0000111 IRQ7 0010000 to 0001000 Reserved 1111111 Reserved

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

Address Offset: 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.

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.

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

Address Offset: 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.

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.

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

Address Offset: 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.

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.

3.1.16 GPXBC—GENERAL PURPOSE PERIPHERAL X-BUS CONTROL REGISTER

Address Offset: 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 XBUS or not. If the General Purpose Peripheral is on the X-Bus, then the corresponding bit is set to 1. Otherwise the bit is set to 0.

Bit	Description
7:3	**Reserved**
2	**XBUSOE# Generation for GPCS2#:** When this bit is enabled XBUSOE# will be generated when GPCS2# is generated; 1 = Enabled, 0 = Disabled.
1	**XBUSOE# Generation for GPCS1#:** When this bit is enabled XBUSOE# will be generated when GPCS1# is generated; 1 = Enabled, 0 = Disabled.
0	**XBUSOE# Generation for GPCS0#:** When this bit is enabled XBUSOE# will be generated when GPCS0# is generated;1 = Enabled, 0 = Disabled.

3.1.17 PAC—PCI/APIC CONTROL REGISTER

Address Offset: 70h
Default Value: 00h
Attribute: Read/Write
Size: 8 Bits

The PAC Register controls the operation of the INTR signal in APIC/PIC configuration and the routing of the System Management Interrupt (SMI).

Bit	Description
7:2	**Reserved**
1	**SMI Routing Control (SMIRC):** When SMIRC = 1, the SMI is routed via the APIC. When SMIRC = 0, the SMI is routed via the SMI# signal. Note that when SMRC = 1, INTR can not be routed through the APIC, since it is sharing the APIC interrupt input with SMI#.
0	**INTR Routing Control (INTRC):** When APIC is enabled (in mixed or pure APIC mode), this bit allows the ESC's external INTR signal to be masked (forces INTR to the inactive state but does not tri-states the signal). Thus, the CPU's INTR pin can be used (by providing a simple -gate) for the APIC Local Interrupt (LINTRx). However, INTR must not be masked via this bit when APIC is disabled and INTR is the only mechanism to signal the 8259 recognized interrupts to the CPU. When INTRC = 1, INTR is disabled (APIC must be enabled). When INTRC = 0, INTR is enabled.

3.1.18 TESTC—TEST CONTROL REGISTER

Address Offset: 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.

3.1.19 SMICNTL—SMI CONTROL REGISTER

Address Offset: A0h
Default Value: 08h
Attribute: Read/Write
Size: 8 Bits

For the 82374SB, the SMICNTL Register provides Fast Off Timer control, STPCLK# enable/disable, and CPU clock scaling. This register also enables/disables the system management interrupt (SMI).

Bit	Description
7	**Reserved: Must be 0 when writing this register.**
6:4	**Reserved**
3	**Fast Off Timer Freeze (CTMRFRZ):** This field enables/disables the Fast Off Timer. When this bit is 1, the Fast Off timer stops counting. This prevents time-outs from occurring while executing SMM code. When this bit is 0, the Fast Off timer counts.

Pin Name	Description
2	**STPCLK# Scaling Enable (CSTPCLKSC):** This bit enables/disables control of the STPCLK# high/low times by the clock scaling timers. When bit 2 = 1, the STPCLK# signal scaling control is enabled. When enabled (and bit 1 = 1, enabling the STPCLK# signal), the high and low times for the STPCLK# signal are controlled by the Clock Scaling STPCLK# High Timer and Clock Scaling STPCLK# Low Timer Registers, respectively. When bit 2 = 0 (default), the scaling control of the STPCLK# signal is disabled.
1	**STPCLK# Signal Enable (CSTPCLKE):** This bit permits software to place the CPU into a low power state. When bit 1 = 1, the STPCLK# signal is enabled and a read from the APMC Register causes STPCLK# to be asserted. When bit 1 = 0 (default), the STPCLK# signal is disabled and is negated (high). Software can set this bit to 0 by writing a 0 to it or by any write to the APMC Register.
0	**SMI# Gate (CSMIGATE):** When bit 0 = 1, the SMI# signal is enabled and a system management interrupt condition causes the SMI# signal to be asserted. When bit 0 = 0 (default), the SMI# signal is masked and negated. This bit only affects the SMI# signal and does not effect the detection/recording of SMI events (i.e., This bit does not effect the SMI status bits in the SMIREQ Register). Thus, SMI conditions can be pending when this bit is set to 1. If an SMI is pending when this bit is set to 1, the SMI# signal is asserted.

3.1.20 SMIEN—SMI ENABLE REGISTER

Address Offset: A2-A3h
Default Value: 0000h
Attribute: Read/Write
Size: 16 Bits

For the 82374SB, his register enables the generation of SMI (asserting the SMI# signal) for the associated hardware events (bits[5:0]), and software events (bit 7). When a hardware event is enabled, the occurrence of a corresponding event results in the assertion of SMI#, if enabled via the SMICNTL Register. The SMI# is asserted independent of the current power state (Power-On or Fast Off). The default for all sources in this register is disabled.

Bit	Description
15:8	**Reserved**
7	**APMC Write SMI Enable:** This bit enables SMI for writes to the APMC Register. When bit 7 = 1, writes to the APMC Register generate an SMI. When bit 7 = 0, writes to the APMC Register do not generate an SMI.
6	**EXTSMI# SMI Enable:** When bit 6 = 1, asserting the EXTSMI# input signal generates an SMI. When bit 6 = 0, asserting EXTSMI# does not generate an SMI.
5	**Fast Off Timer SMI Enable:** This bit enables the Fast Off Timer to generate an SMI. When bit 5 = 1, the timer generates an SMI when it decrements to zero. When bit 5 = 0, the timer does not generate an SMI.
4	**IRQ12 SMI Enable (PS/2 Mouse Interrupt):** This bit enables the IRQ12 signal to generate an SMI. When bit 4 = 1, asserting the IRQ12 input signal generates an SMI. When bit 4 = 0, asserting IRQ12 does not generate an SMI.
3	**IRQ8 SMI Enable (RTC Alarm Interrupt):** This bit enables the IRQ8 signal to generate an SMI. When bit 3 = 1, asserting the IRQ8 input signal generates an SMI. When bit 3 = 0, asserting IRQ8 does not generate an SMI.

Bit	Description
2	**IRQ4 SMI Enable (COM2/COM4 Interrupt or Mouse):** Thsi bit enables the IRQ4 signal to generate an SMI. When bit 1 = 1, asserting the IRQ3 input signal generates an SMI. When bit 2 = 0, asserting IRQ4 does not generate an SMI.
1	**IRQ3 SMI Enable (COM1/COM3 Interrupt or Mouse):** This bit enables the IRQ3 signal to generate an SMI. When bit 1 = 1, asserting the IRQ3 input signal generates an SMI. When bit 1 = 0, asserting IRQ3 does not generate an SMI.
0	**IRQ1 SMI Enable (Keyboard Interrupt):** This bit enables the IRQ1 signal to generate an SMI. When bit 0 = 1, asserting the IRQ1 input signal generates an SMI. When bit 0 = 0, asserting IRQ1 does not generate an SMI.

3.1.21 SEE—SYSTEM EVENT ENABLE REGISTER

Address Offset: A4-A7h
Default Value: 00000000h
Attribute: Read/Write
Size: 32 Bits

For the 82374SB, this register enables hardware events as system events or break events for power management control. Note that all of the functional bits in the SEE Register provide system event control. In addition, all bits also provide break event control. The default for each system/break event in this register is disabled.

System events: Activity by these events can keep the system from powering down. When a system event is enabled, the corresponding hardware event activity prevents a Fast Off powerdown condition. Anytime the corresponding hardware event occurs (signal is asserted), the Fast Off Timer is re-loaded with its initial count.

Break events: These events can awaken a powered down system. When a break event is enabled, the corresponding hardware event activity powers up the system by negating STPCLK#. Note that STPCLK# is not negated until the stop grant special cycle has been generated by the CPU. Thus, from the time that STPCLK# is asserted until the stop grant cycle is returned, the occurrence of subsequent break events are latched in the ESC.

NOTE:
INIT is always enabled as a break event. However, INIT only causes a break event after a stop grant special cycle has been received. If INIT is asserted while STPCLK# is active and then negated before the stop grant cycle is received, INIT does not cause a break event.

Bit	Description
31	**Fast Off SMI Enable (FSMIEN):** When bit 31 = 1 (enabled), an SMI causes a system event that re-loads the Fast Off Timer and a break event that negates the STPCLK# signal. When bit 31 = 0 (disabled), an SMI does not re-load the Fast Off Timer or negate the STPCLK# signal.
30	**Reserved**
29	**Fast Off NMI Enable (FNMIEN):** When bit 29 = 1 (enabled), an NMI (e.g., parity error) causes a system event that re-loads the Fast Off Timer and a break event that negates the STPCLK# signal. When bit 29 = 0 (disabled), an SMI does not re-load the Fast Off Timer or negate the STPCLK# signal.

Bit	Description
28:16	**Reserved**
15:3	**Fast Off IRQ[15:3] Enable (FIRQ[15:3]EN):** These bits are used to prevent the system from entering Fast Off and break any current powerdown state when the selected hardware interrupt occurs. When a bit = 1 (enabled), the corresponding interrupt causes a system event that re-loads the Fast Off Timer and a break event that negates the STPCLK# signal. When a bit = 0 (disabled), the corresponding interrupt does not re-load the Fast Off Timer or negate the STPCLK# signal.
2	**Reserved**
1:0	**Fast Off IRQ[1:0] Enable (FIRQ[1:0]EN):** These bits are used to prevent the system from entering Fast Off and break any current powerdown state when the selected hardware interrupt occurs. When a bit = 1, the corresponding interrupt causes a system event that re-loads the Fast Off Timer and a break event that negates the STPCLK# signal. When a bit = 0 (disabled), the corresponding interrupt does not re-load the Fast Off Timer or negate the STPCLK# signal.

3.1.22 FTMR—FAST OFF TIMER REGISTER

Address Offset: A8h
Default Value: 0Fh
Attribute: Read/Write
Size: 8 Bits

For the 82374SB, the Fast Off Timer is used to indicate (through an SMI) that the system has been idle for a pre-programmed period of time. The Fast Off Timer consists of a count-down timer and the value programmed into this register is loaded into the Fast Off Timer when an enabled system event occurs. When the timer expires, an SMI special cycle is generated. When the Fast Off Timer is enabled (bit 3 = 0 in the SMICNTL Register), the timer counts down from the value loaded into this register. The count time interval is one minute. When the Fast Off Timer reaches 00h, an SMI is generated and the timer is re-load with the value programmed into this register. If an enabled system event occurs before the Fast Off Timer reaches 00h, the Fast Off Timer is re-loaded with the value in this register.

NOTE:
Before writing to the FTMR Register, the Fast Off Timer must be stopped via bit 3 of the SMICNTL Register. In addition, this register should NOT be programmed to 00h.

Bit	Description
7:0	**Fast Off Timer Value:** Bits[7:0] contain the starting count value. A read from the FTMR Register returns the value last written.

3.1.23 SMIREQ—SMI REQUEST REGISTER

Address Offset: AA-ABh
Default Value: 00h
Attribute: Read/Write
Size: 16 Bits

For the 82374SB, the SMIREQ Register contains status bits indicating the cause of an SMI. When an enabled event causes an SMI, the ESC automatically sets the corresponding event's status bit to 1. Software sets the status bits to 0 by writing a 0 to them. Only the ESC hardware can set status bits to a 1. Software

writing a 1 to any of the status bits has no effect. If software attempts to set a status bit to 0 at the same time that the ESC is setting it to 1, the bit is set to 1 (i.e., the ESC hardware dominates).

The SMI handler can query the status bits to see what caused the SMI and then branch to the appropriate routine. As the individual routines complete the handler resets the appropriate status bit by writing a 0 to the corresponding bit.

Each of the SMIREQ bits is set by the ESC in response to the activation of the corresponding SMI event. If the SMI event is still active when the corresponding SMIREQ bit is set to 0, the ESC does not set the status bit back to a 1 (i.e., there is only one status indication per active SMI event).

When an IRQx signal is asserted, the corresponding RIRQx bit is set to a 1. If the IRQx signal is still active when software sets the RIRQx bit to 0, RIRQx is not set back to a 1. The IRQx may be negated before software sets the RIRQx bit to 0. If the RIRQx bit is set to 0 at the same time a new IRQx is activated, RIRQx remains at 1. This indicates to the SMI handler that a new SMI event has been detected.

NOTE:
1. The SMIREQ bits are set, cleared, or read independently of each other and independently of the CSMIGATE bit in the SMICNTL Register.
2. If an IRQx is set in level mode and shared by two devices, the IRQ should not be enabled as an SMI# event. The ESC's SMIREQ bits are essentially set with an edge. When the second IRQ occurs on a shared IRQ, there is no second edge and the SMI# will not be generated for the second IRQ.

Bit	Description
15:8	**Reserved**
7	**APM SMI Status (RAPMC):** The ESC sets this bit to 1 to indicate that a write to the APM Control Register caused an SMI. Software sets this bit to a 0 by writing a 0 to it.
6	**EXTSMI# SMI Status (REXT):** The ESC sets this bit to 1 to indicate that EXTSMI# caused an SMI. Software sets this bit to a 0 by writing a 0 to it.
5	**Fast Off Timer Expired Status (RFOT):** The ESC sets this bit to 1 to indicate that the Fast Off Timer expired and caused an SMI. Software sets this bit to a 0 by writing a 0 to it. When the Fast Off Timer expires, the ESC sets this bit to a 1. Note that the timer re-starts counting one the next clock after it expires.
4	**IRQ12 Request SMI Status (RIRQ12):** The ESC sets this bit to 1 to indicate that IRQ12 caused an SMI. Software sets this bit to a 0 by writing a 0 to it.
3	**IRQ8# Request SMI Status:** The ESC sets this bit to 1 to indicate that IRQ8# caused an SMI. Software sets this bit to a 0 by writing a 0 to it.
2	**IRQ4 Request SMI Status:** The ESC sets this bit to 1 to indicate that IRQ4 caused an SMI. Software sets this bit to a 0 by writing a 0 to it.
1	**IRQ3 Request SMI Status:** The ESC sets this bit to 1 to indicate that IRQ3 caused an SMI. Software sets this bit to a 0 by writing a 0 to it.
0	**IRQ1 Request SMI Status:** The ESC sets this bit to 1 to indicate that IRQ1 caused an SMI. Software sets this bit to a 0 by writing a 0 to it.

3.1.24 CTLTMRCLOCK SCALE STPCLK# LOW TIMER

Address Offset: ACh
Default Value: 00h
Attribute: Read/Write
Size: 8 Bits

For the 82374SB, the value in this register defines the duration of the STPCLK# asserted period when bit 2 in the SMICNTL Register is set to 1. The value in this register is loaded into the STPCLK# Timer when STPCLK# is asserted. However, the timer does not start until the Stop Grant Bus Cycle is received. The STPCLK# timer counts using a 32 μs clock.

Bit	Description
7:0	**Clock Scaling STPCLK# Low Timer Value:** Bits[7:0] define the duration of the STPCLK# asserted period during clock throttling.

3.1.25 CTLTMRH—CLOCK SCALE STPCLK# HIGH TIMER

Address Offset: AEh
Default Value: 00h
Attribute: Read/Write
Size: 8 Bits

For the 82374SB, the value in this register defines the duration of the STPCLK# negated period when bit 2 in the SMICNTL Register is set to 1. The value in this register is loaded into the STPCLK# Timer when STPCLK# is negated. The STPCLK# timer counts using a 32 μs clock.

Bit	Description
7:0	**Clock Scaling STPCLK# High Timer Value:** Bits[7:0] define the duration of the STPCLK# negated period during clock throttling.

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 Location: 08h—Channels 0-3
0D0h—Channels 4-7
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 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.

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 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:** 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	**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.
3	**Reserved:** Must be 0.
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.
1:0	**Reserved:** Must be 0.

3.2.2 DCM—DMA CHANNEL MODE REGISTER

Register Location: 0Bh—Channels 0-3
0D6h—Channels 4-7
Default Value: 000000xxb
Attribute: Write Only
Size: 8 Bits

Each channel has a 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.

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** 00 Demand mode 01 Single mode 10 Block mode 11 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** 00 Verify transfer 01 Write transfer 10 Read Transfer 11 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** 00 Channel 0 (4) 01 Channel 1 (5) 10 Channel 2 (6) 11 Channel 3 (7)

3.2.3 DCEM—DMA CHANNEL EXTENDED MODE REGISTER

Register Location: 040Bh—Channels 0-3
04D6h—Channels 4-7
Default Value: 000000xxb
Attribute: Write Only
Size: 8 Bits

Each channel has an Extended Mode Register. 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.

Bit	Description
7	**Stop Register:** 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	**EOP Input/Output:** 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 tristated, and will not effect the DMA operation. Upon reset, the value of bit 6 is 0 (EOP output selected).

Bit	Description
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 automatically 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-/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. 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- 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.

Bit	Description
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 Dword aligned address. If the starting address is not Dword aligned then the DMA controller will do a partial Dword transfer during the first and last 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] select the particular channel that will have its DMA Channel Extend Mode Register programmed with bits[7:2]. **Bits[1:0]** **Channel** 00 Channel 0 (4) 01 Channel 1 (5) 10 Channel 2 (6) 11 Channel 3 (7)

3.2.4 DR—DMA REQUEST REGISTER

Register Location: 09h—Channels 0-3
0D2h—Channels 4-7
Default Value: 000000xxb
Attribute: Write Only
Size: 8 Bits

Each channel has a Request bit associated with it in one of the two 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.

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. **Bits[1:0] Channel** 00 Channel 0 01 Channel 1 (5) 10 Channel 2 (6) 11 Channel 3 (7)

3.2.5 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 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].

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. **Bits[1:0] Channel** 00 Channel 0 (4) 01 Channel 1 (5) 10 Channel 2 (6) 11 Channel 3 (7)

3.2.6 WAMB—WRITE ALL MASK BITS REGISTER

Register Location: 0Fh—Channels 0-3
0DEh—Channels 4-7
Default Value: 0Fh
Attribute: Read/Write
Size: 8 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 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.

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 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.

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 DB&CA—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 register store the lowest 16-bits of the current memory address. The CPU will program the base register set with a reserve buffer.

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 DB&CBW—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 register store the lowest 16-bits of the current Byte/Word Count. The CPU will then program the base register set with a reserve buffer.

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 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 effected. 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 register 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.

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 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
Attribute: Read/Write
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 register 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.

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 DMAP—DMA PAGE REGISTER

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.

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

3.2.13 DMALPR—DMA LOW PAGE REFRESH REGISTER

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.

Bit	Description
7:0	**DMA LOW PAGE REFRESH:** The contents of the bits are driven on to the address bus(LA[23:16]) during refresh.

3.2.14 DMAMHPG—DMA MEMORY HIGH PAGE REGISTER; DMA MEMORY BASE HIGH PAGE REGISTER

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
Attribute: Read/Write
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 register 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.

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 DMAHPGR—DMA HIGH PAGE REGISTER REFRESH

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.

Bit	Description
7:0	**DMA High Page Refresh:** The contents of the bits are driven on to the address bus (LA[31:24]) during refresh.

3.2.16 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 not reset to 0.

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. Bits[23:16] are the upper stop bits. Bits[15:8] are the mid stop bits and bits[7:2] are the lower stop bits. Bits[1:0] are not used and are don't cares.

3.2.17 CHAIN—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.

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; 1 = generate TC, 0 = Generate IRQ13; 1 = Programming complete, 0 = Don't start chaining.
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 = Enable chaining, 0 = Disable chaining.
1:0	**DMA Channel Select:** Bits[1:0] select the DMA channel mode register to program with bits[4:2]. **Bits[1:0] Channel** 00 0 or 4 01 1 or 5 10 2 or 6 11 3 or 7

3.2.18 CHAINSTA—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.

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.
4	**Reserved**

3.2.19 CHINTST—CHANNEL INTERRUPT STATUS REGISTER

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.

Bit	Description
7:5, 3:0	**Chaining Interrupt Status:** When a channel interrupt status read returns a 0, bits[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.
4	**Reserved**

3.2.20 CHAINBEC—CHAIN BUFFER EXPIRATION CONTROL REGISTER

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.

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.
4	**Reserved**

3.2.21 SCATGA—SCATTER-GATHER COMMAND REGISTER

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 2.

Table 2. 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 Table 2 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.

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 effect 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** **Bits[1:0] Function** 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 SCAGAST—SCATTER-GATHER STATUS REGISTER

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.

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 is 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**

Bit	Description
0	**S-G Active:** The Scatter-Gather Active bit indicates the current S-G transfer status. Bit 0 will be a 1 after a 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 a S-G Stop command has been issued. Following reset, or Master Clear, this bit is reset to 0.

3.2.23 SCAGAD—SCATTER-GATHER DESCRIPTOR TABLE POINTER REGISTER

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, Relocateable
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 until the 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 that 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.

Bit	Description
31:0	**SGD Table Pointer:** The SGD table pointer register contains a 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 CBPFF—CLEAR BYTE POINTER FLIP FLOP REGISTER

Register Location: 00Ch—Channels 0-3
0D8h—Channels 4-7
Default Value: xxh
Attribute: Write Only
Size: 8 Bits

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).

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 Location: 00Dh—Channels 0-3
0DAh—Channels 4-7
Default Value: xxh
Attribute: Write Only
Size: 8 Bits

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.

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 Location: 00Eh—Channels 0-3
0DCh—Channels 4-7
Default Value: xxh
Attribute: Write Only
Size: n/a

Software Command 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.

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 Section 8.0, Interval Timer

3.3.1 TCW—TIMER CONTROL WORD REGISTER

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.

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.

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** 00 Counter 0 select 01 Counter 1 select 10 Counter 2 select 11 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 read/write least significant byte, read/write most significant byte, or read/write 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** 00 Counter Latch Command (see Section 3.3.3) 01 R/W Least Significant Byte (LSB) 10 R/W Most Significant Byte (MSB) 11 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. Note that for the fail safe timer (timer 2, counter 0), modes 1, 2, 3, 4, and 5 are reserved. **Bit[3:1] Mode Function** 000 0 Out signal on end of count (=0) 001 1 Hardware retriggerable one-shot (Reserved for timer 2, counter 0.) X10 2 Rate generator (divide by n counter) (Reserved for timer 2, counter 0.) X11 3 Square wave output (Reserved for timer 2, counter 0.) 100 4 Software triggered strobe (Reserved for timer 2, counter 0.) 101 5 Hardware triggered strobe (Reserved for timer 2, counter 0.)
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 104.

3.3.2 TIMER READ BACK COMMAND REGISTER

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.

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 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.

Bit	Description
7:6	**Counter Selection:** Bits 6 and 7 are used to select the counter for latching. **Bit[7:6] Function** 00 Latch counter 0 select 01 Latch counter 1 select 10 Latch counter 2 select 11 Read Back Command select
5:4	**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 TMSTAT—TIMER STATUS BYTE FORMAT REGISTER

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 an 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.

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 CAPS—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.

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 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:

a. 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.

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 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 4 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.

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 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.

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 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.

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 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.

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 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.

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. While different programming options are sometimes offered for bits 2 and 3, 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 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 effect 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.

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 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.

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[7:5] Function** 001 Non-specific EOI command 011 Specific EOI Command 101 Rotate on Non-Specific EOI Command 100 Rotate in Auto EOI Mode (Set) 000 Rotate in Auto EOI Mode (Clear) 111 *Rotate on Specific EOI Command 110 *Set Priority Command 010 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[2:0] Interrupt Level** 000 IRQ 0(8) 001 IRQ 1(9) 010 IRQ 2(10) 011 IRQ 3(11) 100 IRQ 4(12) 101 IRQ 5(13) 110 IRQ 6(14) 111 IRQ 7(15)

3.4.8 OCW3—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.

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 effect 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. **Bits[1:0] Function** 00 No Action 01 No Action 10 Read IRQ Register 11 Read IS Register

3.4.9 ELCR—EDGE/LEVEL CONTROL REGISTER

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.

Programming Considerations:
If an interrupt is switched from level to edge sensitive, a false interrupt is generated on that interrupt line. If the IRQx line is high, then switching the level/edge bet from a 1 to a 0 causes the interrupt controller to detect an interrupt. Also note that even if this interrupt is masked when programming this register, the interrupt controller still latches the false interrupt. As soon as this interrupt is unmasked, the false interrupt is processed.

Thus, before switching the edge/level function, disable interrupts to the processor (either mask interrupts or CLI instruction). Then program the ELCR Register. Finally, re-initialize the interrupt controller to clear the false interrupt.

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 must be set to 0. After A reset or power-on these registers are set to 00h.

Bit	Port 04D0h	Port 04D1h
0	INT0	INT8
1	INT1	INT9
2	INT2	INT10
3	INT3	INT11
4	INT4	INT12
5	INT5	INT13
6	INT6	INT14
7	INT7	INT15

3.4.10 NMISC—NMI STATUS AND CONTROL REGISTER

Register Location: 061h
Default Value: X0X0 0000
Attribute: Read/Write, Read Only
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. This register also controls NMI generation and reports NMI source status. Note that NMI generation is globally enabled/disabled via the NMIERTC Register and NMI generation for SERR# is controlled via the MS Register. Bits[7:4] of this register are read-only and must be written as 0s when writing to this register. Bits[3:0] are read/write. Following reset, bit 7 returns the PCI System Board Parity Error status (PERR#) and bit 5 is undetermined until Counter 2 is properly programmed.

Bit	Description
7	**System Board Error—RO:** Bit 7 is set if the PERR# line is pulsed. 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. Note that this bit does not reflect status of an NMI caused by SERR#, which is enabled and disabled/cleared via the MS Register.
6	**IOCHK# NMI Source—RO:** Bit 6 is set if an expansion board asserts IOCHK# on the ISA/EISA 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.
5	**Timer 1, Counter 2—RO:** 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.
4	**Refresh Cycle Toggle—RO:** The Refresh Cycle Toggle signal toggles from either 0 to 1 or 1 to 0 following every refresh cycle.
3	**IOCHK# NMI Enable—R/W:** When bit 3 is a 1, IOCHK# NMI's are disabled and cleared, and when bit 3 is a 0 (default), IOCHK# NMI's are enabled.
2	**PCI System Board Error—R/W:** When bit 2 is a 1, the system board error is disabled and cleared. When bit 2 is a 0 (default), the system board parity error is enabled. Note that NMI generation for system board errors is enabled/disabled via bit 3 (System Error) of the Mode Select Register. Following reset, bit 2 is a 0, and system board errors are enabled.
1	**Speaker Data Enable—R/W:** 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 (default), 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.
0	**Timer 1, Counter 2 Gate Enable—R/W:** 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.

3.4.11 NMIERTC—NMI CONTROL AND REAL-TIME CLOCK ADDRESS

Register Location: 070h
Default Value: See below
Attribute: Write Only
Size: 8 Bits

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 The NMIERTC Register to mask the NMI signal. This register 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.

Bit	Description
7	**NMI Enable:** Setting bit 7 to a 1 will disable all NMI sources. Setting 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 NMIESC—NMI EXTENDED STATUS AND CONTROL REGISTER

Register Location: 0461h
Default Value: See below
Attribute: Read/Write, Read Only
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.

Bit	Description
7	**Fail-Safe Timer Status—RO:** 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 has 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—RO:** 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—RO:** 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—RO:** 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—R/W:** 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—R/W:** This bit enables/disables NMI when the Fail-Safe Timer timesout. 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—R/W:** 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—R/W:** When bit 0 is a 0, RSTDRV signal function 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 SOFTNMI—SOFTWARE NMI GENERATION REGISTER

Register Location: 0462h
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.

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 CONFRAMP—CONFIGURATION RAM PAGE REGISTER

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 drivers 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.

Bit	Description
7:5	**Reserved**
4:0	**CRAM Page Address:** The value of these bit 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. The value is driven onto CPG[4:0] during accesses to Configuration RAM.

3.5.2 DIGOUT—DIGITAL OUTPUT REGISTER

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 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 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.

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 details.

3.5.3 PORT 92 REGISTER

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.

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 theses 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 LEISAMG—LAST EISA BUS MASTER GRANTED REGISTER

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.

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.

3.6 Power Management Registers

This section describes two power management registers that are in the 82374SBAPMS and APMC Registers. These registers are located in normal I/O space and must be accessed (via the CPU or PCI Bus) with 8-bit accesses. Note that the rest of the power management registers are part of the ESC configuration registers.

3.6.1 APMC—ADVANCED POWER MANAGEMENT CONTROL PORT

I/O Address: 0B2h
Default Value: 00h
Attribute: Read/Write
Size: 8 Bits

This register passes data (APM Commands) between the OS and the SMI handler. In addition, writes can generate an SMI and reads can cause STPCLK# to be asserted. The ESC operation is not effected by the data in this register.

Bit	Description
7:0	**APM Control Port (APMC):** Writes to this register store data in the APMC Register and reads return the last data written. In addition, writes generate an SMI, if bit 7 of the SMIEN Register and bit 0 of the SMICNTL Register are both is set to 1. Reads cause the STPCLK# signal to be asserted, if bit 1 of the SMICNTL Register is set to 1. Reads do not generate an SMI.

3.6.2 APMS—ADVANCED POWER MANAGEMENT STATUS PORT

I/O Address: 0B3h
Default Value: 00h
Attribute: Read/Write
Size: 8 Bits

This register passes status information between the OS and the SMI handler. The ESC operation is not effected by the data in this register.

Bit	Description
7:0	**APM Status Port (APMS):** Writes store data in this register and reads return the last data written.

3.7 APIC Registers

This section describes the registers used to program the Advanced Programmable Interrupt Controller. The I/O APIC registers are accessed by an indirect addressing scheme using two registers (IOREGSEL and IOWIN) that are located in the CPU's memory space (memory address specified by the APICBASE Register). To reference an I/O APIC register, a Dword memory write loads the IOREGSEL Register with a 32-bit value that specifies the APIC register. The IOWIN Register then becomes a four byte window pointing to the APIC register specified by bits[7:0] of the IOREGSEL Register. The register address table is shown in the Address Decode section.

All APIC registers are accessed using 32-bit loads and stores. This implies that to modify a field (e.g., bit, byte) in any register, the whole 32-bit register must be read, the field modified, and the 32-bits written back. In addition, registers that are described as 64-bits wide are accessed as multiple independent 32-bit registers.

3.7.1 IOREGSEL—I/O REGISTER SELECT REGISTER

Memory Address: FEC0 x000h (82374EB) (x = See APICBASE Register)

FEC0 xy00h (82374SB) (xy = See APICBASE Register)

Default Value: 00h
Attribute: Read/Write
Size: 32 Bits

This register selects an I/O APIC Unit register. The contents of the selected 32-bit register can be manipulated via the I/O Window Register.

Bit	Description
31:8	**Reserved**
7:0	**APIC Register Address:** Bits[7:0] specify the APIC register to be read/written via the IOWIN Register.

3.7.2 IOWIN—I/O WINDOW REGISTER

Memory Address: FEC0 x010h (82374EB) (x = See APICBASE Register)

FEC0 xy10h (82374SB) (xy = See APICBASE Register)

Default Value: 00h
Attribute: Read/Write
Size: 32 Bits

This register is mapped onto the I/O Unit's register selected by the IOREGSEL Register. Readability/writability by software is determined by the I/O APIC register that is currently selected.

Bit	Description
31:0	**APIC Register Data:** Memory references to this register are mapped to the APIC register specified by the contents of the IOREGSEL Register.

3.7.3 APICID—I/O APIC IDENTIFICATION REGISTER

Address Offset: 00h
Default Value: 00h
Attribute: Read/Write
Size: 32 Bits

This register contains the unit's 4-bit APIC ID. The ID serves as a physical name of the I/O APIC Unit. All APIC units using the APIC bus should have a unique APIC ID. The APIC bus arbitration ID for the I/O unit is also writtten during a write to the APICID Register (same data is loaded into both). This register must be programmed with the correct ID value before using the I/O APIC unit for message transmission.

Bit	Description
31:28	**Reserved**
27:24	**I/O APIC Identification:** This 4-bit field contains the I/O APIC identification.
23:0	**Reserved**

3.7.4 APICID—I/O APIC IDENTIFICATION REGISTER

Address Offset: 01h
Default Value: 00h
Attribute: Read Only
Size: 32 Bits

The I/O APIC Version Register identifies the APIC hardware version. Software can use this to provide compatibility between different APIC implementations and their versions. In addition, this register provides the maximum number of entries in the I/O Redirection Table.

Bit	Description
31:24	**Reserved**
23:16	**Maximum Redirection Entry:** This field contains the entry number (0 being the lowest entry) of the highest entry in the I/O Redirection Table. The value is equal to the number of interrupt input pins minus one of this I/O APIC. The range of values is 0 through 239. For the ESC, this value is 0Fh.
15:8	**Reserved**
7:0	**APIC VERSION:** This 8 bit field identifies the implementation version. The version numbers are assigned for 82489DX and APIC as follows: 0Xh = 82489DX 1Xh = APIC 20–FFh = Reserved For the ESC, the current value is 11h.

3.7.5 APICARB—I/O APIC ARBITRATION REGISTER

Address Offset: 02h
Default Value: 000F0011h
Attribute: Read Only
Size: 32 Bits

The APICARB Register contains the bus arbitration priority for the I/O APIC. This register is loaded whenever the I/O APIC ID Register is written.

Bit	Description
31:28	**Reserved**
27:24	**I/O APIC Identification:** This 4 bit field contains the I/O APIC identification.
23:0	**Reserved**

3.7.6 IOREDTBL[15:0]—I/O REDIRECTION TABLE REGISTERS

Address Offset: 10-11h (IOREDTBL0)
12-13h (IOREDTBL1)
14-15h (IOREDTBL2)
16-17h (IOREDTBL3)
18-19h (IOREDTBL4)
1A-1Bh (IOREDTBL5)
1C-1Dh (IOREDTBL6)
1E-1Fh (IOREDTBL7)
20-21h (IOREDTBL8)
22-23h (IOREDTBL9)
24-25h (IOREDTBL10)
26-27h (IOREDTBL11)
28-29h (IOREDTBL12)
2A-2Bh (IOREDTBL13)
2C-2Dh (IOREDTBL14)
2E-2Fh (IOREDTBL15)
Default Value: xx000000 00010xxxh
Attribute: Read/Write, Read Only
Size: 64 Bits each

There are 16 I/O Redirection Table entry registers. Each register is a dedicated entry for each interrupt input pin. Unlike IRQ pins of the 8259A, the notion of interrupt priority is completely unrelated to the position of the physical interrupt input pin on the APIC. Instead, software determines the vector (and therefore the priority) for each corresponding interrupt input pin. For each interrupt pin, the operating system can also specify the signal polarity (low active or high active), whether the interrupt is signaled as edges or levels, as well as the destination and delivery mode of the interrupt. The information in the redirection table is used to translate the corresponding interrupt pin information into an inter-APIC message.

For a signal on an edge-sensitive interrupt input pin to be recognized as a valid edge (and not a glitch), the input level on the pin must remain asserted until the I/O APIC Unit broadcasts the corresponding message over the APIC bus and the message has been accepted by the destination(s) specified in the destination field. Only then will the source APIC be able to recognize a new edge on that Interrupt Input pin. That new edge only results in a new invocation of the handler if its acceptance by the destination APIC causes the Interrupt Request Register bit to go from 0 to 1. (In other words, if the interrupt wasn't already pending at the destination.)

Bit	Description
63:56	**Destination Field:** If the Destination Mode of this entry is Physical Mode (bit 11 = 0), bits[59:56] contain an APIC ID. If Logical Mode is selected (bit 11 = 1), the Destination Field potentially defines a set of processors. Bits[63:56] of the Destination Field specify the logical destination address.
55:17	**Reserved**
16	**Interrupt Mask:** When this bit is 1, the interrupt signal is masked. Edge-sensitive interrupts signaled on a masked interrupt pin are ignored (i.e., not delivered or held pending). Level-asserts or negates occurring on a masked level-sensitive pin are also ignored and have no side effects. Changing the mask bit from unmasked to masked after the interrupt is accepted by a local APIC has no effect on that interrupt. This behavior is identical to the case where the device withdraws the interrupt before that interrupt is posted to the processor. It is software's responsibility to handle the case where the mask bit is set after the interrupt message has been accepted by a local APIC unit but before the interrupt is dispensed to the processor. When this bit is 0, the interrupt is not masked. An edge or level on an interrupt pin that is not masked results in the delivery of the interrupt to the destination.
15	**Trigger Mode—R/W:** The trigger mode field indicates the type of signal on the interrupt pin that triggers an interrupt. This bit is set to 1 for level sensitive and 0 for edge sensitive.
14	**Remote IRR—RO:** This bit is used for level triggered interrupts. Its meaning is undefined for edge triggered interrupts. For level triggered interrupts, this bit is set to 1 when local APIC(s) accept the level interrupt sent by the I/O APIC. The Remote IRR bit is set to 0 when an EOI message with a matching interrupt vector is received from a local APIC.
13	**Interrupt Input Pin Polarity (INTPOL)—R/W:** This bit specifies the polarity of the interrupt signal. A 0 selects high active and a 1 selects low active.
12	**Delivery Status (DELIVS)—RO:** The Delivery Status bit contains the current status of the delivery of this interrupt. Delivery Status is read-only and writes to this bit (as part of a 32 bit word) do not effect this bit. When bit 12 = 0 (IDLE), there is currently no activity for this interrupt. When bit 12 = 1 (Send Pending), the interrupt has been injected. However, its delivery is temporarily held up due to the APIC bus being busy or the inability of the receiving APIC unit to accept that interrupt at that time.
11	**Destination Mode (DESTMOD)—R/W:** This field determines the interpretation of the Destination field. When DESTMOD = 0 (physical mode), a destination APIC is identified by its ID. Bits 56 through 59 of the Destination field specify the 4 bit APIC ID. When DESTMOD = 1 (logical mode), destinations are identified by matching on the logical destination under the control of the Destination Format Register and Logical Destination Register in each Local APIC. Bits 56 through 63 (8 MSB) of the Destination field specify the 8 bit APIC ID.

Bit	Description
10:8	**Delivery Mode (DELMOD)—R/W:** The Delivery Mode is a 3 bit field that specifies how the APICs listed in the destination field should act upon reception of this signal. Note that certain Delivery Modes only operate as intended when used in conjunction with a specific trigger Mode. These restrictions are indicated in the following table for each Delivery Mode.

Bits [10:8]	Mode	Description
000	Fixed	Deliver the signal on the INTR signal of all processor cores listed in the destination. Trigger Mode for ‘‘fixed’’ Delivery Mode can be edge or level.
001	Lowest Priority	Deliver the signal on the INTR signal of the processor core that is executing at the lowest priority among all the processors listed in the specified destination. Trigger Mode for ‘‘lowest priority’’. Delivery Mode can be edge or level.
010	SMI	System Management Interrupt. A delivery mode equal to SMI requires an edge trigger mode. The vector information is ignored but must be programmed to all zeroes for future compatibility.
011	Reserved	
100	NMI	Deliver the signal on the NMI signal of all processor cores listed in the destination. Vector information is ignored. NMI must be programmed as edge-triggered for proper operation.
101	INIT	Deliver the signal to all processor cores listed in the destination by asserting the INIT signal. All addressed local APICs will assume their INIT state. INIT must be programmed as edge-triggered for proper operation
110	Reserved	
111	ExtINT	Deliver the signal to the INTR signal of all processor cores listed in the destination as an interrupt that originated in an externally connected (8259A-compatible) interrupt controller. The INTA cycle that corresponds to this ExtINT delivery is routed to the external controller that is expected to supply the vector. A Delivery Mode of ‘‘ExtINT’’ requires an edge trigger mode.

7:0	**Interrupt Vector (INTVEC)—R/W:** The vector field is an 8 bit field containing the interrupt vector for this interrupt. Vector values range from 10 to FEh.

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.
5. Memory addresses for accessing APIC.

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 MByte boundary between memory address 000E0000h–000FFFFFh. The 128 KByte conventional BIOS memory space is split into one 64 KByte region, and four 16 KByte 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 3).

The conventional BIOS is aliased at multiple memory regions. The aliased memory regions are at 16 MByte boundary (High BIOS only), 4 GByte minus 1 MByte boundary, and 4 GByte 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 3).

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 4 GByte 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.

BIOS Location Auto-Detection

Some applications require that Flash-EPROM based BIOS be updated in the system from the data coming from the floppy disk. To support this, the X-bus signals must be properly controlled (i.e. the ESC's X-Bus control logic must be aware of physical BIOS location—X-bus or ISA Bus). This is supported transparently to the software by configuring ESC's X-Bus logic during RESET using the SLOWH# pin.

Logic level on SLOWH# pin is sampled at the end of reset sequence to determine whether BIOS resides on the X-Bus (1) or on the ISA bus (0). This information is used by the ESC to control the X-Bus transceivers during BIOS access.

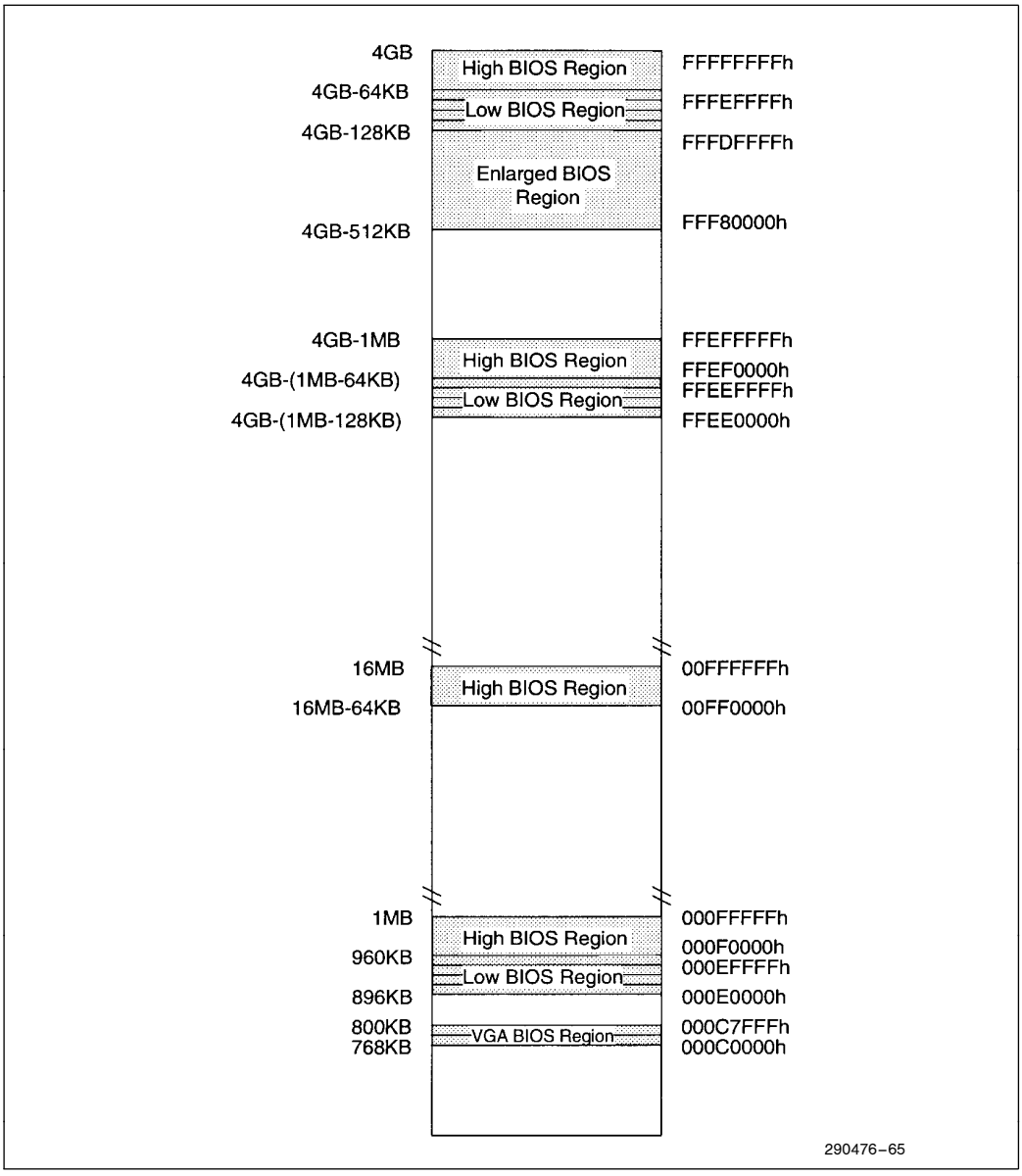


Figure 2. BIOS Memory Map

Table 3. BIOS Chip Select Enable Table

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

Table 3. BIOS Chip Select Enable Table (Continued)

Memory Address Range	Low BIOS En				High BIOS En	ENL BIOS En	Low VGA BIOS En	High VGA BIOS En	16M BIOS En	LBIOSCS# Asserted
	1	2	3	4						
FFFE8000h to	x	x	**0**	x	x	x	x	x	x	No
FFFEBFFFh	x	x	**1**	x	x	x	x	x	x	Yes
FFFEC000h to	x	x	x	**0**	x	x	x	x	x	No
FFFEFFFFh	x	x	x	**1**	x	x	x	x	x	Yes
FFFF0000h to	x	x	x	x	**0**	x	x	x	x	No
FFFFFFFFh	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 documents the I/O address to the ESC internal registers.

Table 4. ESC I/O Register Address Map

Address (Hex)	Address (Bit)				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

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

Address (Hex)	Address (Bit)				Type	Name	Block
	FEDC	BA98	7654	3210			
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	DMA
000Ah	0000	0000	000x	1010	WO	DMA1 Write Single Mask Bit	DMA
000Bh	0000	0000	000x	1011	WO	DMA1 Write Mode	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	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	PIC
0021h	0000	0000	001x	xx01	R/W	INT 1 Mask	PIC
0022h	0000	0000	0010	0010	R/W	Configuration Address Index	CONF
0023h	0000	0000	0010	0011	R/W	Configuration Data Index	CONF
0040h	0000	0000	010x	0000	R/W	Timer 1 Counter 0—System Clock	TC
0041h	0000	0000	010x	0001	R/W	Timer1 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	TC
0048h	0000	0000	010x	1000	R/W	Timer 2 Counter 0—Fail-Safe Timer	TC
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

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

Address (Hex)	Address (Bit)				Type	Name	Block
	FEDC	BA98	7654	3210			
0070H[1]	0000	0000	0111	0xx0	WO	NMI Mask	Control
0080h	0000	0000	100x	0000	R/W	DMA Page Register—Reserved	DMA
0081h	0000	0000	100x	0001	R/W	DMA Channel 2 Page	DMA
0082h	0000	0000	1000	0010	R/W	DMA Channel 3 Page	DMA
0083h	0000	0000	100x	0011	R/W	DMA Channel 1 Page	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	DMA
0088h	0000	0000	100x	1000	R/W	DMA Page Register—Reserved	DMA
0089h	0000	0000	100x	1001	R/W	DMA Channel 6 Page	DMA
008Ah	0000	0000	100x	1010	R/W	DMA Channel 7 Page	DMA
008Bh	0000	0000	100x	1011	R/W	DMA Channel 5 Page	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	DMA
0092h	0000	0000	1001	0010	R/W	System Control Port	Control
00A0h	0000	0000	101x	xx00	R/W	INT 2 Control	PIC
00A1h	0000	0000	101x	xx01	R/W	INT 2 Mask	PIC
00B2h	0000	0000	1011	0010	R/W	Advanced Power Management Control	APM
00B3h	0000	0000	1011	0011	R/W	Advanced Power Management Status	APM
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

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

Address (Hex)	Address (Bit)				Type	Name	Block
	FEDC	BA98	7654	3210			
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	DMA
00D4h	0000	0000	1101	010x	WO	DMA2 Write Single Mask Bit	DMA
00D6h	0000	0000	1101	011x	WO	DMA2 Write Mode	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	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[2]	0000	0011	0111	0010	WO	Secondary Floppy Disk Digital Output	FDCCS#
03F2h[2]	0000	0011	1111	0001	WO	Primary Floppy Disk Digital Output	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

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

Address (Hex)	Address (Bit)				Type	Name	Block
	FEDC	BA98	7654	3210			
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	DMA
040Ch	0000	0100	0000	1100	WO	Chaining Buffer Control	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	DMA
0411h	0000	0100	0010	0001	WO	DMA CH1 S-G Command	DMA
0412h	0000	0100	0010	0010	WO	DMA CH2 S-G Command	DMA
0413h	0000	0100	0010	0011	WO	DMA CH3 S-G Command	DMA
0415h	0000	0100	0010	0101	WO	DMA CH5 S-G Command	DMA
0416h	0000	0100	0010	0110	WO	DMA CH6 S-G Command	DMA
0417h	0000	0100	0010	0111	WO	DMA CH7 S-G Command	DMA
0418h	0000	0100	0010	1000	WO	DMA CH0 S-G Status	DMA
0419h	0000	0100	0010	1001	WO	DMA CH1 S-G Status	DMA
041Ah	0000	0100	0010	1010	WO	DMA CH2 S-G Status	DMA
041Bh	0000	0100	0010	1011	WO	DMA CH3 S-G Status	DMA
041Dh	0000	0100	0010	1101	WO	DMA CH5 S-G Status	DMA
041Eh	0000	0100	0010	1110	WO	DMA CH6 S-G Status	DMA
041Fh	0000	0100	0010	1111	WO	DMA CH7 S-G Status	DMA
0420h	0000	0100	0010	0000	RO	DMA CH0 S-G Descriptor Pointer	DMA
0421h	0000	0100	0010	0001	RO	DMA CH0 S-G Descriptor Pointer	DMA
0422h	0000	0100	0010	0010	RO	DMA CH0 S-G Descriptor Pointer	DMA
0423h	0000	0100	0010	0011	RO	DMA CH0 S-G Descriptor Pointer	DMA
0424h	0000	0100	0010	0100	RO	DMA CH1 S-G Descriptor Pointer	DMA

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

Address (Hex)	Address (Bit)				Type	Name	Block
	FEDC	BA98	7654	3210			
0425H	0000	0100	0010	0101	RO	DMA CH1 S-G Descriptor Pointer	DMA
0426h	0000	0100	0010	0110	RO	DMA CH1 S-G Descriptor Pointer	DMA
0427h	0000	0100	0010	0111	RO	DMA CH1 S-G Descriptor Pointer	DMA
0428h	0000	0100	0010	1000	RO	DMA CH2 S-G Descriptor Pointer	DMA
0429h	0000	0100	0010	1001	RO	DMA CH2 S-G Descriptor Pointer	DMA
042Ah	0000	0100	0010	1010	RO	DMA CH2 S-G Descriptor Pointer	DMA
042Bh	0000	0100	0010	1011	RO	DMA CH2 S-G Descriptor Pointer	DMA
042Ch	0000	0100	0010	1100	RO	DMA CH3 S-G Descriptor Pointer	DMA
042Dh	0000	0100	0010	1101	RO	DMA CH3 S-G Descriptor Pointer	DMA
042Eh	0000	0100	0010	1110	RO	DMA CH3 S-G Descriptor Pointer	DMA
042Fh	0000	0100	0010	1111	RO	DMA CH3 S-G Descriptor Pointer	DMA
0434h	0000	0100	0011	0100	RO	DMA CH5 S-G Descriptor Pointer	DMA
0435h	0000	0100	0011	0101	RO	DMA CH5 S-G Descriptor Pointer	DMA
0436h	0000	0100	0011	0110	RO	DMA CH5 S-G Descriptor Pointer	DMA
0437h	0000	0100	0011	0111	RO	DMA CH5 S-G Descriptor Pointer	DMA
0438h	0000	0100	0011	1000	RO	DMA CH6 S-G Descriptor Pointer	DMA
0439h	0000	0100	0011	1001	RO	DMA CH6 S-G Descriptor Pointer	DMA
043Ah	0000	0100	0011	1010	RO	DMA CH6 S-G Descriptor Pointer	DMA
043Bh	0000	0100	0011	1011	RO	DMA CH6 S-G Descriptor Pointer	DMA

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

Address (Hex)	Address (Bit)				Type	Name	Block
	FEDC	BA98	7654	3210			
043CH	0000	0100	0011	1100	RO	DMA CH7 S-G Descriptor Pointer	DMA
043Dh	0000	0100	0011	1101	RO	DMA CH7 S-G Descriptor Pointer	DMA
043Eh	0000	0100	0011	1110	RO	DMA CH7 S-G Descriptor Pointer	DMA
043Fh	0000	0100	0011	1111	RO	DMA CH7 S-G Descriptor Pointer	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	DMA
0482h	0000	0100	1000	0010	R/W	DMA CH3 High Page	DMA
0483h	0000	0100	1000	0011	R/W	DMA CH1 High Page	DMA
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	DMA
0488h	0000	0100	1000	1000	R/W	Reserved	DMA
0489h	0000	0100	1000	1001	R/W	DMA CH6 High Page	DMA
048Ah	0000	0100	1000	1010	R/W	DMA CH7 High Page	DMA
048Bh	0000	0100	1000	1011	R/W	DMA CH5 High Page	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	DMA
04C2h	0000	0100	1100	0010	R/W	Reserved	DMA

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

Address (Hex)	Address (Bit)				Type	Name	Block
	FEDC	BA98	7654	3210			
04C6H	0000	0100	1100	0110	R/W	DMA CH5 High Base & Current Count	DMA
04CAh	0000	0100	1100	1010	R/W	DMA CH6 High Base & Current Count	DMA
04CEh	0000	0100	1100	1110	R/W	DMA CH7 High Base & Current Count	DMA
04D0h	0000	0100	1101	0000	R/W	INT-1 Edge/Level Control	PIC
04D1h	0000	0100	1101	0001	R/W	INT-2 Edge/Level Control	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	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	DNA
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

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

Address (Hex)	Address (Bit)				Type	Name	Block
	FEDC	BA98	7654	3210			
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
04EC	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

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

Address (Hex)	Address (Bit)				Type	Name	Block
	FEDC	BA98	7654	3210			
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:
1. Port 70h resides in the ESC in addition the lower 7 bits of Port 70h reside in Real Time Clock also.
2. 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 5. Configuration Register Index Address

Configuration Offset	Abbreviation	Register Name
00–01h	—	Reserved
02h	ESCID	ESC ID
03–07h	—	Reserved
08h	RID	Revision ID
09–3Fh	—	Reserved

Table 5. Configuration Register Index Address (Continued)

Configuration Offset	Abbreviation	Register Name
40h	MS	Mode Select
41h	—	Reserved
42h	BIOSCSA	BIOS Chip Select A
43h	BIOSCSB	BIOS Chip Select B
44–4Ch	—	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
54–56h	—	Reserved
57h	SGRBA	Scatter-Gather Relocate Base Address
58h	—	Reserved
59h	APICBA	APIC Base Address Relocation
60h	PIRQRC0	PIRQ0# Route Control
61h	PIRQRC1	PIRQ1# Route Control
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

Table 5. Configuration Register Index Address (Continued)

Configuration Offset	Abbreviation	Register Name
6Eh	GPCSM2	General Purpose Chip Select 2 Mask
6Fh	GPXBC	General Purpose Peripheral X-Bus Control
70h	PACC	PIC/APIC Configuration Control
71–87h	—	Reserved
88h	TSTC	Test Control
89–9Fh	—	Reserved
A0h	SMICNTL	SMI Control
A2-A3h	SMIEN	SMI Enable
A4-A7h	SEE	System Event Enable
A8h	FTMR	Fast Off Timer
A9h	—	Reserved
AA-ABh	SMIREQ	SMI Request
ACh	CTLTMRL	Clock Scaling STPCLK# Low Timer
ADh	—	Reserved
AEh	CTLTMRH	Clock Scaling STPCLK# High Timer
AF-FFh	—	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 select signals for the 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 6. X-Bus Chip Selects Decode

Address (Hex)	Address (Bit)				R/W	Name	Chip Select
	FEDC	BA98	7654	3210			
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–0177h	0000	0001	0111	0xxx	R/W	IDE Controller 0-Secondary	ECS[2:0] = 011 (IDECS0#)
01F0h–01F7h	0000	0001	1111	0xxx	R/W	IDE Controller 0-Primary	ECS[2:0] = 011 (IDECS0#)
0278h–027Bh	0000	0010	0111	1000 to 1011	R/W	Parallel Port LPT3	ECS[2:0] = 010 (LPTCS#)
02F8h–02FFh	0000	0010	1111	xxxx	R/W	Serial Port COM2	ECS[2:0] = 00x (COMxCS#)
0370h–0375h	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–037Bh	0000	0011	0111	1000 to 1011	R/W	Parallel Port LPT2	ECS[2:0] = 010 (LPTCS#)
03BCh–03BFh	0000	0011	1011	11xx	R/W	Parallel Port LPT 1	ECS[2:0] = 010 (LPTCS#)
03F0h–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#)

Table 6. X-Bus Chip Selects Decode (Continued)

Address (Hex)	Address (Bit)				R/W	Name	Chip Select
	FEDC	BA98	7654	3210			
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-Primary	FDCCS#
03F8h–03FFh	0000	0011	1111	1000	R/W	Serial Port COM 1	ECS[2:0] = 00x (COMxCS#)
0800h–08FFh	0000	1000	xxxx	xxxx	W/R	Configuration RAM	CRAMWR#/ CRAMRD#

4.5 I/O APIC Registers

The APIC's registers are indirectly address through two 32 bit registers located in the CPU's memory space—the I/O Register Select (IOREGSEL) and I/O Window (IOWIN) Registers (Table 7). These registers can be relocated via the APIC Base Address Relocation Register and are aligned on 128 bit boundaries.

To access an I/O APIC register, the IOREGSEL Register is written with the address of the intended APIC register. Bits[7:0] of the IOREGSEL Register provide the address offset (Table 8). The IOWIN Register then becomes a 32-bit window pointing to the register selected by the IOREGSEL Register. Note that, for each redirection table register, there are two offset addresses (e.g., address offset 10h selects IOREDTBL0 bits[31:0] and 11h selects IOREDTBL0 bits[63:32].

Table 7. Memory Address For Accessing APIC Registers

Memory Address	Mnemonic	Register Name	Access
FEC0 x000h (82374EB)	IOREGSEL	I/O Register Select	R/W
FEC0 xy00h (82374SB)			
FEC0 x010h (82374EB)	IOWIN	I/O Window	R/W
FEC0 xy10h (82374SB)			

NOTE:
xy are determined by the x and y (82374SB only) fields in the APIC Base Address Relocation Register. Range for x = 0-Fh and the range for y = 0,4,8,Ch.

Table 8. I/O APIC Registers

Memory Address	Mnemonic	Register Name	Access
00h	IOAPICID	I/O APIC ID	R/W
01h	IOAPICVER	I/O APIC Version	RO
02h	IOAPICARB	I/O APIC Arbitration ID	RO
10-2Fh	IOREDTBL[0:15]	Redirection Table (Entries 0-15) (63 bits each)	R/W

NOTE:
Address Offset is determined by I/O Register Select Bits[7:0]

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. These 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 is 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 protect 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 9. 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# minimum 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.

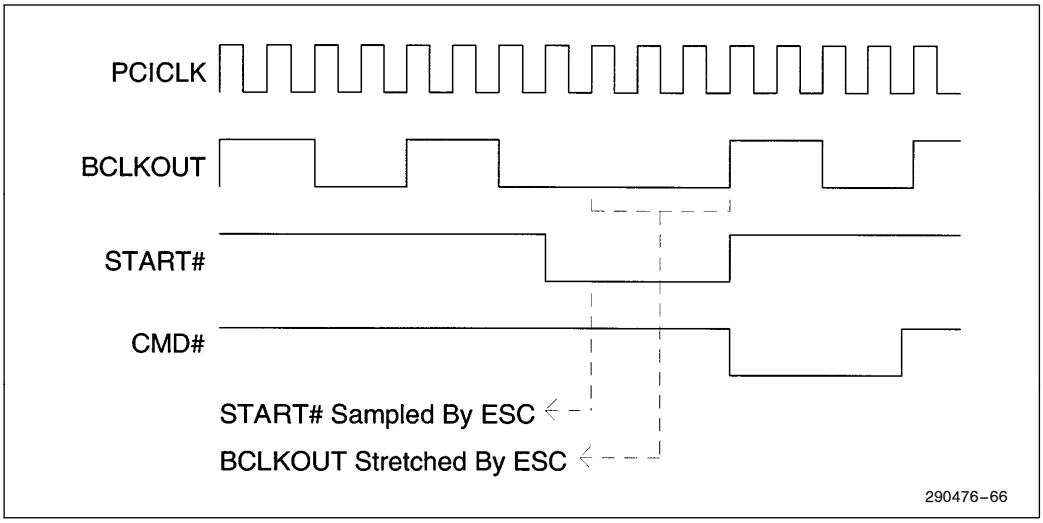


Figure 3. BCLK Stretching

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 Section 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-BIT EISA SLAVE

An EISA slave after decoding it's 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 specification 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 it's 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 resource 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 the 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#.

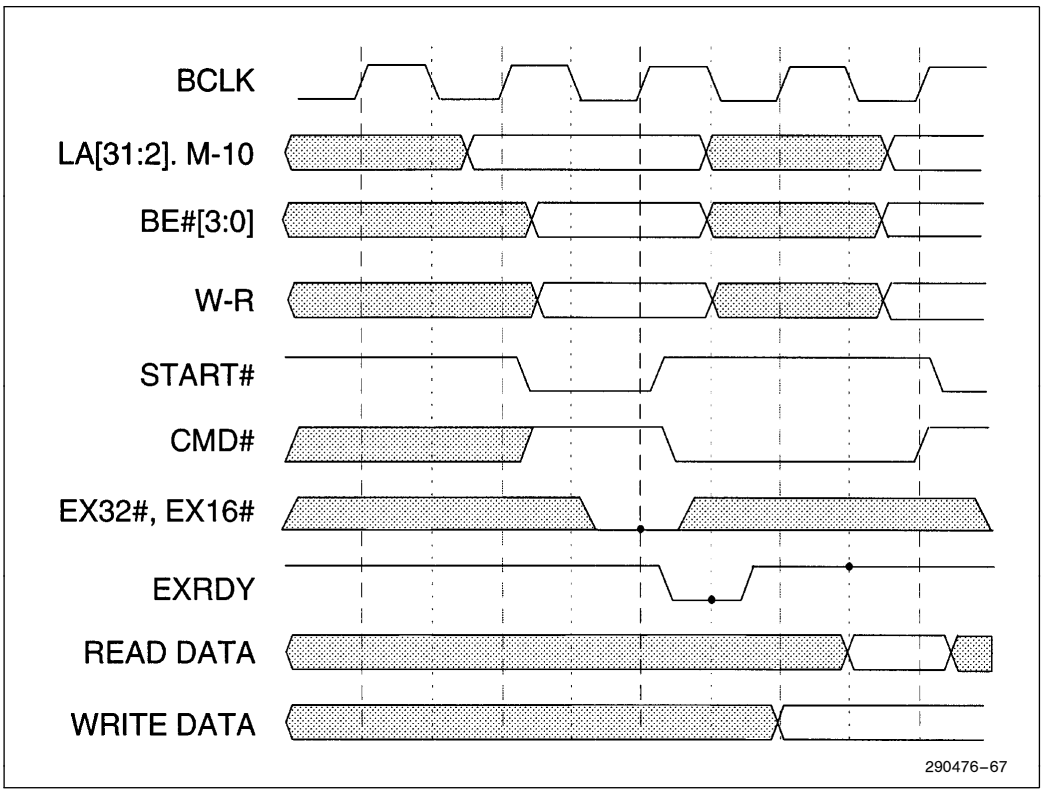


Figure 4. Standard EISA Master to EISA Slave Cycle

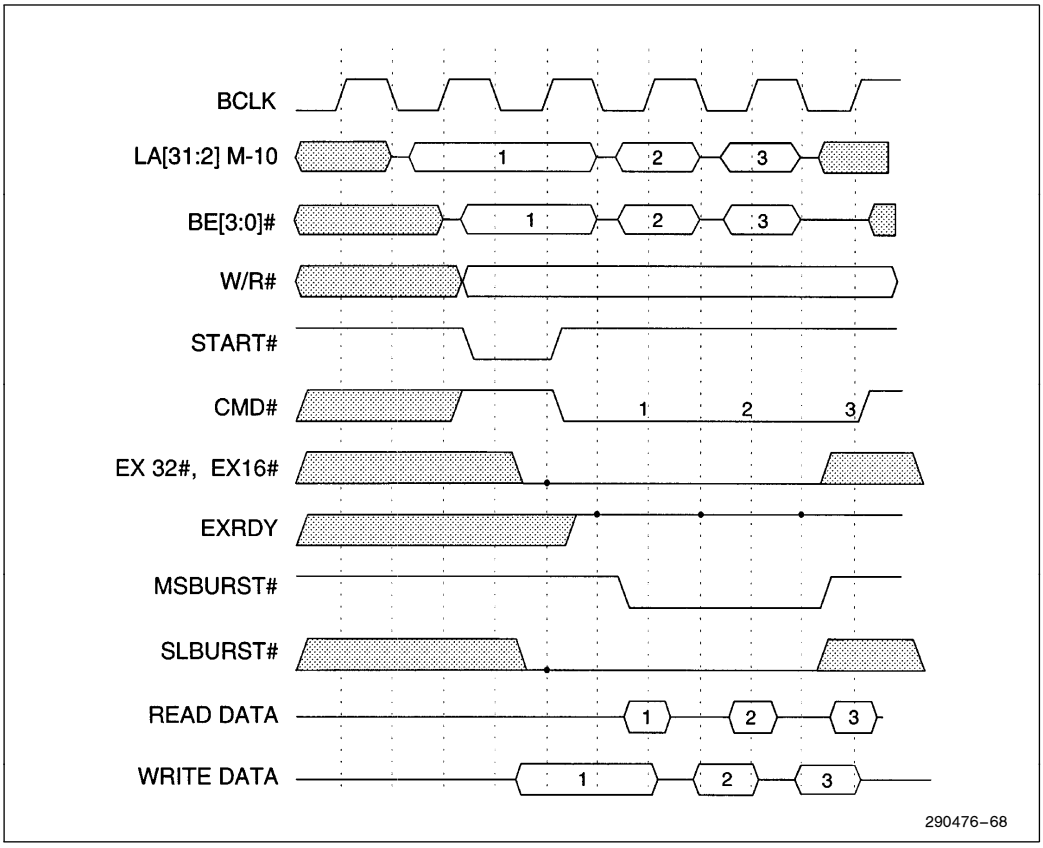


Figure 5. Burst EISA Master to EISA Slave Cycle

5.3.2 EISA MASTER TO 16-BIT ISA SLAVE

An ISA slave, after decoding it's 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 it's 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 performs translation cycle. The master must back-off the bus if a master/slave data size mis-match is determined, regardless if data size translation 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.

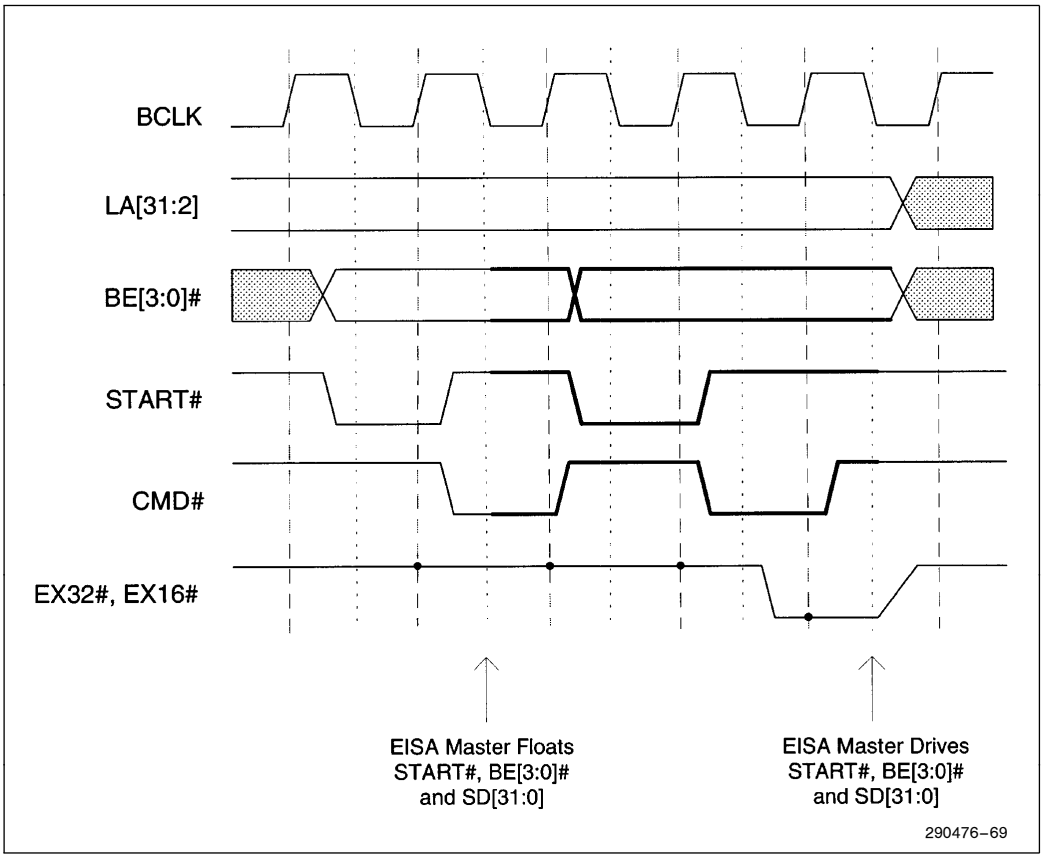


Figure 6. EISA Master Back Off Cycle

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-/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 mismatched 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- 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.

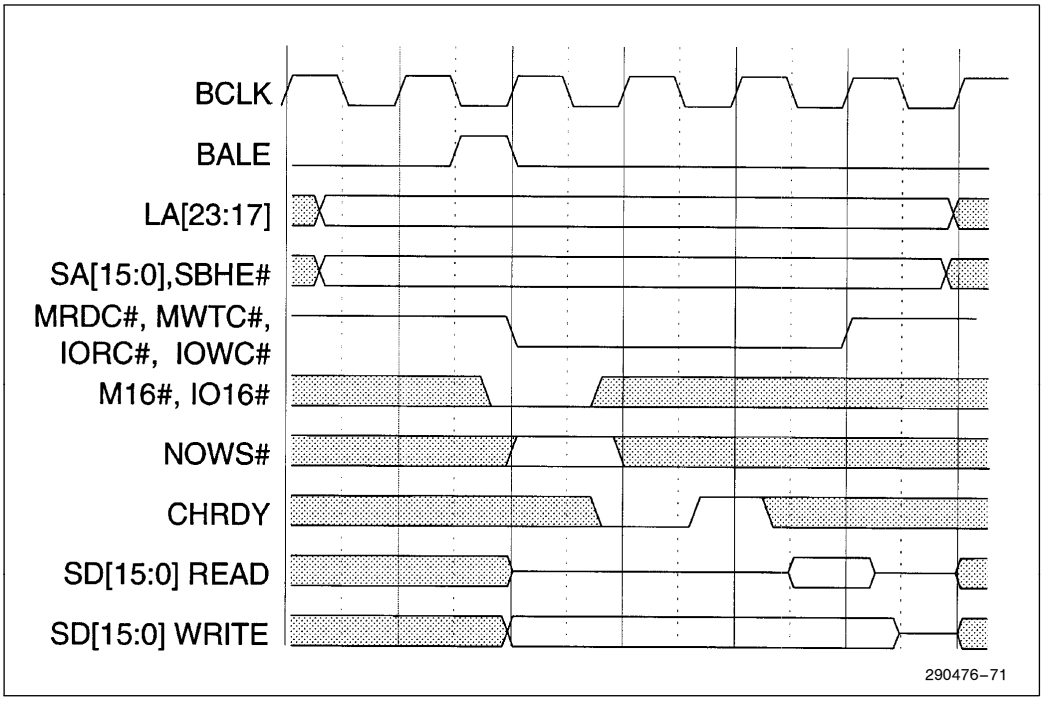


Figure 7. ISA Master to 16-Bit ISA Slave Cycles (3 BCLKs)

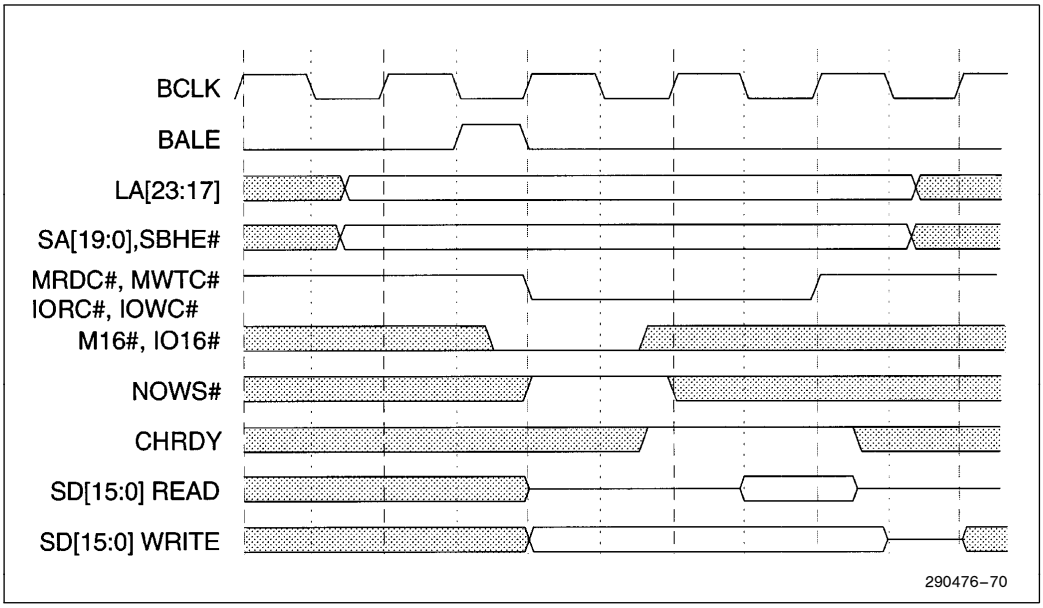


Figure 8. ISA Master to 16-Bit ISA Slave Extended Cycle (4 BCLKs)

5.4.3 ISA MASTER TO 8-BIT EISA/ISA SLAVE

An 8-bit slave does not positively acknowledge it's 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 pull-up 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- 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 10 shows the number of BCLKs for each cycle type (Memory, I/O, DMA), default, wait states added and with NOWS# asserted.

Table 10. 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(1)	8/16	NA	6	7
DMA Type B(1)	8/16	NA	4	5
DMA Type C(2)	8/16	NA	2	3

NOTES:
1. If ISA memory responds, the ESC will extend the cycle by 1 BCLK.
2. 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 11. Mis-Match Master Slave Combinations

Master Type	Cycle 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 Section 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 in to 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 transfer 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 transfer 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 to 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 12). 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 12. AEN Generation

Cycle	A[15:12]	A[11:8]	A[7:4]	A[3:0]	AEN4	AEN3	AEN2	AEN1
DMA	xxxx	xxxx	xxxx	xxxx	1	1	1	1
IO	0000	xx00	xxxx	xxxx	1	1	1	1
IO	0001	xx00	xxxx	xxxx	1	1	1	0
IO	0010	xx00	xxxx	xxxx	1	1	0	1
IO	0011	xx00	xxxx	xxxx	1	0	1	1
IO	0100	xx00	xxxx	xxxx	0	1	1	1
IO	0101-1111	xx00	xxxx	xxxx	1	1	1	1
IO	xxxx	xx01	xxxx	xxxx	0	0	0	0
IO	xxxx	xx10	xxxx	xxxx	0	0	0	0
IO	xxxx	xx11	xxxx	xxxx	0	0	0	0
MEM	xxxx	xxxx	xxxx	xxxx	0	0	0	0

Table 13. Encoded AEN (AEN) Generation

Cycle	A[15:12]	A[11:8]	A[7:4]	A[3:0]	EAEN4	EAEN3	EAEN2	EAEN1
DMA	xxxx	xxxx	xxxx	xxxx	1	1	1	1
IO	0000	xx00	xxxx	xxxx	1	1	1	1
IO	0001	xx00	xxxx	xxxx	0	0	0	1
IO	0010	xx00	xxxx	xxxx	0	0	1	0
IO	0011	xx00	xxxx	xxxx	0	0	1	1
IO	0100	xx00	xxxx	xxxx	0	1	0	0
IO	0101	xx00	xxxx	xxxx	0	1	0	1
IO	0110	xx00	xxxx	xxxx	0	1	1	0
IO	0111	xx00	xxxx	xxxx	0	1	1	1
IO	1000	xx00	xxxx	xxxx	1	0	0	0
IO	1001-1111	xx00	xxxx	xxxx	1	1	1	1
IO	xxxx	xx01	xxxx	xxxx	0	0	0	0
IO	xxxx	xx10	xxxx	xxxx	0	0	0	0
IO	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 for to directly support 4 masters through 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 14. 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’’.

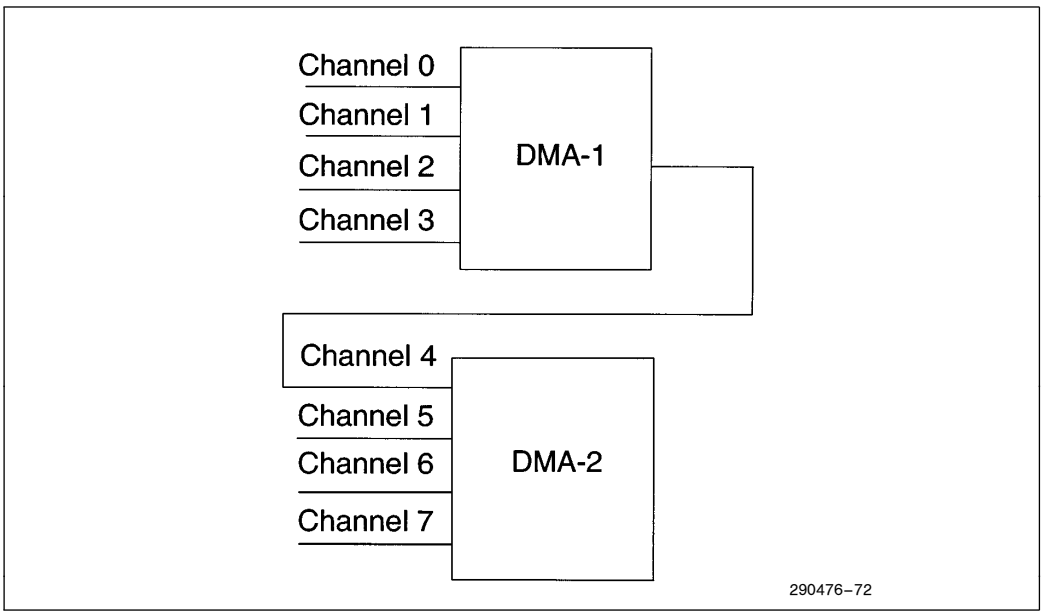


Figure 9. Internal DMA Controller

Each DMA channel can be programmed for 8- 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 I/O 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 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 Section 7.0 on the EISA Bus Arbitration for a detailed description of the 4 μs timeout mechanism. Note that scatter-gather block mode is not supported.

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 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.

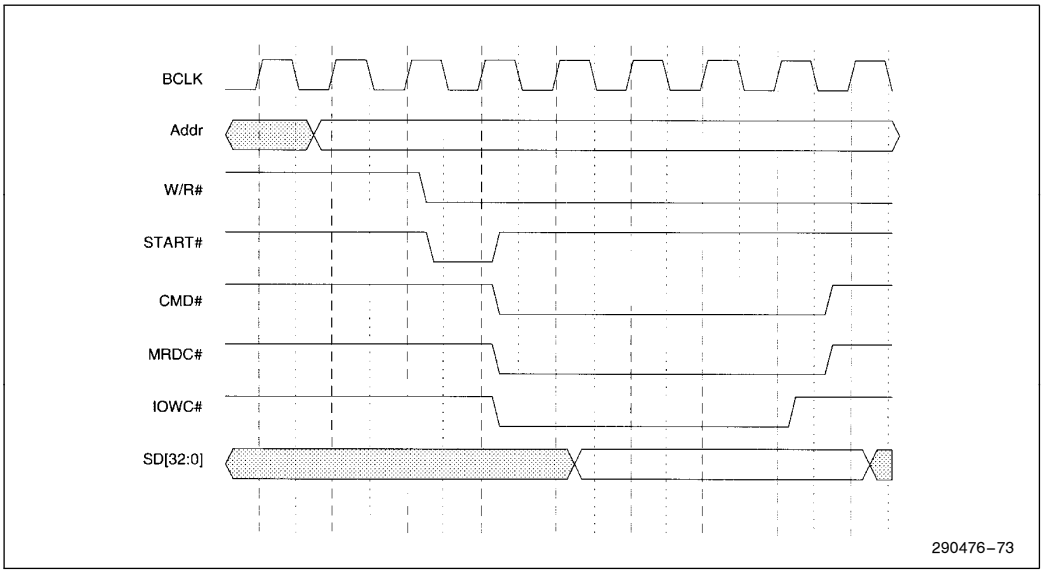


Figure 10. Compatible DMA Read Transfer (8 BCLKs)

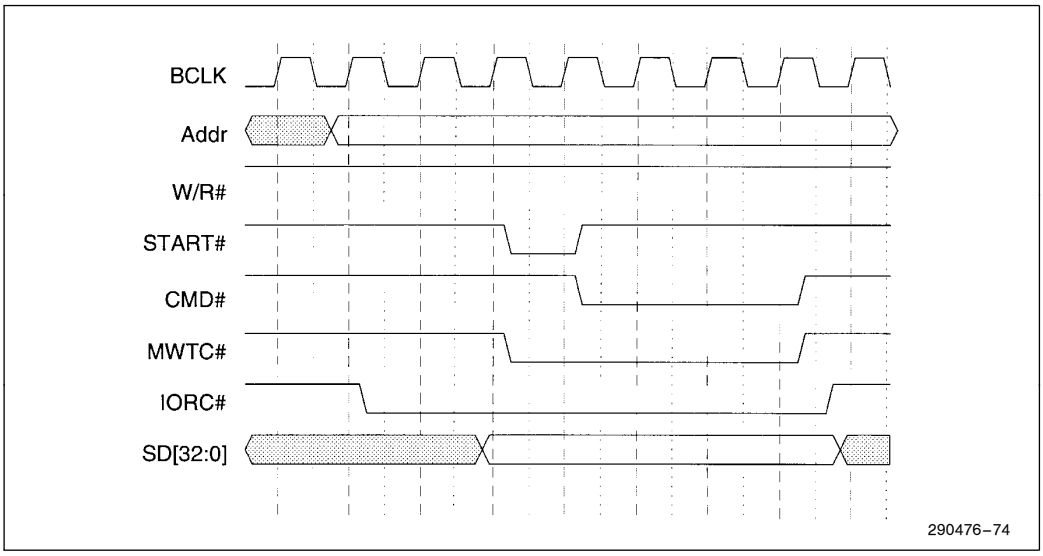


Figure 11. Compatible DMA Write Transfer (8 BCLKS)

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.

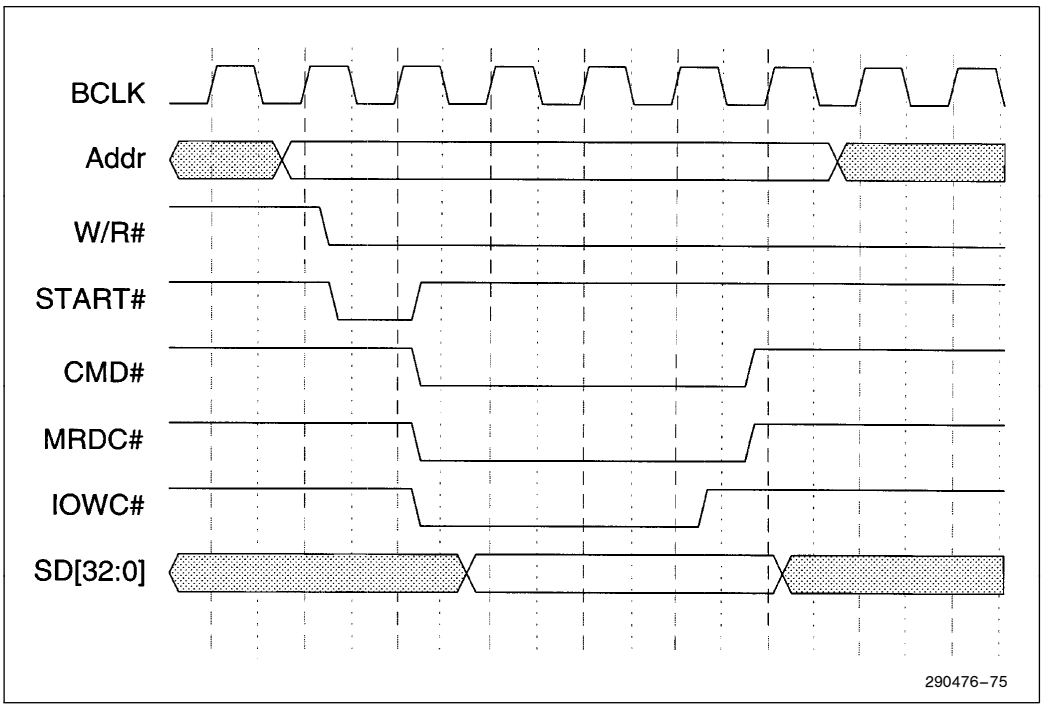


Figure 12. Type "A" DMA Read Transfers (6 BCLKS)

6.4.3 TYPE "B" TIMING

Type "B" timing is provided for 8-/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.

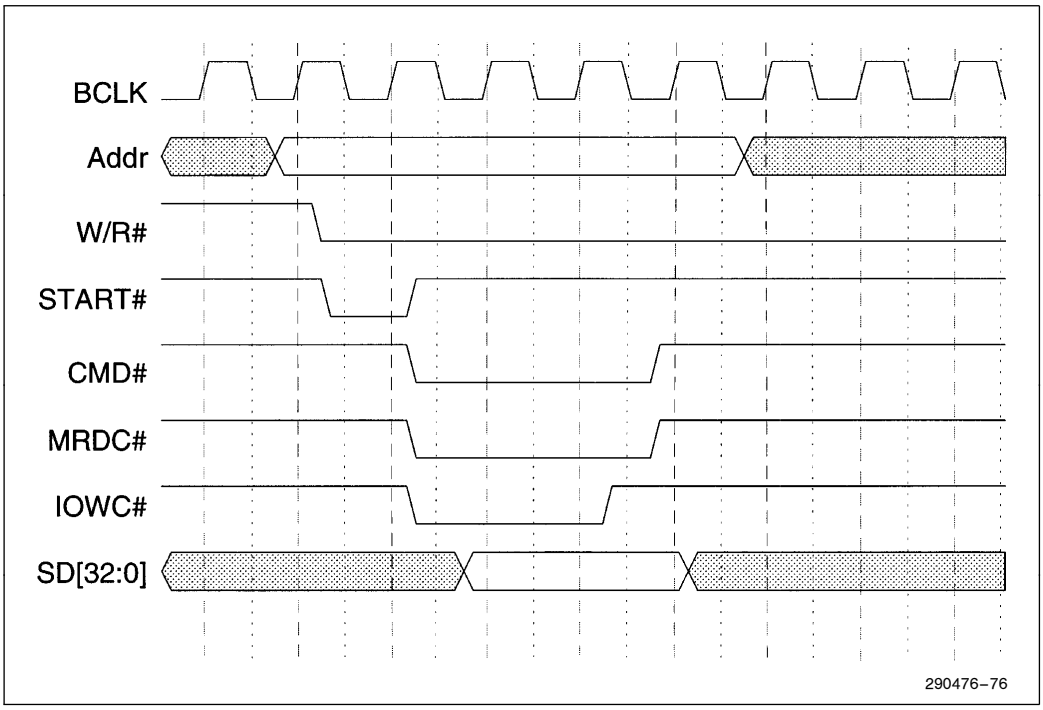


Figure 13. Type "B" DMA Read Transfer (4 BCLKS)

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- 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.

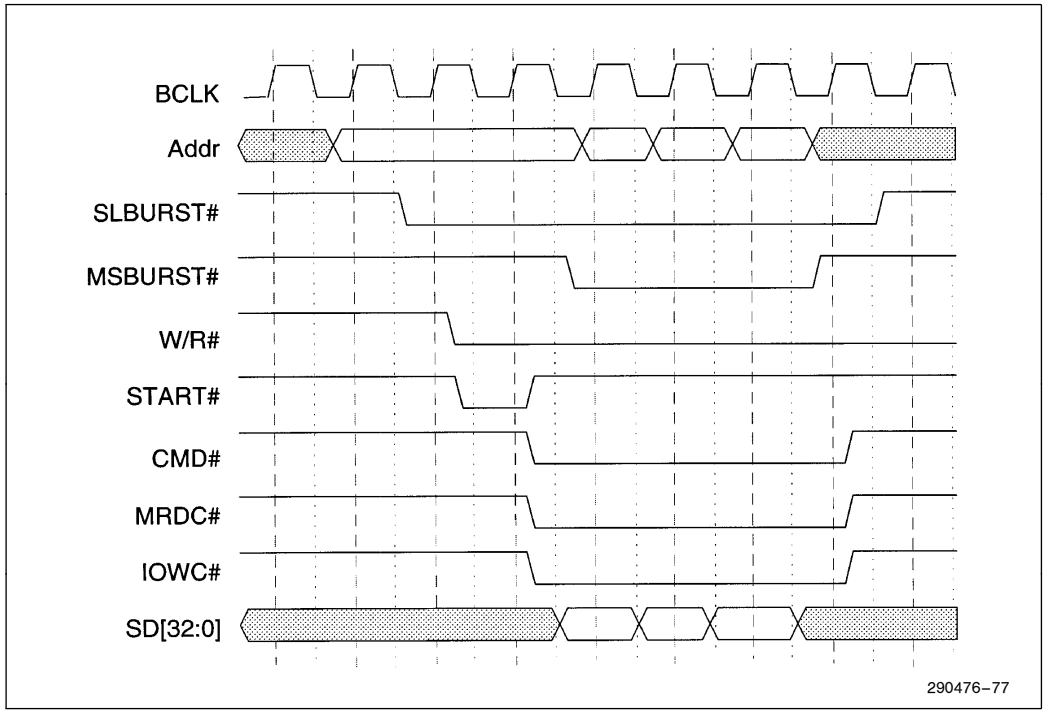


Figure 14. Type "C" (Burst) DMA Read Transfers (1 BCLK)

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 3.0 for Request Register programming information.

Fixed Priority

The initial fixed priority structure is as follows:

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 partners in the rotation, while Channel group (0-3) comprises the fourth position in the arbitration. Table 15 demonstrates rotation priority:

Table 15. 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	4) After servicing channel 6	7, (1, 2, 3, 0), 5, 6
channel 0 caused double rotation).	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 Descriptor 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 Descriptor 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 an I/O device the following steps are involved:

1. 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 consecutive SGDs are offset by 8 bytes and are aligned on a 4-byte boundary.
2. Each Scatter-Gather Descriptor for the linked list must contain 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)

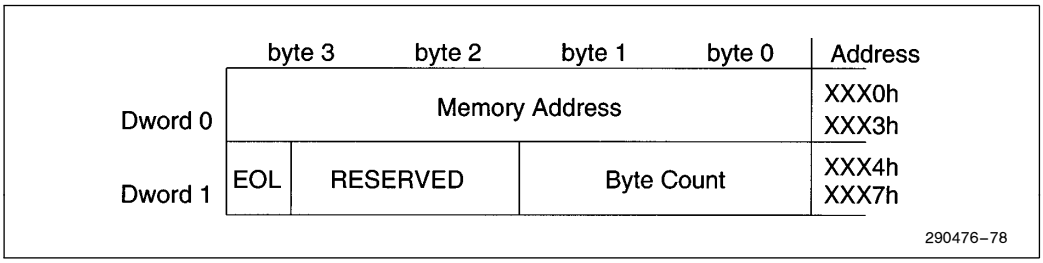


Figure 15. Scatter-Gather Descriptor Format

3. Initialize DMA Mode and Extended Mode registers with transfer specific information like 8-/16-bit I/O device, Transfer Mode, Transfer Type, etc.
4. Software provides the starting address of the Scatter-Gather Descriptor Table by loading the Scatter-Gather Descriptor Table Pointer register.
5. Engage the Scatter-Gather machine by writing a Start command to the Scatter-Gather Command register.
6. The Mask register should be cleared as last the last step of programming the DMA register set. This is to prevent DMA from starting a transfer with a partially loaded command description.
7. 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.
8. 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.
9. 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.

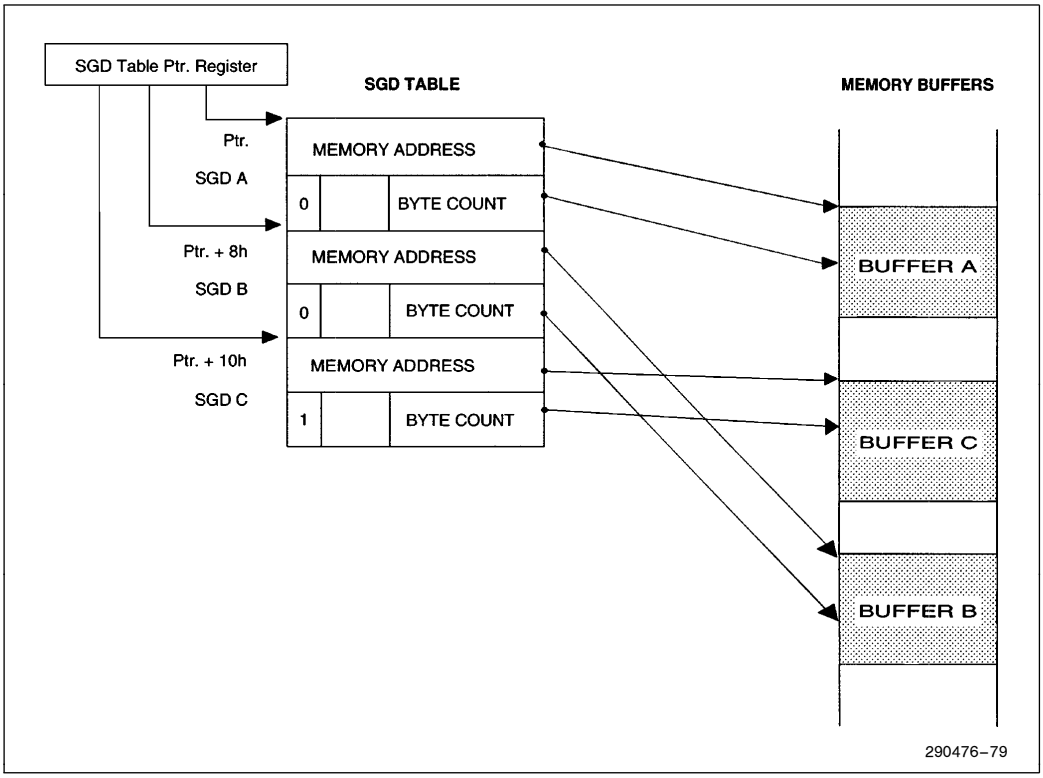


Figure 16. Link List Example

6.7 Register Functionality

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 take 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 16 lists each of the DMA device transfer sizes. The column labeled ''Word Count Register'' indicates that the register contents represent 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 16. 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 PC/AT 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 count6.7.3-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 as shown in Table 17.

Table 17. 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 [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.

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 8-bit and two 8-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 Dword 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.

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 Stop 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 are shown in Table 18.

Table 18. Stop Register Functionality Example

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

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

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 Autoinitialization, 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. Clear Byte Pointer Flip-Flop.
2. Master Clear.
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

Table 19 is a summary of the events that happen as a result of a terminal count or external EOP when running DMA in various modes.

Table 19. 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:
1. load = Load Current From Base
2. "-" = No Change
3. X = Don't Care
4. 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 to 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 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 determine the winner. If the new winner is an EISA Master or DMA channel, the ESC will assert NMFLUSH#. The ESC tri-states 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 tri-state the output buffer. After sampling NMFLUSH# negated, the ESC resumes driving NMFLUSH# on the next PCI clock. This way the ESC does not assert MACK# or DACK# until the PCEB acknowledges that all line buffers have been flushed.

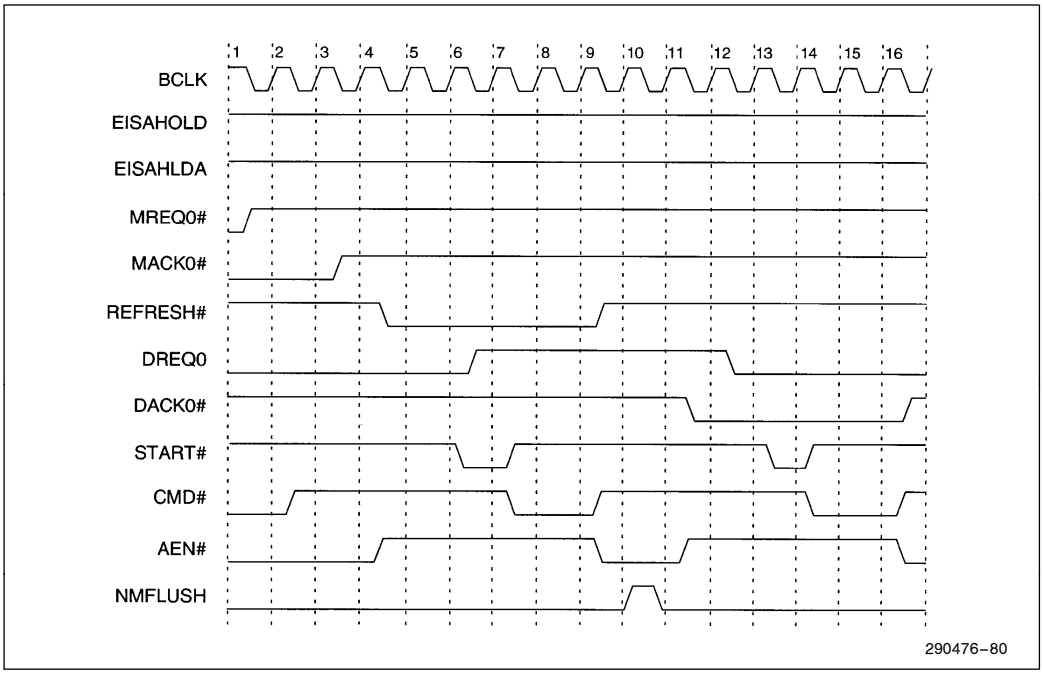


Figure 17. EISA Arbitration

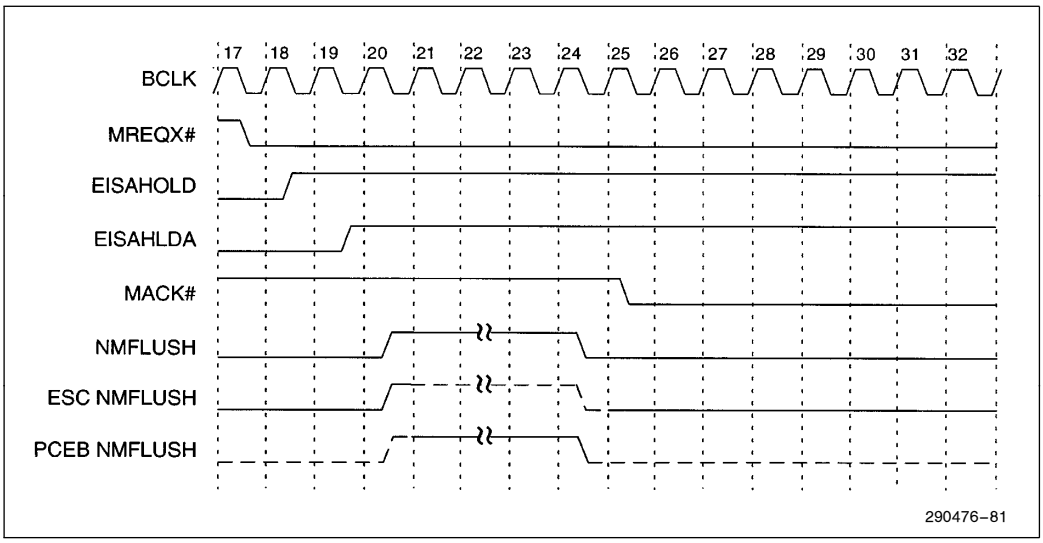


Figure 18. PCEB Preemption

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 20. 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	Refresh Request
Clock In	Port 61h	
Out	1.193 MHz(OSC/12)	8 MHz (BCLK)
	Speaker	CPU Speed Control (SLOWH#)

8.1 Interval Timer Address Map

Table 21 shows the I/O address map of the interval timer counters:

Table 21. 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-save 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 0461h. 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. Table 22 lists the six operating modes for the interval counters. Note that for the fail safe timer (timer 2, counter 0), only mode 0 is supported.

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

1. The Control Word Command specifies:
   — which counter to read or write
   — the operating mode
   — the count format (binary or BCD)
2. The Counter Latch Command latches the current count so that it can be read by the system. The count-down process continues.
3. 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.

Table 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 come 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 effected 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 effecting 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 effect 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] (Figure 19). 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.

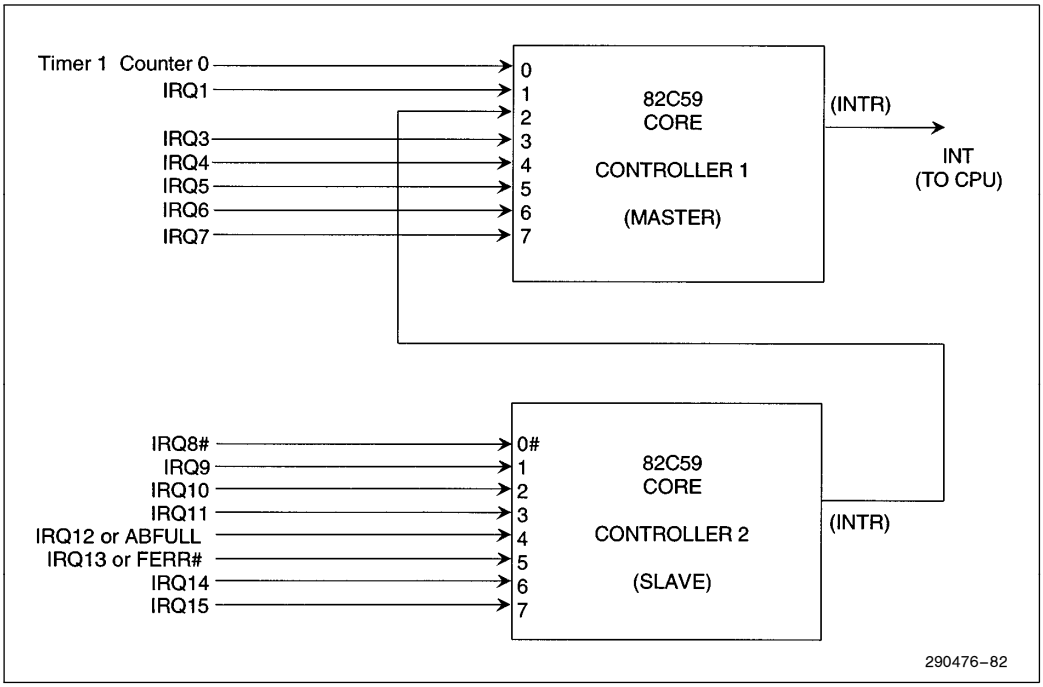


Figure 19. Block Diagram of the Interrupt Controller

Table 23 lists the I/O port address map for the interrupt registers.

Table 23. 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 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	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 effect 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 INTR to the CPU, if appropriate.
3. The CPU acknowledges the INTR 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.

9.3.1 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 25. Content of Interrupt Vector Byte for 80x86 System Mode

Interrupt	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 Register Description section).

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 25.

9.4 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:

a. 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.

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 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 ESC interrupt unit, IRQ2 on CNTRL-1 is used to cascade the INTR output of CNTRL-2. Consequently, bit 2 of ICW3 on CNTRL-1 is set to a 1, and the other bits are set to 0's.

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

ICW4 must be programmed on both controllers. At the very least, bit 0 must be set to a 1 to indicate that the controllers are operating in an 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 26 shows an example of typical values programmed by the BIOS at power-up for the ESC interrupt controller.

Table 26. 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, 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	CNTLR-2, ICW4 8086 Mode
4D1h	00h	CNTLR-2, Edge/Level Control Register
0A1h	BDh	CNTLR-2, Interrupt Mask (may vary)

Figure 20 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. As shown in Figure 20, all ICWs must be programmed prior to programming the OCWs.

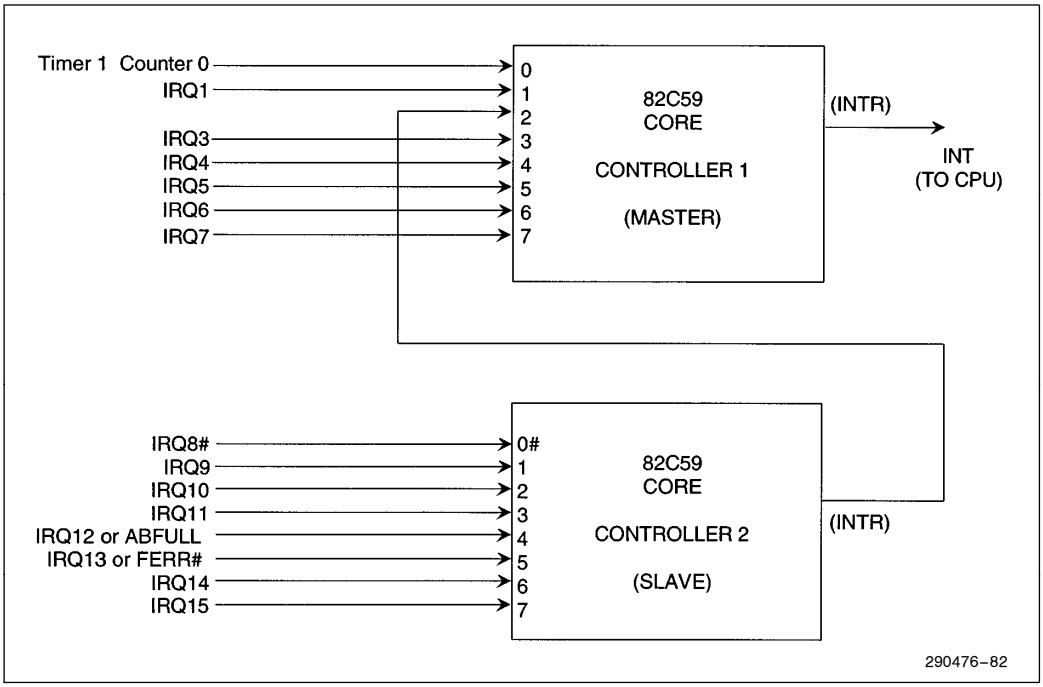


Figure 20. Initialization Sequence for ESC Initialization Command Words (ICWs)

9.5 End-Of-Interrupt Operation

9.5.1 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 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 L0-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.

9.5.2 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).

9.6 Modes Of Operation

9.6.1 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.

9.6.2 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:

1. 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.)
2. 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.

9.6.3 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 21 shows an example of automatic rotation.

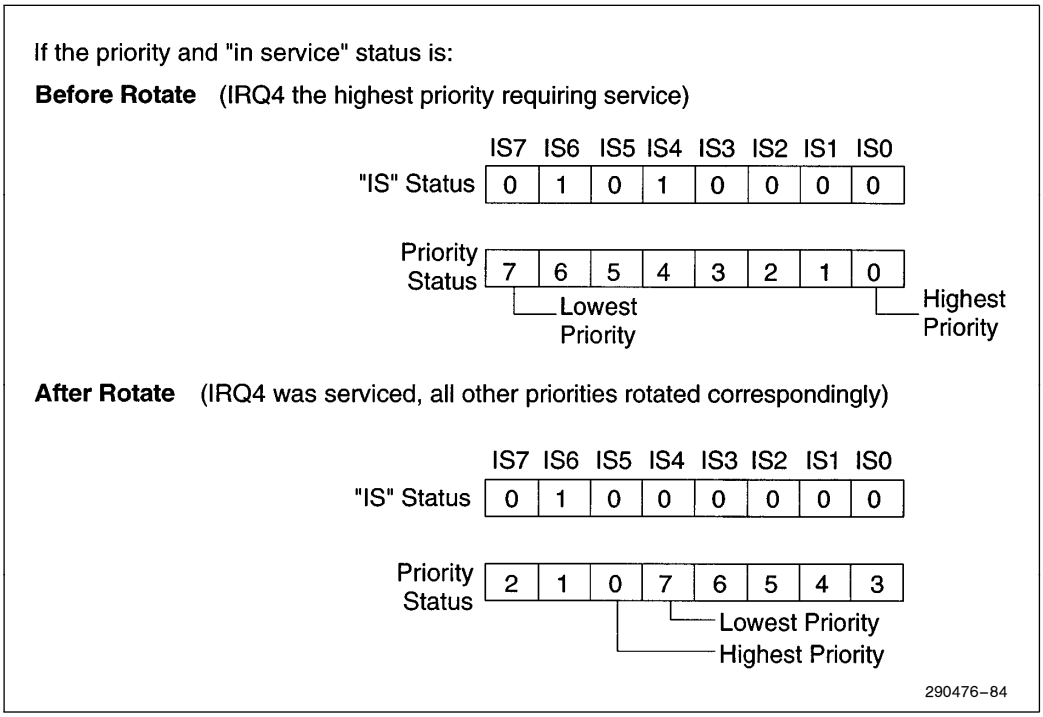


Figure 21. Automatic Rotation Mode Example

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).

9.6.4 SPECIFIC ROTATION (SPECIFIC PRIORITY)

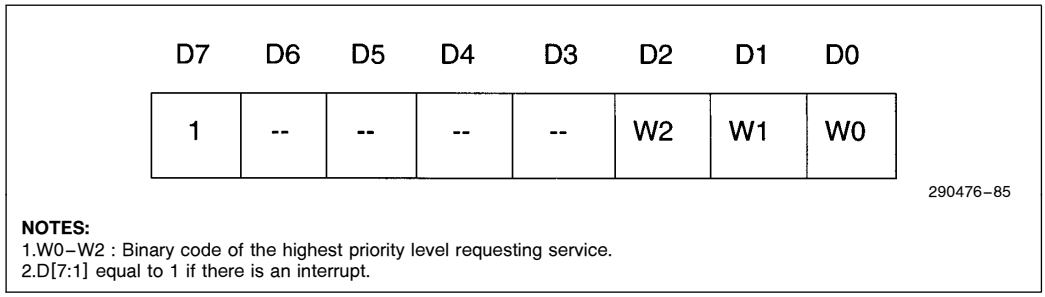
The programmer can change priorities by programming the bottom priority and thus fixing all other priorities. For example, if IRQ5 is programmed as the bottom priority device, then IRQ6 will be the highest priority device. The Set Priority command is issued in OCW2 where: R = 1, SL = 1; L0–L2 is the 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 L0–L2 = IRQ level to receive bottom priority).

9.6.5 POLL COMMAND

The Polled Mode can be used to conserve space in the interrupt vector table. Multiple interrupts that can be serviced by one interrupt service routine do not need separate vectors if the service routine uses the poll command. The Polled Mode can also be used to expand the number of interrupts. The polling interrupt service routine can call the appropriate service routine, instead of providing the interrupt vectors in the vector Table. In this mode the INTR output is not used and the microprocessor internal Interrupt Enable flip-flop is reset, disabling its interrupt input. Service to devices is achieved by software using a Poll command.

The Poll command is issued by setting P = 1 in OCW3. The Interrupt Controller treats the next I/O read pulse to the Interrupt Controller as an interrupt acknowledge, sets the appropriate IS bit if there is a request, and reads the priority level. Interrupts are frozen from the I/O write to the I/O read. The word enabled onto the data bus during I/O read is shown in Figure 22.



NOTES:
1.W0–W2 : Binary code of the highest priority level requesting service.
2.D[7:1] equal to 1 if there is an interrupt.

Figure 22. 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).

9.6.6 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.

9.6.7 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.

9.7 Register Functionality

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

9.7.1 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.

9.7.2 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).

9.8 Interrupt Masks

9.8.1 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 effect 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 INTR output is physically connected to the CNTRL-1 IRQ[2] input.

9.8.2 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.

9.9 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 effected 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).

9.10 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. **Motherboard Parity Errors:** Memory parity errors for the motherboard memory. These errors are reported to the ESC through the PERR# signal line.
2. **System Errors:** System error on the motherboard. 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. See Section 3.4, Interrupt Controller Registers, for the register details.

Table 27. 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 28. NMI Source Enable/Disable And Status Port Bits

NMI Source	IO Port Bit for Status Reads	IO 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 ADVANCED PROGRAMMABLE INTERRUPT CONTROLLER (APIC)

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

In a uni-processor system, APIC's dedicated interrupt bus can reduce interrupt latency over the standard interrupt controller (i.e., the latency associated with the propagation of the interrupt acknowledge cycle across multiple busses using the standard interrupt controller approach). Interrupts can be controlled by the standard EISA compatible interrupt controller unit, the I/O APIC unit, or mixed mode where both the standard and I/O APIC are used. The selection of which controller responds to an interrupt is determined by how the interrupt controllers are programmed. Note that it is the programmer's responsibility to make sure that the same interrupt input signal is not handled by both interrupt controllers.

At the system level, APIC consists of two parts (Figure 23)—one residing in the I/O subsystem (called the I/O APIC) and the other in the CPU (called the Local APIC). The ESC contains an I/O APIC unit. The local APIC and the I/O APIC communicate over a dedicated APIC bus. The ESC's I/O APIC bus interface consists of two bi-directional data signals (APICD[1:0]) and a clock input (APICCLK).

The CPU's Local APIC Unit contains the necessary intelligence to determine whether or not its processor should accept interrupts broadcast on the APIC bus. The Local Unit also provides local pending of interrupts, nesting and masking of interrupts, and handles all interactions with its local processor (e.g., the INTR/INTA/EOI protocol). The Local Unit further provides inter-processor interrupts and a timer, to its local processor. The register level interface of a processor to its local APIC is identical for every processor.

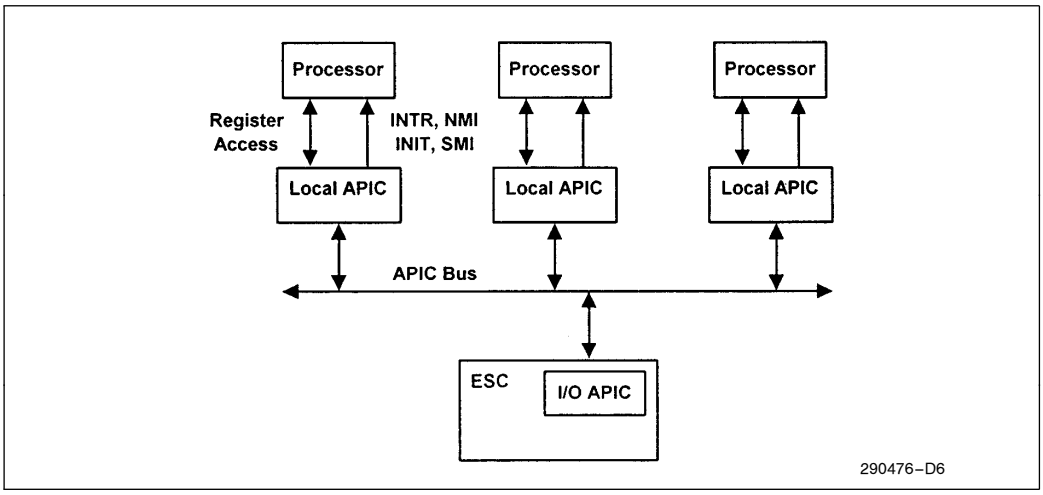


Figure 23. APIC System Structure

The ESC's I/O APIC Unit consists of a set of interrupt input signals, a 16-entry Interrupt Redirection Table, programmable registers, and a message unit for sending and receiving APIC messages over the APIC bus (Figure 24). I/O devices inject interrupts into the system by asserting one of the interrupt lines to the I/O APIC (Figure 25). The I/O APIC selects the corresponding entry in the Redirection Table and uses the information in that entry to format an interrupt request message. Each entry in the Redirection Table can be individually programmed to indicate edge/level sensitive interrupt signals, the interrupt vector and priority, the destination processor, and how the processor is selected (statically or dynamically). The information in the table is used to transmit a message to other APIC units (via the APIC bus).

The ESC's I/O APIC contains a set of programmable registers. Two of the registers (I/O Register Select and I/O Window Registers) are located in the CPU's memory space and are used to indirectly access the other APIC registers as described in Section 3.0, Register Description. The Version Register provides the implementation version of the I/O APIC. The I/O APIC ID Register is programmed with an ID value that serves as a physical name of the I/O APIC. This ID is loaded into the ARB ID Register when the I/O APIC ID Register is written and is used during bus arbitration.

NOTE:

1. When the ESC's I/O APIC receives an interrupt request, the ESC instructs the PCEB to flush its buffers and to request all system buffers pointing to PCI to be flushed (via the AFLUSH# signal). The APIC does not send the interrupt message over the APIC bus until the ESC receives confirmation from the PCEB (via the AFLUSH# signal) that all buffers have been flushed and temporarily disabled.
2. The interrupt number or the vector does not imply a particular priority for being sent. The I/O APIC continually polls the 16 interrupts in a rotating fashion, one at a time. The pending interrupt polled first is the one sent.

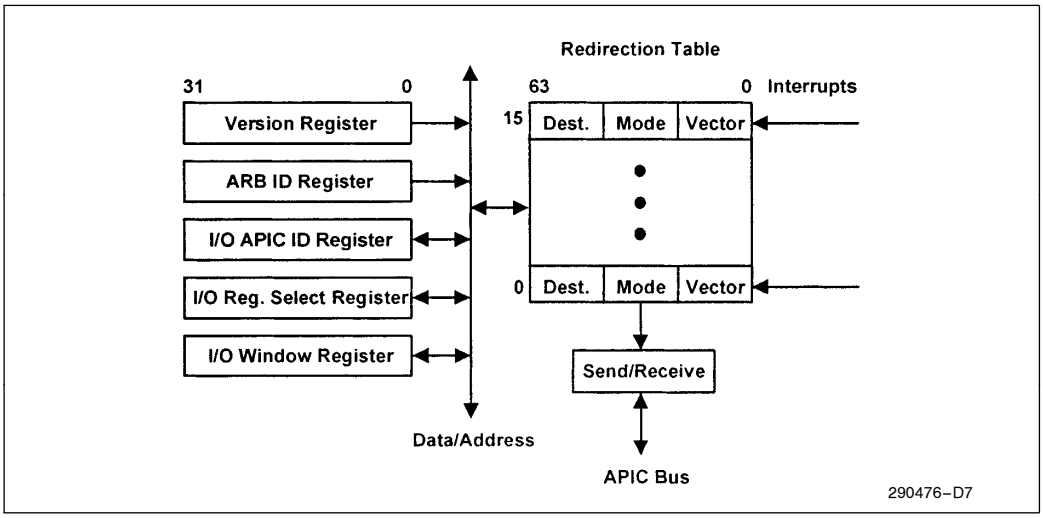


Figure 24. APIC Register Block Diagram

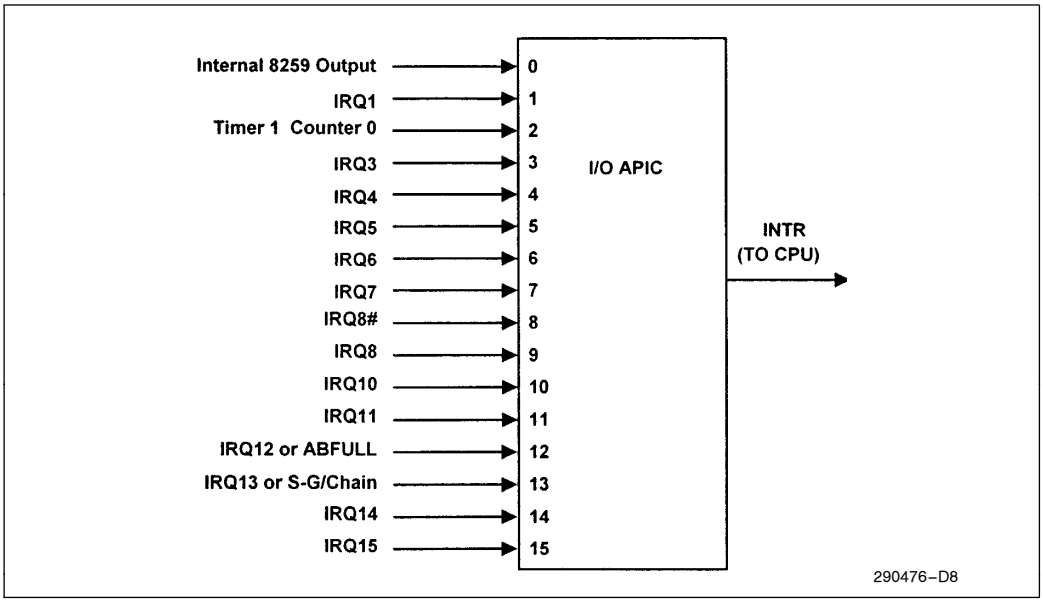


Figure 25. I/O APIC Interrupt Mapping

10.1 Physical Characteristics Of APIC Bus

The APIC bus is a 3-wire synchronous bus connecting all APICs (all I/O units and all local units). Two of these wires are used for data transmission, and one wire is a clock. For bus arbitration, the APIC uses only one of the data wires. The bus is logically a wire-OR and electrically an open-drain connection providing for both message transmission and arbitration for lowest priority. All the values mentioned in the protocol description are logical values (i.e. ''Bus Driven'' is logical 1 and ''Bus Not Driven'' is logical 0). The electrical values are 0 for logical one and 1 for logical zero.

10.2 Arbitration For APIC Bus

The APIC uses one wire arbitration to win the bus ownership. A rotating priority scheme is used for arbitration. The winner of the arbitration becomes the lowest priority agent and assumes an arbitration ID of 0. All other agents, except the agent whose arbitration ID is 15, increment their arbitration IDs by one. The agent whose ID was 15 takes the winner's arbitration ID and increments it by one. Arbitration IDs are changed (incremented or assumed) only for messages that are transmitted successfully. For lowest priority messages, the arbitration ID is updated before the final status cycle, which ultimately decides if the message is successful. A message is transmitted successfully if no CS error or acceptance error is reported for that message.

An APIC agent can acquire the bus using two different priority schemes; normal, or EOI (End Of Interrupt). EOI has the highest priority. EOI priority is used to send EOI messages for level interrupts from local APIC to the I/O APIC. When an agent requests the bus with EOI priority, all others requesting the bus with normal priorities back off.

A bus arbitration cycle starts by the agent driving a start cycle (bit 1 = EOI, bit 0 = 1) on the APIC bus (Table 29). Bit 1 = 1 indicates ''EOI'' priority and bit 1 = 0 indicates normal priority. Bit 0 should be 1.

In cycles 2 through 5, the agent drives the arbitration ID on bit 1 of the bus. High-order ID bits are driven first with successive cycles proceeding to the low bits of the ID. All arbitration losers in a given cycle drop off the bus, using every subsequent cycle as a tie breaker for the previous cycle. When all arbitration cycles are completed, there will be only one agent left driving the bus.

Table 29. Bus Arbitration Cycles

Cycle	Bit 1	Bit 0	Comments
1	EOI	1	0 1 = normal, 1 1 = EOI
2	ArbID3	0	Arbitration ID bits 3 through 0
3	ArbID2	0	
4	ArbID1	0	
5	ArbID0	0	

10.3 Bus Message Formats

After bus arbitration the winner is granted exclusive use of the bus and drives its message on the bus. APIC messages come in four formats—14 cycle EOI Message, 21 cycles Short Message, 33 cycles Lowest Priority Message, and 39 cycles Remote Read Message. All APICs on the APIC bus know the length of an interrupt message by checking the appropriate fields in the message.

EOI Message For Level Triggered Interrupts

The EOI Message is used to send an EOI cycle occurring for a level triggered interrupt from local APIC to the I/O APIC. This cycle contains the priority vector (V[7:0]) of the interrupt. When this message is received, the I/O APIC resets the Remote IRR bit for that interrupt. If the interrupt signal is still active after the RIRR bit is reset, the I/O APIC will treat it as a new interrupt.

Table 30. EOI Message

Cycle	Bit 1	Bit 0	Comments
1	EOI	1	0 1 = normal, 1 1 = EOI
2	ArbID3	0	Arbitration ID bits 3 through 0
3	ArbID2	0	
4	ArbID1	0	
5	ArbID0	0	
6	V7	V6	Interrupt vector V7–V0
7	V5	V4	
8	V3	V2	
9	V1	V0	

Table 30. EOI Message (Continued)

Cycle	Bit 1	Bit 0	Comments
10	C	C	Check Sum
11	0	0	Postamble
12	A	A	Status Cycle0
13	A1	A1	Status Cycle1
14	0	0	Idle

Short Message

Short Messages are used for the delivery of Fixed, NMI, SMI (82374SB only), Reset, ExtINT and Lowest Priority with Focus processor interrupts. The delivery mode bits (M[2:0]) specify the message. All short messages take 21 cycles, including the idle cycle.

Cycle 1 is the start cycle (Table 31). Cycles 2 through 5 are for bus arbitration as described earlier. APIC ID bits are sent on the bus one bit at a time (Only one data bus bit is used). The other bit should be zero. Cycles 6 and 7 provide destination mode and delivery mode bits. Cycle 8 provides level and trigger mode information. Cycles 10 through 13 are the 8-bit interrupt vector. The vector is only defined for delivery modes fixed, and lowest-priority. For delivery mode of ''Remote Read'', the vector field contains the address of the register to be read remotely.

If Destination Mode (DM) is 0 (physical mode), then cycles 15 and 16 are the APIC ID and cycles 13 and 14 are zero. If DM is 1 (logical mode), then cycles 13 through 16 are the 8-bit destination field. The interpretation of the logical mode 8-bit destination field is performed by the local units using the Destination Format Register. Shorthands of ''all-incl-self'' and ''all-excl-self'' both use physical destination mode and a destination field containing APIC ID value of all ones. The sending APIC knows whether it should (incl) or should not (excl) respond to its own message.

Cycle 17 is a checksum for the data in cycles 6 through 16. The (single) APIC driving the message provides this checksum in cycle 17.

Cycle 18 is a postamble cycle driven as 00 by all APICs to perform various internal computations based on the information contained in the received message. One of the computations takes the computed checksum of the data received in cycles 6 through 16 and compares it against the value in cycle 17. If any APIC computes a different checksum than the one passed in cycle 17, then that APIC signals an error on the APIC bus in cycle 19 by driving it as 11. If this happens, all APICs assume the message was never sent and the sender must try sending the message again, which includes re-arbitrating for the APIC bus. In lowest priority delivery when the interrupt has a focus processor, the focus processor signals this by driving 10 during cycle 19. This tells all the other APICs that the interrupt has been accepted, the LP arbitration is preempted, and short message format is used. Cycle 19 and 20 indicates the status of the message (i.e. accepted, check sum error, retry or error). Table 32 shows the status signals combinations and their meanings for all delivery modes.

The checksum is calculated iteratively on each cycle by adding the following terms:

1. The two least significant bits from the last cycle's checksum.
2. The current two data bits.
3. The carry bit from the last cycle's checksum shifted to the least significant bit.

Note that, at the beginning of the calculation, the three bits composing the previous cycle's checksum (two lower bits and carry) are zero.

Table 31. Short Message

Cycle	Bit 1	Bit 0	Comments
1	0	1	0 1 = Normal, 1 1 = EOI
2	ArbID3	0	Arbitration ID bits 3 through 0
3	ArbID2	0	
4	ArbID1	0	
5	ArbID0	0	
6	DM	M2	DM = Destination Mode
7	M1	M0	M2–M0 = Delivery Mode
8	L	TM	L = Level, TM = Trigger Mode
9	V7	V6	V7–V0 = Interrupt Vector
10	V5	V4	
11	V3	V2	
12	V1	V0	
13	D7	D6	Destination
14	D5	D4	
15	D3	D2	
16	D1	D0	
17	C	C	Checksum for Cycles 6–16
18	0	0	Postamble
19	A	A	Status Cycle 0
20	A1	A1	Status Cycle 1
21	0	0	Idle

Table 32. APIC Bus Message Status Information

Delivery Mode	Focus Processor	Status: A A	Comments	Status: A1 A1	Comments
Fixed, EOI	N/A	00	CS is OK	10	Accepted
Fixed, EOI	N/A	00	CS is OK	11	Retry
Fixed, EOI	N/A	00	CS is OK	0X	Error
Fixed, EOI	N/A	11	CS Error	XX	
Fixed, EOI	N/A	10	Error	XX	
Fixed, EOI	N/A	01	Error	XX	
NMI, SMM, Reset, ExtINT	N/A	00	CS is OK	10	Accepted
NMI, SMM, Reset, ExtINT	N/A	00	CS is OK	11	Error
NMI, SMM, Reset, ExtINT	N/A	00	CS is OK	0X	Error
NMI, SMM, Reset, ExtINT	N/A	11	CS Error	XX	
NMI, SMM, Reset, ExtINT	N/A	10	CS Error	XX	
NMI, SMM, Reset, ExtINT	N/A	01	CS Error	XX	
Lowest Priority	No	00	CS is OK. No Focus Processor	11	Go for LP Arb.
Lowest Priority	No	00	CS is OK. No Focus Processor	10	End and Retry
Lowest Priority	No	00	CS is OK. No Focus Processor	0X	Error
Lowest Priority	Yes	10	CS is OK. Focus Processor	XX	
Lowest Priority	Yes	11	CS Error	XX	
Lowest Priority	Yes	01	Error	XX	
Remote Read	N/A	00	CS is OK	XX	
Remote Read	N/A	11	CS Error	XX	
Remote Read	N/A	01	Error	XX	
Remote Read	N/A	10	Error	XX	

NOTE:
CS = Check Sum

Lowest Priority (LP) without Focus Processor (FP) Message

This message format is used to deliver an interrupt in the lowest priority mode in which it does not have a Focus Processor. Cycles 1 through 20 for this message are the same as for the Short Message discussed above. Status cycle 19 identifies if there is not a Focus processor (00) and a status value of 11 in cycle 20 indicates the need for lowest priority arbitration.

Cycle 21 through 28 are used to arbitrate for the lowest priority processor. The processors that take part in the arbitration drive their processor priority on the bus. Only the local APICs that have ''free interrupt slots'' participate in the lowest priority arbitration.

Cycle 29 through 32 are used to break a tie in case two or more processors have lowest priority. The bus arbitration IDs are used to break the tie.

Cycle 33 is an additional status cycle driven by the accepting local APIC. By receiving 10 during this cycle, the sending I/O APIC knows that there was a local APIC left after LP arbitration. Otherwise, the message is retried. Cycle 34 is an idle cycle.

Table 33. Lowest Priority without Focus Processor Message

Cycle	Bit 1	Bit 0	
1	0	1	0 1 = normal, 1 1 = EOI
2	ArbID3	0	Arbitration ID bits 3 through 0
3	ArbID2	0	
4	ArbID1	0	
5	ArbID0	0	
6	DM	M2	DM = Destination mode
7	M1	M0	M2–M0 = Delivery mode
8	L	TM	L = Level, TM = Trigger Mode
9	V7	V6	V7–V0 = Interrupt Vector
10	V5	V4	
11	V3	V2	
12	V1	V0	
13	D7	D6	Destination
14	D5	D4	
15	D3	D2	
16	D1	D0	
17	C	C	Checksum for cycles 6-16
18	0	0	Postamble
19	A	A	Status cycle 0
20	A1	A1	Status cycle 1

Table 33. Lowest Priority without Focus Processor Message (Continued)

Cycle	Bit 1	Bit 0	
21	P7	0	Inverted Processor Priority P7–P0
22	P6	0	
23	P5	0	
24	P4	0	
25	P3	0	
26	P2	0	
27	P1	0	
28	P0	0	
29	ArbID3	0	Arbitration ID 3 -0
30	ArbID2	0	
31	ArbID1	0	
32	ArbID0	0	
33	S	S	Status (10 means status OK; all other values indicate an error)
34	0	0	Idle

Remote Read Message

The Remote Read Message (Table 34) is used for a local APIC to read the register in another local APIC. The message format is the same as the Short Message for the first 20 cycles.

Cycles 21 through 36 contain the Remote Register data. The status information in cycle 37 specifies if the data is good or not. The Remote Read Message is always successful in that it is never retried (although the data may be valid or invalid). The reason is that the Remote Read Message is a debug feature, and a ‘‘hung’’ remote APIC that is unable to respond should not cause the debugger to hang up.

Table 34. Remote Read Message

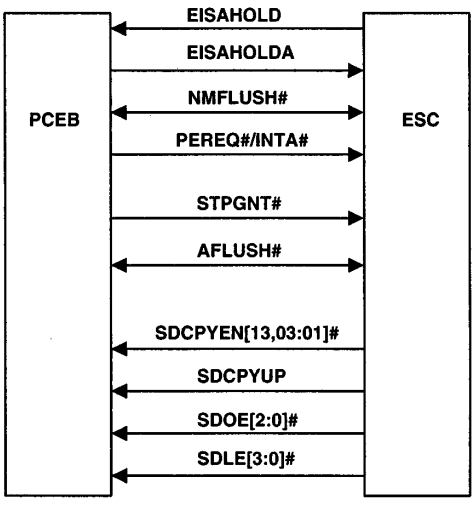
Cycle	Bit 1	Bit 0	
1	0	1	0 1 = normal, 1 1 = EOI
2	ArbID3	0	Arbitration ID bits 3 through 0
3	ArbID2	0	
4	ArbID1	0	
5	ArbID0	0	
6	DM	M2	DM = Destination mode
7	M1	M0	M2–M0 = Delivery mode
8	L	TM	L = Level, TM = Trigger Mode

Table 34. Remote Read Message (Continued)

Cycle	Bit 1	Bit 0	
9	V7	V6	V7–V0 = Interrupt Vector
10	V5	V4	
11	V3	V2	
12	V1	V0	
13	D7	D6	Destination
14	D5	D4	
15	D3	D2	
16	D1	D0	
17	C	C	Checksum for cycles 6-16
18	0	0	Postamble
19	A	A	Status cycle 0
20	A1	A1	Status cycle 1
21	d31	d30	Remote register data 31-0
22	d29	d28	
23	d27	d26	
24	d25	d24	
25	d23	d22	
26	d21	d20	
27	d19	d18	
28	d17	d16	
29	d15	d14	
30	d13	d12	
31	d11	d10	
32	d09	d08	
33	d07	d06	
34	d05	d04	
35	d03	d02	
36	d01	d00	
37	S	S	Data Status: 11 = valid, 00 = invalid
38	C	C	Check Sum for data d[31:00]
39	0	0	Idle

11.0 PCEB/ESC INTERFACE

The PCEB/ESC interface (Figure 26) 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 logic), and interrupt acknowledge cycles.



NOTE:
STPGNT# is only on the 82374SB (and companion 82375SB).

Figure 26. PCEB/ESC Interface Signals

11.1 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 does 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 tristates NMFLUSH# after asserting if 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.

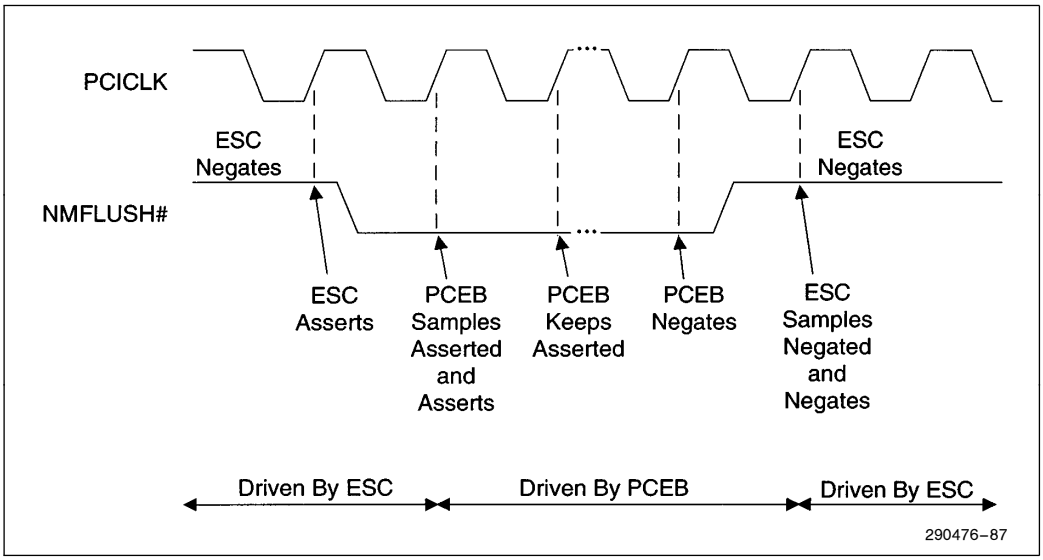


Figure 27. NMFLUSH# Protocol

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 11.5, Interrupt Acknowledge Control.

11.2 System Buffer Coherency Control—APIC

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

11.3 Power Management

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

11.4 EISA Data Swap Buffer Control Signals

The cycles in the EISA environment may require data size translations before the data can be transferred to its intermediate or final destination. As an example, a 32-bit EISA master write cycle to a 16-bit EISA slave requires a disassembly of a 32-bit Dword into 16 bit Words. Similarly, a 32-bit EISA master read cycle to a 16-bit slave requires an assembly of two 16-bit Words into a 32-bit Dword. The PCEB contains EISA data swap buffers to support data size translations on the EISA Bus. The operation of the data swap logic is described in the PCEB data sheet. The ESC controls the operation of the PCEB's data swap logic 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[13,3:1]#)

These signals enable the byte copy operations between data byte lanes 0, 1, 2 and 3 as shown in the Table 35. 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 35. 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 36). 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 36. 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.

11.5 Interrupt Acknowledge Control

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

12.0 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

NOTE:
The ESC directly supports X-Bus Floppy Disk Controller and IDE. If the IDE resides on another bus (e.g., ISA or PCI Bus), additional hardware (to modify the X-Bus signals) is required to support the X-Bus Floppy Disk Controller.

12.1 EISA Address Buffer Control

The EISA Bus consists of unlatched 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 (Figure 28). 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, 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 by driven by the F543s.

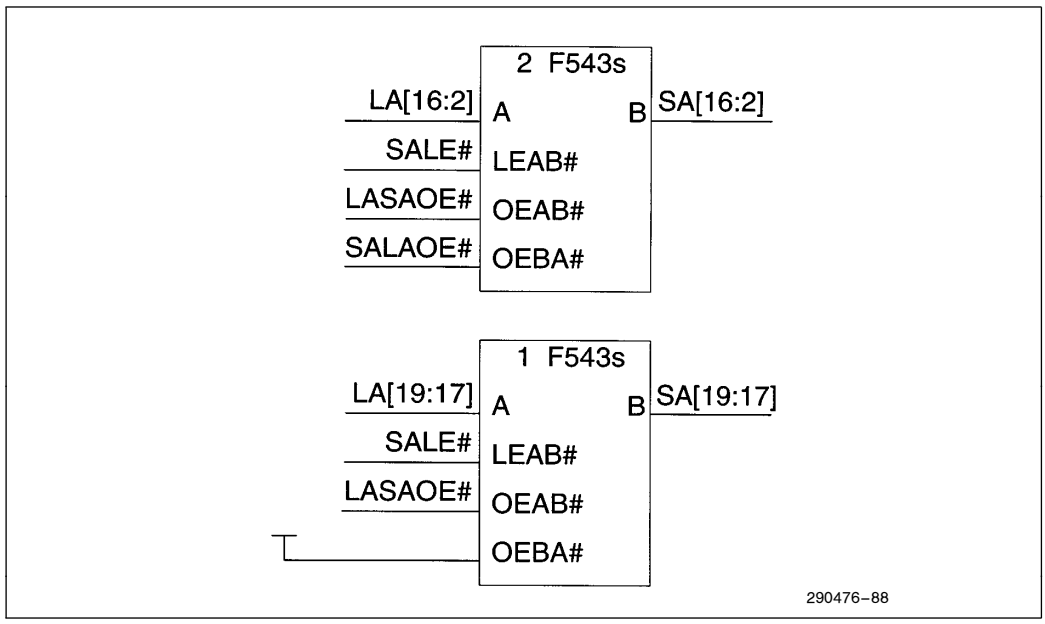


Figure 28. EISA Address Buffers

Table 37. 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

12.2 Coprocessor Interface

The numeric coprocessor interface is designed to support PC/AT compatible numeric coprocessor exception handling. The EISA Clock Divider 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 INTR 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 Divider configuration register then the ESC IRQ13 pins cannot be used, and this pin should be tied to ground.

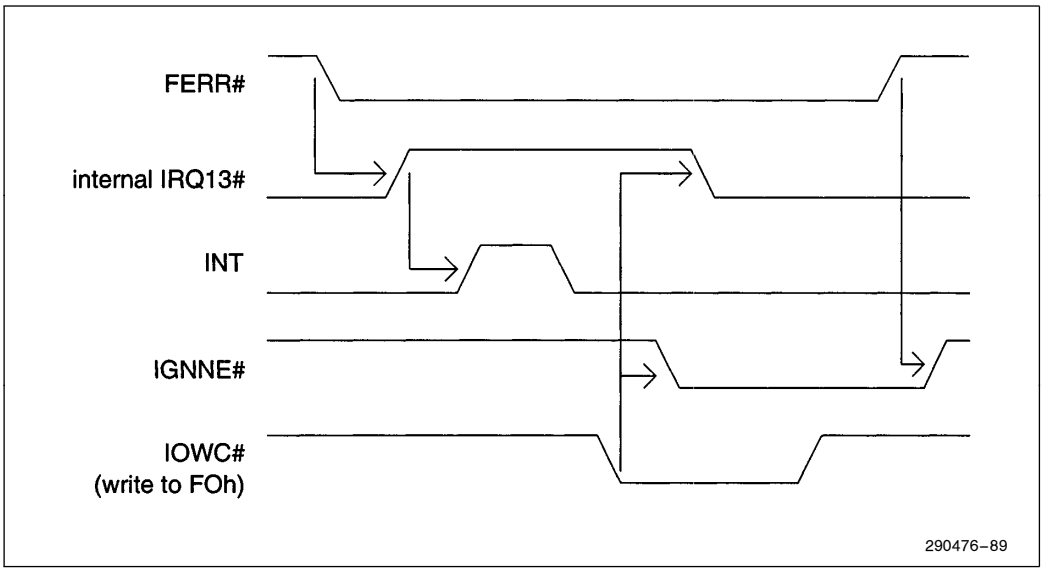


Figure 29. Coprocessor Interface Waveform

12.3 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).

12.4 Keyboard Controller Interface

The ESC provides a complete interface 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 (82374EB/SB), 62h (82374EB only), 64h (82374EB/SB), and 66h (82374EB only) 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. ALTRST# is asserted for approximately 4 BCLKs. The ESC will not pulse the ALTRST# signal if bit 0 has previously been set to a 1.

12.5 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 access are addressed to Real Time Clock addresses (write to 70h) 36h to 3Fh if the power on password protection is enabled.

12.6 Floppy Disk Control Interface

The ESC supports 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 samples 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.

12.7 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 Kbyte 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.

12.8 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 granularity.

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 a encoded chip select for COMACS# or COMBCS#.

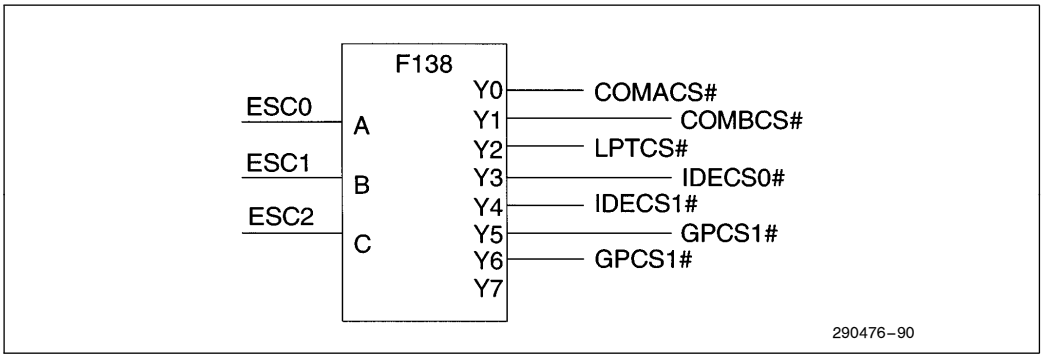


Figure 30. Encoded Chip Select Decoder Logic

Table 38. 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

NOTE:
Refer to Section 4.5 for the address decode of the peripheral chip selects.

12.9 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, XBUST/R# and XBUSOE#, for the discrete F245 buffer.

NOTE:
The ESC directly supports X-Bus floppy disk controller and X-Bus IDE hard disk controller. If IDE resides elsewhere (e.g., ISA Bus or PCI Bus), additional hardware is required to support the X-Bus floppy. The additional hardware is used to modify the X-Bus control signals. Figure 31 shows the logic needed to support an X-Bus floppy disk with IDE on the ISA or PCI bus.

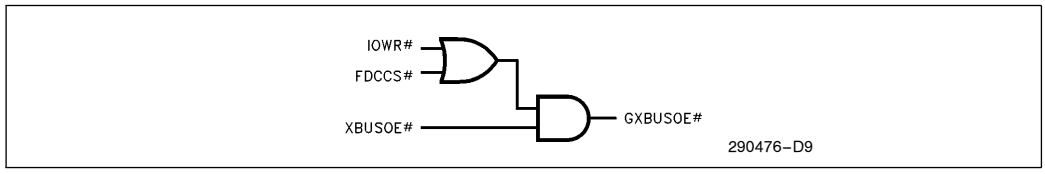


Figure 31

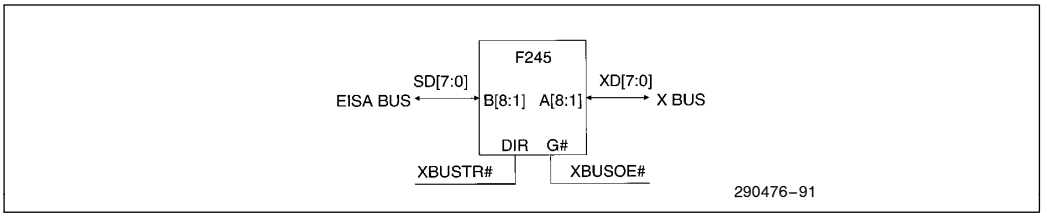


Figure 32. X-Bus Data Buffer

The XBUST/R# signal controls the direction of the data flow of the F245. When the XBUST/R# 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 XBUST/R# 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 XBUST/R# 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 XBUST/R# 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.

13.0 POWER MANAGEMENT (82374SB)

The ESC has extensive power management capability permitting a system to operate in a low power state without being powered down. In a typical desktop personal computer there are two states—Power On and Power-Off. Leaving a system powered on when not in use wastes power. The ESC provides a Fast On/Off feature that creates a third state called Fast Off (Figure 33). When in the Fast Off state, the system consumes less power than the Power-On state.

The ESC's power management architecture is based on three functionsSystem Management Mode (SMM), Clock Control, and Advanced Power Management (APM). Software (called SMM code) controls the transitions between the Power On state and the Fast Off state. The ESC invokes this software by generating an SMI to the CPU (asserting the SMI# signal). A variety of programmable events are provided that can generate an SMI. The SMM code places the system in either the Power On state or the Fast Off state.

A Fast On event is an event that instructs the computer (via an SMI to the CPU) to enter the Power-On state in anticipation of system activity by the user. Fast On events are programmable and include moving the mouse, pressing a key on the keyboard, an external hardware event, an incoming call to a system FAX/Modem, a RTC alarm, or the operating system.

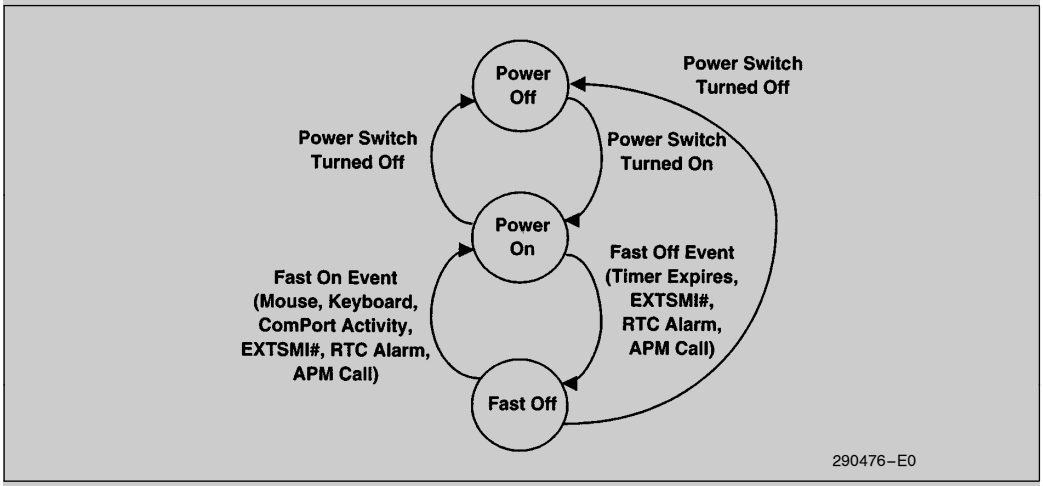


Figure 33. Fast On/Off Flow

13.1 SMM Mode

SMM mode is invoked by asserting the SMI# signal to the CPU. The ESC provides a variety of programmable events that can generate an SMI. When the CPU receives an SMI, it enters SMM mode and executes SMM code out of SMRAM. The SMM code places the system in either the Power On state or the Fast Off state. In the Power On state, the computer system operates normally. In this state one of the four programmable events listed below can trigger an SMI.

1. A global idle timer called the Fast Off timer expires (an indication that the end user has not used the computer for a programmed period of time).
2. The EXTSMI# pin is asserted.
3. An RTC alarm interrupt is detected.
4. The operating system issues an APM call.

13.2 SMI Sources

The SMI# signal can be asserted by hardware interrupt events, the Fast Off Timer, an external SMI event (EXTSMI#), and software events (via the APMC and APMS Registers). Enable/disable bits (in the SMIEN Register) permit each event to be individually masked from generating an SMI. In addition, the SMI# signal can be globally enabled/disabled in the SMICNTL Register. Status of the individual events causing an SMI is provided in the SMIREQ Register. For detailed information on the SMI control/status registers, refer to Section 3.0, Register Description.

Hardware Interrupt Events

Hardware events (IRQ[12,8#,4,3,1] and the Fast Off Timer) are enabled/disabled from generating an SMI in the SMIEN Register. When enabled, the occurrence of the corresponding hardware event generates an SMI (asserts the SMI# signal), regardless of the current power state of the system.

Fast Off Timer

The Fast Off Timer is used to indicate (through an SMI) that the system has been idle for a programmed period of time. The timer counts down from a programmed start value and when the count reaches 00h, can generate an SMI. The timer decrement rate is 1 count every minute and is re-loaded each time a System Event occurs. This counter should NOT be programmed to 00h.

System events are programmable events that can keep the system in the Power On state when there is system activity (Figure 34). These events are indicated by the assertion of IRQ[15:9,8#,7:3,1:0], NMI, or SMI signals. The system event prevents the system from entering the Fast Off state by re-loading the Fast Off Timer.

In addition to system events, *break events cause the system to transition from a Fast Off state to the Power On state. System events (and break events) are enabled/disabled in the SEE Register. When enabled and the associated hardware event occurs (signal is asserted), the Fast Off Timer is re-loaded with its initial count.*

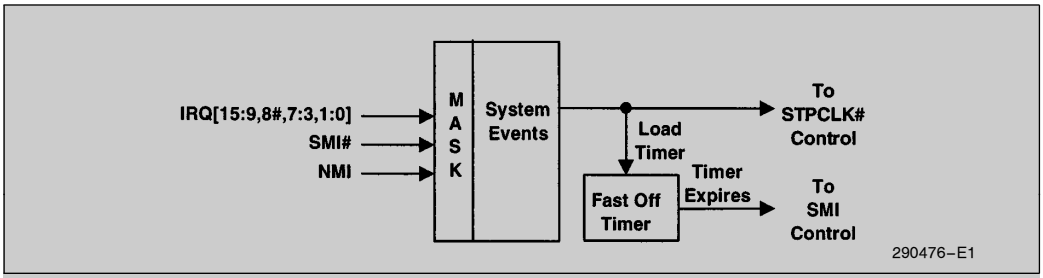


Figure 34. System Events and the Fast Off Timer

EXTSMI#

The EXTSMI# input pin provides the system designer the capability to invoke SMM with external hardware. For example, the EXTSMI# input could be connected to a ''green button'' permitting the user to enter the Fast Off state by depressing a button. The EXTSMI# generation of an SMI is enabled/disabled in the SMIEN Register.

Software Events

Software events (accessing the APMx Registers) indicate that the OS is passing power management information to the SMI handler. There are two Advanced Power Management (APM) registers—APM Control (APMC) and APM Status (APMS) Registers. These registers permit software to generate an SMI; by writing to the APMC Register. For example, the APMC can be used to pass an APM command between APM OS and BIOS and the APMS Register could be used to pass data between the OS and the SMI handler. For detailed descriptions of these registers, see Section 3.0, Register Description.

The two APM Registers are located in normal I/O space. The PCEB subtractively decodes PCI accesses to these registers and forwards the accesses to the EISA Bus. The APM Registers are not accessible by EISA masters. Note that the remaining power management registers are located in PCI configuration space.

13.3 SMI# And INIT Interaction

The SMI# input to the CPU is an edge sensitive signal. When an S-series processor is reset (INIT asserted), the processor resets the SMI# edge detect logic. After INIT is negated, it takes two clocks before the edge detect circuit can catch an edge. The ESC only asserts SMI# when INIT is negated. If the ESC asserts SMI# and then the INIT signal is sampled asserted, the ESC negates SMI#.

13.3.1 CLOCK CONTROL

The CPU can be put in a low power state by asserting the STPCLK# signal. STPCLK# is an interrupt to the CPU. However, for this type of interrupt, the CPU does not generate an interrupt acknowledge cycle. Once the STPCLK# interrupt is executed, the CPU enters the stop grant state. In this state, the CPU's internal clocks are disabled and instruction execution is stopped. The stop grant state is exited when the STPCLK# signal is negated.

Software can assert STPCLK#, if enabled via the SMICNTL Register, by a read of the APMC Register. Note that STPCLK# can also be periodically asserted by using clock scaling as described below.

The ESC automatically negates STPCLK# when a break event occurs (if enabled in the SEE Register) and the CPU stop grant special cycle has been received. Software can negate STPCLK# by disabling STPCLK# in the SMICNTL Register or by a write to the APMC Register.

NOTE:

1. INIT is always enabled as a break event. Otherwise, INIT acts exactly as other break events:
   — If STPCLK# is negated when INIT is asserted, the STPCLK high timer is reloaded.
   — If INIT is asserted when STPCLK# is asserted but before the stop grant bus cycle, STPCLK# negation waits until after the stop grant bus cycle. This happens after the CPU is reset when it samples STPCLK# still asserted.
   — If INIT is asserted when STPCLK# is asserted and after the stop grant bus cycle, STPCLK# is negated immediately. This guarantees that STPCLK# will be negated after the CPU is reset.
2. While the STPCLK# signal is asserted, the external interrupts (NMI, SMI# and INT) may be asserted to the CPU. If INTR is asserted, it will remain asserted until the CPU INTA cycle is detected. If SMI# (or NMI) is asserted, it remains asserted until the SMI (or NMI) handler clears the ESC's CSMIGATE (or sets the ESC's NMIMASK bit). Thus, SMI#, NMI and INTR can be applied to the CPU independent of the STPCLK# signal state. Note that when SMI#, NMI, and IRQx are enabled as break events, the occurrence of the break event negates STPCLK#.

Clock Scaling (Emulating Clock Division)

Clock scaling permits the ESC to periodically place the CPU in a low power state. This emulates clock division. When clock scaling is enabled, the CPU runs at full frequency for a pre-defined time period and then is stopped for a pre-defined time period. The run/stop time interval ratio emulates the clock division effect from a power/performance point of view. However, clock scaling is more effective than dividing the CPU frequency. For example, if the CPU is in the stop grant state and a break event occurs, the CPU clock returns to full frequency. In addition, there is no recovery time latency to start the clock.

Two programmable 8-bit clock scale timer control registers set the STPCLK# high (negate) and low (assert) times the CTLTMRH and CTLTMRL Registers. The timer is clocked by a 32 μs internal clock. This allows a programmable timer interval for both the STPCLK# high and low times of 0-8 ms. When enabled via the SMICNTL Register, the STPCLK# Timer operates as follows:

- When STPCLK# is negated, the timer is loaded with the value in the CTLTMRH Register and starts counting down. When the timer reaches 00h, STPCLK# is asserted. Since the timer is re-loaded with the contents of the CTLTMRH Register every time STPCLK# is negated (for break events or clock throttling), the STPCLK# minimum inactive time is guaranteed.
- When STPCLK# is asserted, the timer is loaded with the value in the CTLTMRL Register. The timer does not begin to count until the Stop Grant Special Cycle is received. When the timer reaches 00h, STPCLK# is negated. Note that a break event also negates STPCLK#.

NOTE:

If STPCLK# is negated and a break event occurs, the STPCLK# Timer is loaded with the value in the CTLTMRH Register.

13.4 Stop Grant Special Cycle

The Host-Bridge (e.g. PCMC) translates the CPU's stop grant cycle into a PCI special cycle. The PCEB recognizes the stop grant PCI special cycle and asserts STPGNT# low to ESC for one PCICLK. The ESC does not start the STPCLK low timer until STPGNT# is asserted.

During Halt or Autohalt state, the P54C does not respond to STPCLK# assertion with a stop grant cycle. However, during this state an INTR, SMI# or NMI# assertion causes the CPU to exit the halted state, and eventually recognize the STPCLK# assertion with a stop grant cycle. The system design must guarantee that INTR, SMI# and NMI# assertion is not blocked outside of the chipset while STPCLK# is asserted. Otherwise, a potential deadlock situation will exist.

13.5 Dual-Processor Power Management Support

Figure 35 depicts the power management support for dual-processor (DP) or P54CT upgrade processor configuration. The input signals of SMI#, STPCLK#, and NMI of both OEM and upgrade sockets are tied together.

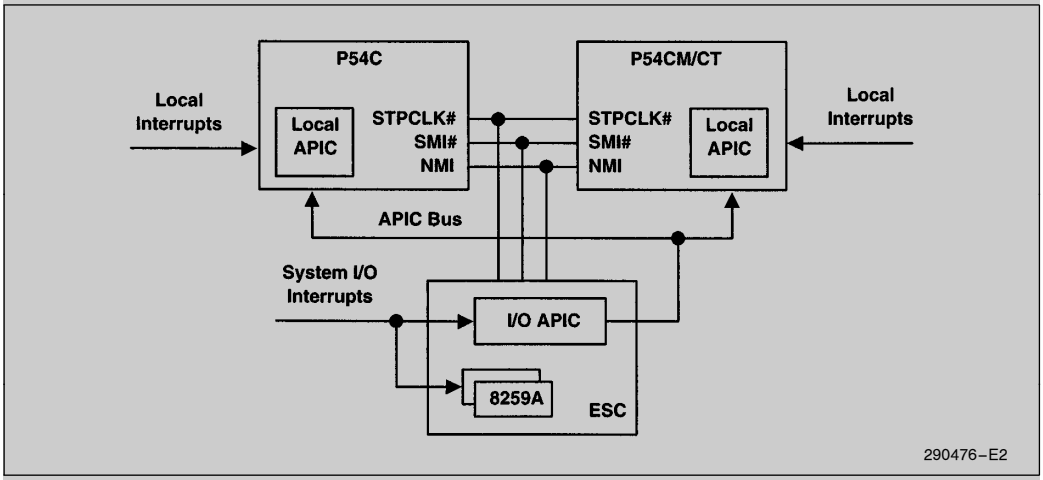


Figure 35. Dual Processor System Configuration

13.5.1 SMI# DELIVERY MECHANISM

For Uni or CT upgrade processor system configuration, SMI# can either be delivered through the ESC SMI# signal or I/O APIC. For the P54C/CM Dual-processor configuration, SMI# should be delivered through I/O APIC only. Ideally, the OS will put the CM processor in Autohalt after the CM processor received a Fast-Off SMI#. The CM processor will wake up if any non-masked system events occur.

13.5.2 STPCLK# TIED TO BOTH SOCKETS

To support a glueless upgrade socket, it is necessary to tie STPCLK# to both sockets. For P54C/CT processor configuration, the P54CT processor will disable P54C and the toggling of STPCLK# has no effect to P54C. For a P54C/CM DP configuration, the toggling of STPCLK# effects both processors (unless the processor is in Autohalt state). Both processors respond with a STPGNT special bus cycle after recognizing STPCLK# low. Both of the STPGNT special bus cycles are passed onto PCI by the PCMC as PCI STPGNT special cycles. The PCI STPGNT PCI cycle causes the PCEB component to assert the STPGNT# signal, depending on how the SCE bit in PCEB is programmed. The ESC recognizes the first STPGNT# assertion, and negates STPCLK# upon the Stop Clock timer expiration or a stop break event.

13.5.3 SMI#/INTR (APIC MODE)

When the APIC is used for interrupt delivery, additional considerations exist regarding ordering. If local interrupts (LINT0/1) are used in APIC mode, then the system can not guarantee an ordering between the local interrupts and any related SMI# events.

In DP mode, interrupts can generally be directed to a specific processor, which may not be the same processor that the SMI# is directed. The IRQ blocking logic in the ESC still operates with APIC delivery mode. Thus, if an IRQ is enabled to cause an SMI# event, it will be blocked until the CSMIGATE is cleared, regardless of where the IRQ or SMI is to be directed by the APIC.

14.0 ELECTRICAL CHARACTERISTICS

14.1 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

WARNING:

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

14.2 NAND Tree

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

The TEST pin, along with IRQ5 and IRQ3, activates the NAND Tree. Asserting TEST causes the output pulse train to appear on the EISAHOLD pin. IRQ5 must be driven high in order to enable the NAND Tree. The assertion of IRQ3 causes the ESC to disable its buffers.

The sequence of the ATE test is as follows:

1. Drive TEST low, IRQ3 high, and IRQ5 high.
2. Drive each pin that is a part of the NAND Tree high. Please note that not every pin is included in the tree. See table below for details.
3. Starting at pin 165 (DLIGHT#) and continuing with pins 167, 168, etc., individually drive each pin low. Expect EISAHOLD to toggle after each corresponding input pin is toggled. The final pin in the tree is pin 100 (EISAHOLD). Not every pin is toggled in sequential order. Please refer to the table for tree ordering. When IRQ3 is driven low, the test mode is exited, and the ESC's buffers will be enabled.
4. Before enabling the ESC's buffers (via IRQ3), turn off tester drivers.
5. Reset the ESC prior to proceeding with further testing.

Table 39. NAND Tree Cell Order (82374EB)

Pin #	Name
165	DLIGHT#[1]
167	FDCCS#
168	RTCWR#
169	RTCRD#
102	AFLUSH#
106	REFRESH#
107	APICCLK
108	APICD1
109	APICD0
110	IOCHK#
111	RSTDRV
112	IRQ9
113	DREQ2
114	NOWS#
115	CHRDY#
116	SMWTC#
121	SMRDC#
122	IOWC#
123	IORC#
125	DREQ3
127	DREQ1
135	IRQ7
136	IRQ6
141	BALE
142	OSC
143	SA1
144	SA0
145	M16#
146	SBHE#
147	IO16#
148	IRQ10

Pin #	Name
149	IRQ11
150	IRQ12
151	IRQ15
152	IRQ14
164	CRAMRD#
166	DSKCHG
171	ABFULL
179	CMD#
180	START#
186	EXRDY
187	EX32#
188	EX16#
189	SLBURST#
190	EOP
191	SPKR
193	IRQ8#
194	IRQ13
195	IRQ1
197	DREQ0
198	MRDC#
200	MWTC#
201	DREQ5
203	DREQ6
205	DREQ7
206	MASTER16#
3	LA31#/CPG4
4	LA30#/CPG3
5	LA29#/CPG2
6	LA28#/CPG1
7	LA27#/CPG0
8	LA26#
10	LA25#

Table 39. NAND Tree Cell Order (82374EB) (Continued)

Pin #	Name
11	LA24#
12	LA16
13	LA15
15	LA14
16	LA13
17	LA12
19	LA11
20	LA10
21	LA9
22	LA8
23	LA7
24	LA6
28	LA5
29	LA4
30	LA23
31	LA3
32	LA22
33	LA2
34	LA21
36	LA20
40	LA19
42	MREQ7#/PIRQ0#
43	LA18
44	MREQ6#/PIRQ1#
45	LA17
46	MREQ5#/PIRQ2#
47	MREQ4#/PIRQ3#
48	MREQ3#

Pin #	Name
49	MREQ2#
50	MREQ1#
51	MREQ0#
55	BE0#
56	BE1#
57	BE2#
58	BE3#
59	SD0
60	SD1
61	SD2
63	SD3
64	SD4
65	SD5
66	SD6
67	SD7
70	W/R#
71	M/IO#
72	MSBURST#
91	FERR#
95	RESET#
96	PERR#
97	SERR#
98	NMFLUSH#
99	PEREQ#/INTA#
138	IRQ4
139	IRQ3(2)
100	EISAHOLD(3)

Table 40. NAND Tree Cell Order (82374SB)

Pin #	Name
165	DLIGHT#(1)
167	FDCCS#
168	RTCWR#
169	RTCRD#
170	RTCALE
171	ABFULL
172	KYBDCS#
173	LBIOSCS#
174	SALAOE#
175	LASAOE#
176	SALE#
102	AFLUSH#
106	REFRESH#
107	APICCLK
108	APICD1
109	APICD0
110	IOCHK#
111	RSTDRV
112	IRQ9
113	DREQ2
114	NOWS#
115	CHRDY#
116	SMWTC#
117	AEN4
118	AEN3
119	AEN2
120	AEN1
121	SMRDC#
122	IOWC#
123	IORC#
124	DACK3#

Pin #	Name
125	DREQ3
126	DACK1#
127	DREQ1
133	STPGNT#
134	AEN#
135	IRQ7
136	IRQ6
140	DACK2#
141	BALE
142	OSC
143	SA1
144	SA0
145	M16#
146	SBHE#
147	IO16#
148	IRQ10
149	IRQ11
150	IRQ12
151	IRQ15
152	IRQ14
155	GPCS0#
159	GPCS1#
160	GPCS2#
161	XBUSOE#
162	XBUST/R#
163	CRAMWR#
164	CRAMRD#
166	DSKCHG
179	CMD#
180	START#
185	EXTSMI#
186	EXRDY

Table 40. NAND Tree Cell Order (82374SB) (Continued)

Pin #	Name	Pin #	Name
187	EX32#	16	LA13
188	EX16#	17	LA12
189	SLBURST#	19	LA11
190	EOP	20	LA10
191	SPKR	21	LA9
192	SLOWH#	22	LA8
193	IRQ8#	23	LA7
194	IRQ13	24	LA6
195	IRQ1	28	LA5
196	DACK0#	29	LA4
197	DREQ0	30	LA23
198	MRDC#	31	LA3
199	DACK5#	32	LA22
200	MWTC#	33	LA2
201	DREQ5	34	LA21
202	DACK6#	36	LA20
203	DREQ6	37	MACK2#
204	DACK7#	38	MACK1#
205	DREQ7	40	LA19
206	MASTER16#	41	MACK0#
207	MACK3#	42	MREQ7#/PIRQ0#
3	LA31#/CPG4	43	LA18
4	LA30#/CPG3	44	MREQ6#/PIRQ1#
5	LA29#/CPG2	45	LA17
6	LA28#/CPG1	46	MREQ5#/PIRQ2#
7	LA27#/CPG0	47	MREQ4#/PIRQ3#
8	LA26#	48	MREQ3#
10	LA25#	49	MREQ2#
11	LA24#	50	MREQ1#
12	LA16	51	MREQ0#
13	LA15	55	BE0#
15	LA14	56	BE1#

Table 40. NAND Tree Cell Order (82374SB) (Continued)

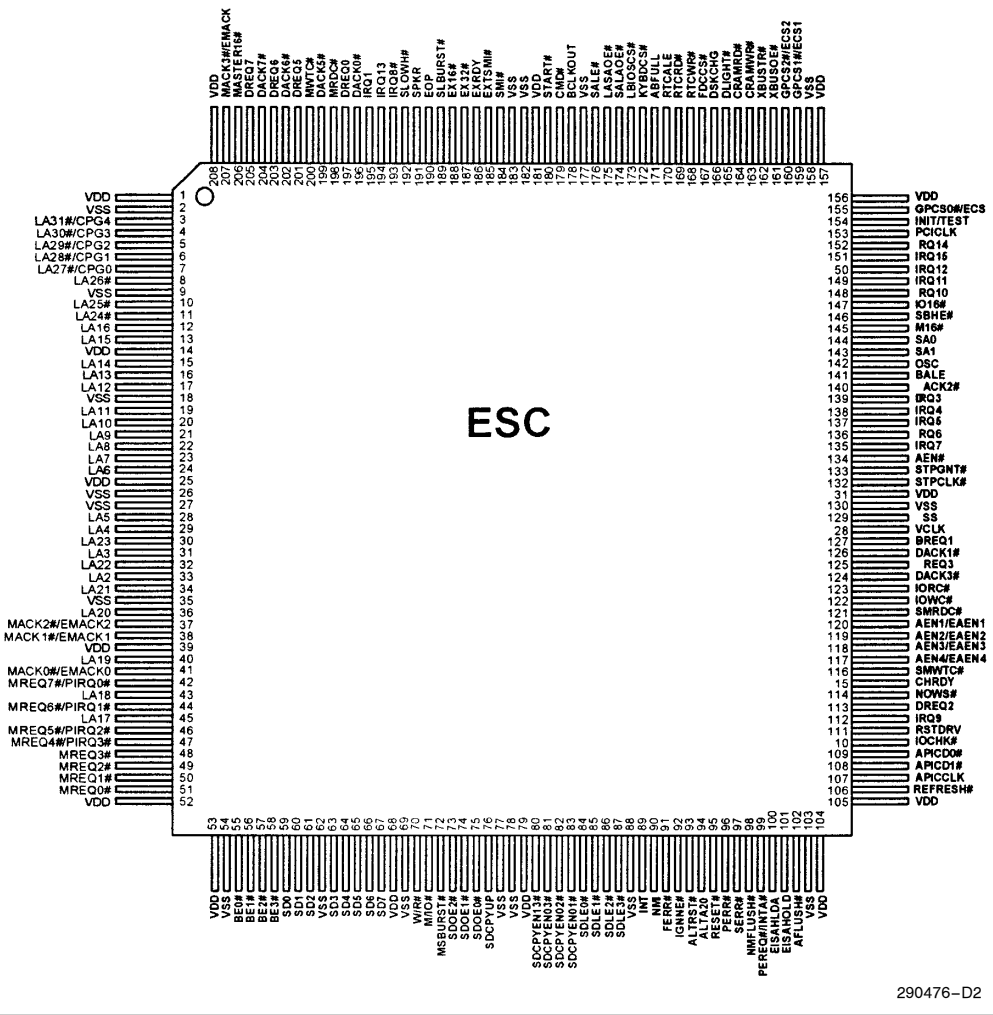
Pin #	Name
57	BE2#
58	BE3#
59	SD0
60	SD1
61	SD2
63	SD3
64	SD4
65	SD5
66	SD6
67	SD7
70	W/R#
71	M/IO#
72	MSBURST#
73	SDOE2#
74	SDOE1#
75	SDOE0#
76	SDCPYUP
80	SDCPYEN13#
81	SDCPYEN03#
82	SDCPYEN02#
83	SDCPYEN01#
84	SDLE0#
85	SDLE1#
86	SDLE2#
87	SDLE3#
91	FERR#
95	RESET#
96	PERR#
97	SERR#
98	NMFLUSH#
99	PEREQ#/INTA#
138	IRQ4
139	IRQ3(2)
100	EISAHOLD(3)

NOTES:
1. First Pin in NAND Tree.
2. Enables ESC's Buffers when 0.
3. Last Pin in NAND Tree.

15.0 PINOUT AND PACKAGE INFORMATION

The ESC package is a 208-pin Plastic Quad Flat Pack (PQFP). The package signals are shown in Figure 36 and listed in Table 41 and Table 42. Note that NC pins require individual pull-up resistors of 8 KΩ–10 KΩ.

15.1 Pinout And Pin Assignment



NOTE:
INIT, SMI#, STPCLK#, STPGNT#, EXTSMI#, are on the 82374SB only.

Figure 36. ESC Package Pinout

Table 41. 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
AFLUSH#	102	t/s
ALTA20	94	out
ALTRST#	93	out
APICCLK	107	in
APICD0#	109	od
APICD1#	108	od
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

Name	Pin #	Type
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
EOP	190	t/s
EX16#	188	o/d
EX32#	187	o/d
EXRDY	186	o/d
EXTSMI# (82374SB)	185	in
FDCCS#	167	out
FERR#	91	in
GPCS0#/ECS0	155	out
GPCS1#/ECS1	159	out
GPCS2#/ECS2	160	out
IGNNE#	92	out
INIT/TEST (82374SB)	154	in
INTR	89	out
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

Table 41. ESC Alphabetical Pin Assignment (Continued)

Name	Pin #	Type
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
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

Name	Pin #	Type
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
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

Table 41. ESC Alphabetical Pin Assignment (Continued)

Name	Pin #	Type
MWTC#	200	t/s
NC (82374EB)	132	—
NC (82374EB)	133	—
NC (82374EB)	184	—
NC (82374EB)	185	—
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
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

Name	Pin #	Type
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
SMI# (82374SB)	184	out
SMRDC#	121	out
SMWTC#	116	out
SPKR	191	out
START#	180	t/s
STPCLK# (82374SB)	132	out
STPGNT# (82374SB)	133	in
TEST (82374EB)	154	in
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}	68	V
V_{DD}	79	V
V_{DD}	104	V

Table 41. ESC Alphabetical Pin Assignment (Continued)

Name	Pin #	Type
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

Name	Pin #	Type
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
XBUST/R#	162	out

Table 42. ESC Numerical Pin Assignment

Name	Pin #	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

Name	Pin #	Type
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	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

Table 42. ESC Numerical Pin Assignment (Continued)

Name	Pin #	Type
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	SD2	t/s
65	SD5	t/s
66	SD6	t/s
67	SD7	t/s
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

Name	Pin #	Type
88	V_{SS}	V
89	INTR	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	AFLUSH#	t/s
103	V_{SS}	V
104	V_{DD}	V
105	V_{DD}	V
106	REFRESH#	t/s
107	APICCLK	in
108	APICD1#	od
109	APICD0#	od
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

Table 42. ESC Numerical Pin Assignment (Continued)

Name	Pin #	Type
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 (82374EB) STPCLK# (82374SB)	— out
133	NC (82374EB) STPGNT# (82374SB)	— in
134	AEN#	out
135	IRQ7	in
136	IRQ6	in
137	IRQ5	in
138	IRQ4	in
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

Name	Pin #	Type
149	IRQ11	in
150	IRQ12	in
151	IRQ15	in
152	IRQ14	in
153	PCICLK	in
154	TEST (82374EB) INIT/TEST (82374SB)	in 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	XBUST/R#	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
175	LASAOE#	out
176	SALE#	out
177	V_{SS}	V
178	BCLKOUT	out
179	CMD#	out

Table 42. ESC Numerical Pin Assignment (Continued)

Name	Pin #	Type
180	START#	t/s
181	V_{DD}	V
182	V_{SS}	V
183	V_{SS}	V
184	NC (82374EB) SMI# (82374SB)	— out
185	NC (82374EB) EXTSMI# (82374SB)	— in
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

Name	Pin #	Type
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

15.2 Package Characteristics

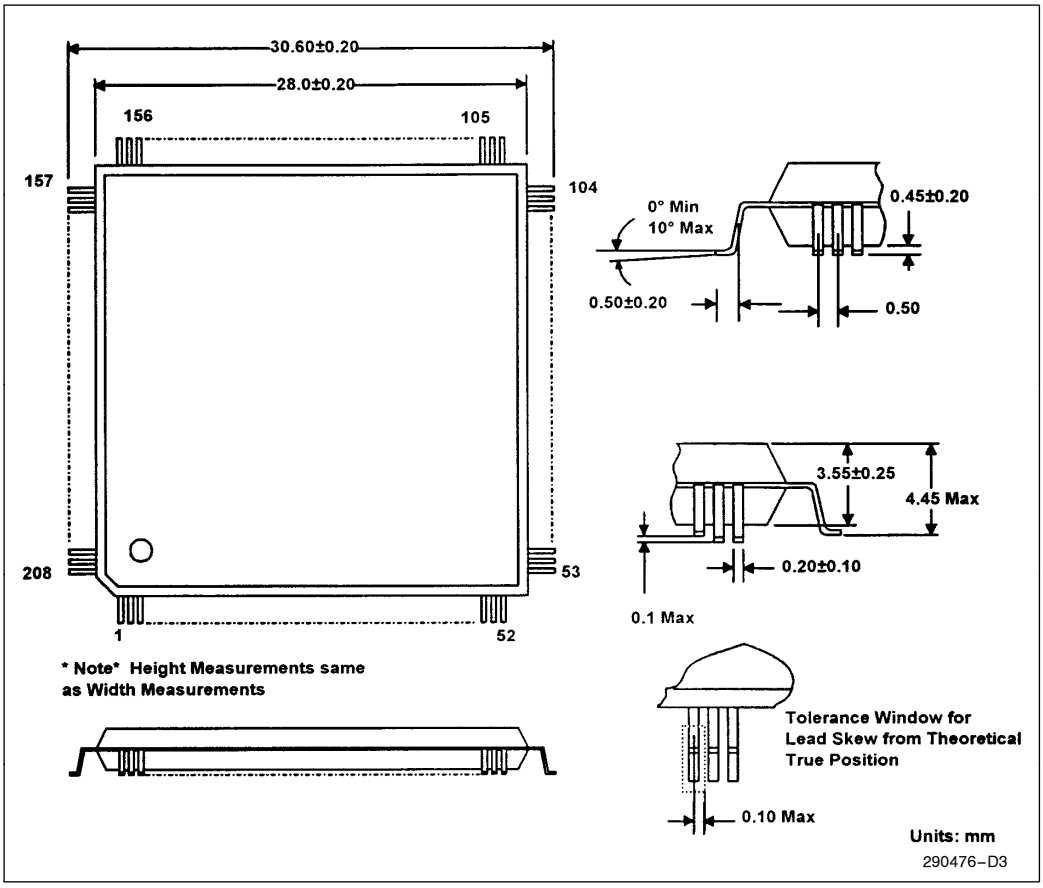


Figure 37. Packaging Dimension Information