82378ZB SYSTEM I/O (SIO) AND 82379AB SYSTEM I/O APIC (SIO.A)

- **Provides the Bridge Between the PCI Bus and ISA Bus**
- **100% PCI and ISA Compatible**
  - **— PCI and ISA Master/Slave Interface**
  - **— Directly Drives 10 PCI Loads and 6 ISA Slots**
  - **— PCI at 25 MHz and 33 MHz**
  - **— ISA from 6 MHz to 8.33 MHz**
- **Enhanced DMA Functions**
  - **— Scatter/Gather (S/G) (82378ZB)**
  - **— Fast DMA Type A, B and F (82378ZB)**
  - **— Compatible DMA Transfers**
  - **— 32-bit Addressability(82378ZB)**
  - **— 27-bit Addressability(82379AB)**
  - **— Seven Independently Programmable Channels**
  - **— Functionality of Two 82C37A DMA Controllers**
- **Data Buffers to Improve Performance**
  - **— 8-Byte DMA/ISA Master Line Buffer**
  - **— 32-bit Posted Memory Write Buffer to ISA**
- **Integrated 16-bit BIOS Timer**
- **Non-Maskable Interrupts (NMI)**
  - **— PCI System Errors**
  - **— ISA Parity Errors**
- **Arbitration for ISA Devices**
  - **— ISA Masters**
  - **— DMA and Refresh**
- **Four Dedicated PCI Interrupts**
  - **— Level Sensitive**
  - **— Mapped to Any Unused Interrupt**
- **Arbitration for PCI Devices**
  - **— Six PCI Masters Supported**
  - **— Fixed, Rotating, or a Combination**
- **Utility Bus (X-Bus) Peripheral Support**
  - **— Provides Chip Select Decode**
  - **— Controls Lower X-Bus Data Byte Transceiver**
- **Functionality of One 82C54 Timer**
  - **— System Timer**
  - **— Refresh Request**
  - **— Speaker Tone Output**
- **Functionality of Two 82C59 Interrupt Controllers**
  - **— 14 Interrupts Supported**
  - **— Edge/Level Selectable Interrupts**
- **I/O APIC (Advanced Programmable Interrupt Controller (82379AB)**
  - **— Support for Multi-Processor Systems**
- **System Power Management (Intel SMM Support)**
  - **— Programmable System Management Interrupt (SMI)—Hardware Events, Software Events, EXTSMI#**
  - **— Programmable CPU Clock Control (STPCLK#)**
  - **— Fast-On/Off Mode**
- **208-Pin QFP Package**

The 82378ZB System I/O (SIO) and 82379AB System I/O APIC (SIO.A) components are PCI-to-ISA Bus Bridge devices. These devices integrate many of the common I/O functions found in today's ISA-based PC systems—a seven channel DMA controller, two 82C59 interrupt controllers, an 8254 timer/counter, a BIOS timer, Intel SMM power management support, and logic for NMI generation. In addition, the SIO and SIO.A each support a total of six PCI Masters, and four PCI Interrupts. Decode is provided for peripheral devices such as the flash BIOS, real time clock, keyboard/mouse controller, floppy controller, two serial ports, one parallel port, and IDE hard disk drive.

For both the SIO and SIO.A, each DMA channel supports compatibility transfers. The SIO also supports types A, B, and F transfers and scatter/gather. In addtion to the standard ISA-compatible interrupt controller that is in both the SIO and SIO.A, the SIO.A contains an Advance Programmable Interrupt Controller (IO APIC) for use in multi-processing systems.

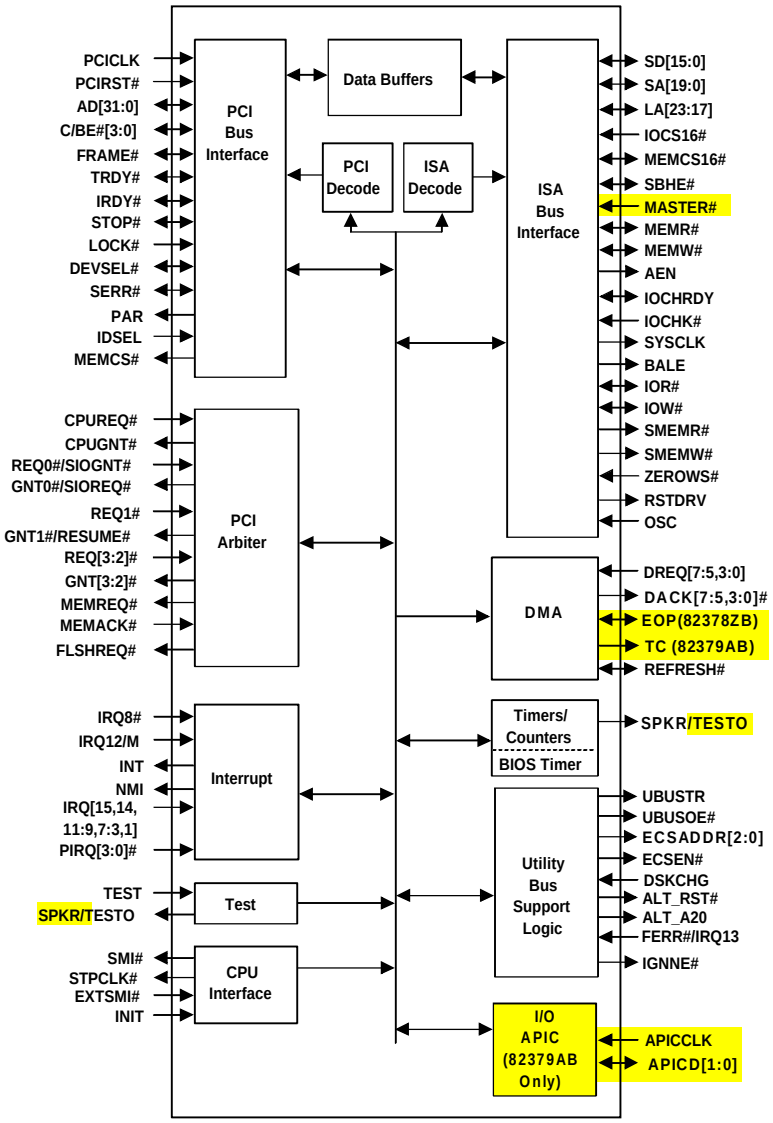
This document describes both the 82378ZB (SIO) and 82379AB (SIO.A) components. Unshaded areas describe the 82378ZB. Shaded areas, like this one, describe differences between the 82379AB and 82378ZB.

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

© INTEL CORPORATION 1996

March 1996

Order Number: 290571-001



Note: I/O APIC signals are only on the 82379AB. Also, SPKR/TESTO are only multiplexed on the 82379AB and are separate signals on the 82378ZB. The MASTER# signal is only on the 82378ZB.

82378ZB SIO and 82379AB SIO.A Component Block Diagram

CONTENTS

1.0. ARCHITECTURAL OVERVIEW

The major functions of the SIO and SIO.A components are broken up into blocks as shown in the SIO and SIO.A Component Block Diagrams. A description of each block is provided below.

PCI Bus Interface

The PCI Bus Interface provides the interface between the SIO/SIO.A and the PCI Bus. The SIO/SIO.A provides both a master and slave interface to the PCI Bus. As a PCI master, the SIO/SIO.A runs cycles on behalf of DMA, ISA masters, and the internal data buffer management logic when buffer flushing is required. The SIO/SIO.A burstS a maximum of two Dwords when reading from PCI memory, and one Dword when writing to PCI memory. The SIO/SIO.A does not generate PCI I/O cycles as a master. As a PCI slave, the SIO/SIO.A accepts cycles initiated by PCI masters targeted for the SIO/SIO.A internal register set or the ISA Bus. The SIO/SIO.A accepts a maximum of one data transaction before terminating the transaction. This supports the Incremental Latency Mechanism as defined in the Peripheral Component Interconnect (PCI) Specification.

As a master, the SIO/SIO.A generates address and command signal (C/BE#) parity for read and write cycles, and data parity for write cycles. As a slave, the SIO/SIO.A generates data parity for read cycles. Parity checking is not supported. The SIO/SIO.A also provides support for system error reporting by generating a Non-Maskable-Interrupt (NMI) when SERR# is driven active.

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

The SIO/SIO.A directly supports the PCI Interface running at either 25 MHz or 33 MHz. If a frequency of less than 33 MHz is required (not including 25 MHz), a SYSCLK divisor value (as indicated in the ISA Clock Divisor Register) must be selected that guarantees that the ISA Bus frequency does not violate the 6 MHz to 8.33 MHz SYSCLK range.

PCI Arbiter

The PCI arbiter provides support for six PCI masters; the Host Bridge, SIO/SIO.A, and four PCI masters. The arbiter can be programmed for a purely rotating scheme, fixed, or a combination of the two. The Arbiter can also be programmed to support bus parking. This gives the Host Bridge default access to the PCI Bus when no other device is requesting service. The arbiter can be disabled if an external arbiter is used.

Data Buffers

To isolate the slower ISA Bus from the PCI Bus, the SIO/SIO.A provides two types of data buffers. One Dword-deep posted write buffer is provided for the posting of PCI initiated memory write cycles to the ISA Bus. The second buffer is a bi-directional, 8-byte line buffer used for ISA master and DMA accesses to the PCI Bus. All DMA and ISA master read and write cycles go through the 8-byte line buffer. The data buffers also provide the data assembly or disassembly when needed for transactions between the PCI and ISA Buses. Buffering is programmable and can be enabled or disabled through software.

ISA Bus Interface

The SIO/SIO.A incorporates a fully ISA-Bus compatible master and slave interface. The SIO/SIO.A directly drives six ISA slots without external data or address buffering. The ISA interface also provides byte swap logic, I/O recovery support, wait-state generation, and SYSCLK generation. The SIO/SIO.A supports ISA Bus frequencies from 6 to 8.33 MHz.

As an ISA master, the SIO/SIO.A generates cycles on behalf of DMA, Refresh, and PCI master initiated cycles. The SIO/SIO.A supports compressed cycles when accessing ISA slaves (i.e., ZEROWS# asserted). As an ISA slave, the SIO/SIO.A accepts ISA master accesses targeted for the SIO/SIO.A internal register set or ISA master memory cycles targeted for the PCI Bus. The SIO/SIO.A does not support ISA master initiated I/O cycles targeted for the PCI Bus.

The SIO/SIO.A also monitors ISA master to ISA slave cycles to generate SMEMR# or SMEMW#, and to support data byte swapping, if necessary.

DMA

The DMA controller in the SIO/SIO.A 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. The DMA controller is also responsible for generating ISA refresh cycles.

For the 82378ZB, ISA-compatible or fast DMA type "A", type "B", or type "F" timings are supported and 32-bit addressing is supported as an extension of the ISA-compatible specification. The SIO supports an enhanced feature called Scatter/Gather (S/G). This feature provides the capability of transferring multiple buffers between memory and I/O without CPU intervention. In S/G mode, the DMA can read the memory address and word count from an array of buffer descriptors, located in system memory, called the S/G Descriptor (SGD) Table. This allows the DMA controller to sustain DMA transfers until all of the buffers in the SGD table are read.

For the SIO.A, the DMA supports 8-/16-bit device size using ISA-compatible timings and 27-bit addressing as an extension of the ISA-compatible specification. Scatter/Gather is not supported.

Timer Block

The timer block contains three counters that are equivalent in function to those found in one 82C54 programmable interval timer. These three counters are combined to provide the System Timer function, Refresh Request, and speaker tone. The three counters use the 14.31818 MHz OSC input for a clock source.

In addition to the three counters, the SIO/SIO.A provides a programmable 16-bit BIOS timer. This timer can be used by BIOS software to implement timing loops. The timer uses the ISA system clock (SYSCLK) divided by 8 as a clock source. An 8:1 ratio between the SYSCLK and the BIOS timer clock is always maintained. The accuracy of the BIOS timer is $\pm$ 1 ms.

Utility Bus (X-Bus) Logic

The SIO/SIO.A provides four encoded chip selects that are decoded externally to provide chip selects for flash BIOS, real time clock, keyboard/Mouse Controller, floppy controller, two serial ports, one parallel port, and an IDE hard disk drive. The SIO/SIO.A provides the control for the buffer that isolates the lower 8 bits of the Utility Bus from the lower 8 bits of the ISA Bus. In addition to providing the encoded chip selects and Utility Bus buffer control, the SIO/SIO.A also provides Port 92 functions (Alternate Reset and Alternate A20), Coprocessor error reporting, the Floppy DSKCHG function, and a mouse interrupt input.

Interrupt Controller Block

The SIO/SIO.A provides an ISA compatible interrupt controller that incorporates the functionality of two 82C59 interrupt controllers. The two interrupt controllers are cascaded so that 14 external and two internal interrupts are possible.

Advanced Programmable Interrupt Controller (APIC) (SIO.A Only)

In addition to the standard ISA-compatible interrupt controller described above, the SIO.A 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.

Power Management

Extensive power management capability permits a system to operate in a low power state without being powered down. Once in the low power state (called 'Fast-Off' state), the computer appears to be off. For example, the System Memory Management (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.

Test

The test block provides the interface to the test circuitry within the SIO/SIO.A. The test input can be used to tri-state all of the SIO/SIO.A outputs.

2.0. SIGNAL DESCRIPTION

This section contains a detailed description of each signal. The signals are arranged in functional groups according to the interface.

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

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

The following notations are used to described the signal type.

Signal Type	Description
I	**Input** is a standard input-only signal.
O	**Totem Pole Output** is a standard active driver.
OD	**Open Drain** Input/Output.
IO	**Input/Output** is a bidirectional, tri-state pin.
s/t/s	**Sustained Tri-State** is an active low tri-state signal owned and driven by one and only one agent at a time. The agent that drives a s/t/s pin low must drive it high for at least one clock before letting it float. A new agent can not start driving a s/t/s signal any sooner than one clock after the previous owner tri-states it. A pull-up sustains the inactive state until another agent drives it and is provided by the central resource.
t/s/o	**Tri-State Output**

2.1. PCI Bus Interface Signals

Signal Name	Type	Description
PCICLK	I	**PCI CLOCK:** PCICLK provides timing for all transactions on the PCI Bus. All other PCI signals are sampled on the rising edge of PCICLK, and all timing parameters are defined with respect to this edge. Frequencies supported by the SIO/SIO.A include 25 and 33 MHz.
PCIRST#	I	**PCI RESET:** PCIRST# forces the SIO/SIO.A to a known state. AD[31:0], C/BE[3:0]#, and PAR are always driven low by the SIO/SIO.A synchronously from the leading edge of PCIRST#. The SIO/SIO.A always tri-states these signals from the trailing edge of PCIRST#. If the internal arbiter is enabled (CPUREQ# sampled high on the trailing edge of PCIRST#), the SIO/SIO.A will drive these signals low again (synchronously 2-5 PCICLKs later) until the bus is given to another master. If the internal arbiter is disabled (CPUREQ# sampled low on the trailing edge of PCIRST#), these signals remain tri-stated until the SIO/SIO.A is required to drive them valid as a master or slave. FRAME#, IRDY#, TRDY#, STOP#, DEVSEL#, MEMREQ#, FLSHREQ#, CPUGNT#, GNT0#/SIOREQ#, and GNT1#/RESUME# are tri-stated from the leading edge of PCIRST#. FRAME#, IRDY#, TRDY#, STOP#, and DEVSEL# remain tri-stated until driven by the SIO/SIO.A as either a master or a slave. MEMREQ#, FLSHREQ#, CPUGNT#, GNT0#/SIOREQ#, and GNT1#/RESUME# are tri-stated until driven by the SIO/SIO.A. After PCIRST#, MEMREQ# and FLSHREQ# are driven inactive asynchronously from PCIRST# inactive. CPUGNT#, GNT0#/SIOREQ#, and GNT1#/RESUME# are driven based on the arbitration scheme and the asserted REQx#'s. All registers are set to their default values. PCIRST# may be asynchronous to PCICLK when asserted or negated. Although asynchronous, negation must be a clean, bounce-free edge. Note that PCIRST# must be asserted for more than 1 µs.
AD[31:0]	I/O	**PCI ADDRESS/DATA.** The standard PCI address and data lines. The address is driven with FRAME# assertion and data is driven or received in following clocks.
C/BE[3:0]#	I/O	**BUS COMMAND AND BYTE ENABLES:** The command is driven with FRAME# assertion. Byte enables corresponding to supplied or requested data are driven on following clocks.
FRAME#	I/O (s/t/s)	**CYCLE FRAME:** Assertion indicates the address phase of a PCI transfer. Negation indicates that one more data transfer is desired by the cycle initiator. FRAME# is tri-stated from the leading edge of PCIRST#.
TRDY#	I/O (s/t/s)	**TARGET READY:** Asserted when the target is ready for a data transfer. TRDY# is tri-stated from the leading edge of PCIRST#.
IRDY#	I/O (s/t/s)	**INITIATOR READY:** Asserted when the initiator is ready for a data transfer. IRDY# is tri-stated from the leading edge of PCIRST#.
STOP#	I/O (s/t/s)	**STOP:** Asserted by the target to request the master to stop the current transaction. STOP# is tri-stated from the leading edge of PCIRST#.
LOCK#	I	**LOCK:** LOCK# indicates an atomic operation that may require multiple transactions to complete. LOCK# is always an input to the SIO/SIO.A.
IDSEL	I	**INITIALIZATION DEVICE SELECT:** IDSEL is used as a chip select during configuration read and write transactions.

Signal Name	Type	Description
DEVSEL#	I/O (s/t/s)	**DEVICE SELECT:** The SIO/SIO.A asserts DEVSEL# to claim a PCI transaction through positive or subtractive decoding. DEVSEL# is tri-stated from the leading edge of PCIRST#. DEVSEL# remains tri-stated until driven by the SIO/SIO.A as either a master or a slave.
PIRQ[3:0]#	I	**PCI INTERRUPT REQUEST:** PIRQ#s are used to generate asynchronous interrupts to the CPU via the Programmable Interrupt Controllers (82C59s) integrated in the SIO/SIO.A. These signals are defined as level sensitive and are asserted low. The PIRQx# interrupts can be steered into any unused IRQ interrupt. The PIRQx# Route Control Register determines which IRQ interrupt each PCI interrupt is steered into. These pins include a weak internal pull-up resistor.
PAR	O	**CALCULATED PARITY SIGNAL:** A single parity bit is provided over AD[31:0] and C/BE[3:0]. PAR is always driven low by the 82379AB synchronously from the leading edge of PCIRST#.
SERR#	I	**SYSTEM ERROR:** SERR# can be pulsed active by any PCI device that detects a system error condition. Upon sampling SERR# active, the SIO/SIO.A generates a non-maskable interrupt (NMI) to the CPU.

2.2. PCI Arbiter Signals

Signal Name	Type	Description
CPUREQ#	I	**CPU REQUEST:** This signal provides the following functions: 1. If CPUREQ# is sampled high on the trailing edge of PCIRST#, the internal arbiter is enabled. If CPUREQ# is sampled low on the trailing edge of PCIRST#, the internal arbiter is disabled. This requires that the host bridge drive CPUREQ# high during PCIRST#. 2. If the SIO/SIO.A internal arbiter is enabled, this pin is configured as CPUREQ#. An active low assertion indicates that the CPU initiator desires the use of the PCI Bus. If the internal arbiter is disabled, this pin is meaningless after reset. This pin has a weak internal pull-up resistor.
REQ0#/ SIOGNT#	I	**REQUEST 0/SIO GRANT:** If the SIO/SIO.A internal arbiter is enabled, this pin is configured as REQ0#. An active low assertion indicates that Initiator0 desires the use of the PCI Bus. If the internal arbiter is disabled, this pin is configured as SIOGNT#. When asserted, SIOGNT# indicates that the external PCI arbiter has granted use of the bus to the SIO/SIO.A. This pin has a weak internal pull-up resistor.
REQ1#	I	**REQUEST 1:** If the SIO/SIO.A internal arbiter is enabled through the Arbiter Configuration Register, then this signal is configured as REQ1#. An active low assertion indicates that Initiator1 desires the use of the PCI Bus. If the internal arbiter is disabled, the SIO/SIO.A ignores REQ1# after reset. This pin has a weak internal pull-up resistor.

Signal Name	Type	Description
CPUGNT#	t/s/o	**CPU GRANT:** If the SIO/SIO.A internal arbiter is enabled, this pin is configured as CPUGNT#. The SIO/SIO.A internal arbiter asserts CPUGNT# to indicate that the CPU initiator has been granted the PCI Bus. If the internal arbiter is disabled, this signal is meaningless. CPUGNT# is tri-stated from the leading edge of PCIRST#. CPUGNT# is tri-stated until driven by the SIO/SIO.A. CPUGNT# is driven based on the arbitration scheme and the asserted REQx#s.
GNT0#/ SIOREQ#	t/s/o	**GRANT 0/SIO REQUEST:** If the SIO/SIO.A internal arbiter is enabled, this pin is configured as GNT0#. The SIO/SIO.A internal arbiter asserts GNT0# to indicate that Initiator0 has been granted the PCI Bus. If the internal arbiter is disabled, this pin is configured as SIOREQ#. The SIO/SIO.A asserts SIOREQ# to request the PCI Bus. GNT0#/SIOREQ# is tri-stated from the leading edge of PCIRST#. GNT0#/SIOREQ# is tri-stated until driven by the SIO/SIO.A. GNT0#/SIOREQ# is driven based on the arbitration scheme and the asserted REQx#'s.
GNT1#/ RESUME#	t/s/o	**GRANT 1/RESUME:** If the SIO's internal arbiter is enabled, this pin is configured as GNT1#. The SIO/SIO.A internal arbiter asserts GNT1# to indicate that Initiator1 has been granted the PCI Bus. If the internal arbiter is disabled, this pin is configured as RESUME#. The SIO/SIO.A asserts RESUME# to indicate that the conditions causing the SIO/SIO.A to retry the cycle has passed. GNT1#/RESUME# is tri-stated from the leading edge of PCIRST#. GNT1#/RESUME# is tri-stated until driven by the SIO/SIO.A. GNT1#/RESUME# is driven based on the arbitration scheme and the asserted REQx#'s.
REQ2#	I	**REQUEST 2:** This pin is an active low signal that indicates that Initiator2 desires the use of the PCI Bus. This signal has a weak internal pull-up resistor.
REQ3#	I	**REQUEST 3:** This pin is an active low signal that indicates that Initiator3 desires the use of the PCI Bus. This signal has a weak internal pull-up resistor.
GNT2#	t/s/o	**GRANT 2:** This pin is configured as GNT2#. The SIO/SIO.A internal arbiter asserts GNT2# to indicate that Initiator2 has been granted the PCI Bus. GNT2# is high upon reset.
GNT3#	t/s/o	**GRANT 3:** This pin is configured as GNT3#. The SIO/SIO.A internal arbiter asserts GNT3# to indicate that Initiator3 has been granted the PCI Bus. GNT3# is high upon reset.

Signal Name	Type	Description
MEMREQ#	t/s/o	**MEMORY REQUEST:** If the SIO/SIO.A is configured in Guaranteed Access Time (GAT) Mode, MEMREQ# will be asserted when an ISA master or DMA is requesting the ISA Bus (along with FLSHREQ#) to indicate that the SIO/SIO.A requires ownership of the main memory. MEMREQ# is tri-stated from the leading edge of PCIRST#. MEMREQ# remains tri-stated until driven by the SIO/SIO.A. After PCIRST, MEMREQ# is driven inactive asynchronously from PCIRST# inactive. The SIO/SIO.A asserts FLSHREQ# concurrently with asserting MEMREQ#. **FLSHREQ# MEMREQ# Meaning** 1 1 Idle 0 1 Flush buffers pointing towards PCI to avoid ISA deadlock 1 0 82378ZB. Reserved 82379AB. GAT enabled or disabled: For buffer coherency in APIC systems, the buffers pointing to main memory must be flushed and disabled for the duration of assertion. 0 0 GAT mode. Guarantee PCI Bus immediate access to main memory (this may or may not require the PCI-to-main memory buffers to be flushed first depending on the number of buffers).
FLSHREQ#	t/s/o	**FLUSH REQUEST:** FLSHREQ# is generated by the SIO/SIO.A to command all of the system's posted write buffers pointing towards the PCI Bus to be flushed. This is required before granting the ISA Bus to an ISA master or the DMA. FLSHREQ# is tri-stated from the leading edge of PCIRST#. FLSHREQ# remains tri-stated until driven by the SIO/SIO.A. After PCIRST, FLSHREQ# is driven inactive asynchronously from PCIRST# inactive.
MEMACK#	I	**MEMORY ACKNOWLEDGE:** MEMACK# is the response handshake that indicates to the SIO/SIO.A that the function requested over the MEMREQ# and/or FLSHREQ# signals has been completed. In GAT mode (MEMREQ# and FLSHREQ# asserted), the main memory bus is dedicated to the PCI Bus and the system's posted write buffers pointing towards the PCI Bus have been flushed and are disabled. In non-GAT mode (FLSHREQ# asserted alone), this means the system's posted write buffers have been flushed and are disabled. In either case, the SIO/SIO.A can now grant the ISA Bus to the requester.

2.3. Address Decoder Signal

Signal Name	Type	Description
MEMCS#	O	**MEMORY CHIP SELECT.** MEMCS# is a programmable address decode signal provided to a Host CPU bridge. A CPU bridge can use MEMCS# to forward a PCI cycle to main memory behind the bridge. MEMCS# is driven one PCI clock after FRAME# is sampled active (address phase) and is valid for one clock cycle before going inactive. MEMCS# is high upon reset.

2.4. Power Management Signals

Signal Name	Type	Description
SMI#	O	**SYSTEM MANAGEMENT INTERRUPT:** SMI# is asserted by the 82379AB in response to one of many enableable hardware or software events. This signal is driven low during a hard reset (PCIRST# asserted) and driven high when PCIRST# is negated.
STPCLK#	O	**STOP CLOCK:** STPCLK# is asserted by the 82379AB in response to one of many enableable hardware or software events. STPCLK# connects directly to the CPU. This signal is driven low during a hard reset (PCIRST# asserted) and driven high when PCIRST# is negated.
EXTSMI#	I	**EXTERNAL SYSTEM MANAGEMENT INTERRUPT:** EXTSMI# is a falling edge triggered input to the 82379AB indicating that an external device is requesting the system to enter SMM mode. This pin includes a weak internal pull-up resistor.
INIT	I	**INIT:** INIT is an input to the SIO/SIO.A indicating that the CPU is actually being soft reset. It is connected to the INIT pin of the CPU. This pin includes a weak internal pull-up resistor.

2.5. ISA Interface Signals

Signal Name	Type	Description
AEN	O	**ADDRESS ENABLE:** AEN is asserted during DMA cycles to prevent I/O slaves from misinterpreting DMA cycles as valid I/O cycles. This signal is also driven high during refresh cycles. AEN is driven low upon reset.
BALE	O	**BUS ADDRESS LATCH ENABLE:** BALE is an active high signal asserted by the SIO/SIO.A to indicate that the address (SA[19:0], LA[23:17]), AEN and SBHE# signal lines are valid. The LA[23:17] address lines are latched on the trailing edge of BALE. BALE remains asserted throughout DMA and ISA master cycles. BALE is driven low upon reset.
SYSCLK	O	**SYSTEM CLOCK:** SYSCLK is an output of the SIO/SIO.A component. The frequencies supported are 6 to 8.33 MHz.
IOCHRDY	I/O	**I/O CHANNEL READY:** Resources on the ISA Bus assert IOCHRDY to indicate that additional time (wait states) is required to complete the cycle. IOCHRDY is tri-stated upon reset.
IOCS16#	I	**16-BIT I/O CHIP SELECT:** This signal is driven by I/O devices on the ISA Bus to indicate that they support 16-bit I/O bus cycles.
IOCHK#	I	**I/O CHANNEL CHECK:** IOCHK# can be driven by any resource on the ISA Bus. When asserted, it indicates that a parity or an un-correctable error has occurred for a device or memory on the ISA Bus. A NMI will be generated to the CPU if the NMI generation is enabled.
IOR#	I/O	**I/O READ:** IOR# is the command to an ISA I/O slave device that the slave may drive data on to the ISA data bus (SD[15:0]). IOR# is driven high upon reset.
IOW#	I/O	**I/O WRITE:** IOW# is the command to an ISA I/O slave device that the slave may latch data from the ISA data bus (SD[15:0]). IOW# is driven high upon reset.

Signal Name	Type	Description
LA[23:17]	I/O	**UNLATCHED ADDRESS:** These address lines allow accesses to physical memory on the ISA Bus up to 16 Mbytes. The LA[23:17] signals are at an unknown state upon reset. For the 82378ZB, these signals are undefined during DMA type "A", "B", and "F" cycles.
SA[19:0]	I/O	**SYSTEM ADDRESS BUS:** These bi-directional address lines define the selection with the granularity of one byte within the one Mbyte section of memory defined by the LA[23:17] address lines. The address lines SA[19:17] that are coincident with LA[19:17] are defined to have the same values as LA[19:17] for all memory cycles. For I/O accesses, only SA[15:0] are used. SA[19:0] are outputs when the SIO/SIO.A owns the ISA Bus. SA[19:0] are inputs when an external ISA Master owns the ISA Bus. SA[19:0] are at an unknown state upon reset. For the 82378ZB, SA[19:0] are undefined during DMA type "A", "B", or "F" cycles.
SBHE#	I/O	**SYSTEM BYTE HIGH ENABLE:** SBHE# indicates, when asserted, that a byte is being transferred on the upper byte (SD[15:8]) of the data bus. SBHE# is negated during refresh cycles. SBHE# is at an unknown state upon reset.
MEMCS16#	OD	**MEMORY CHIP SELECT 16:** MEMCS16# is a decode of LA[23:17] without any qualification of the command signal lines. ISA slaves that are 16-bit memory devices drive this signal low. The SIO/SIO.A drives this signal low during ISA master to PCI memory cycles. MEMCS16# is at an unknown state upon reset.
MASTER# (82378ZB Only)	I	**MASTER:** An ISA Bus master asserts MASTER# to indicate that it has control of the ISA Bus. Before the ISA master can assert MASTER#, it must first sample DACK# active. Once MASTER# is asserted, the ISA master has control of the ISA Bus until it negates MASTER#.
MEMR#	I/O	**MEMORY READ:** MEMR# is the command to a memory slave that it may drive data onto the ISA data bus. MEMR# is an output when the SIO/SIO.A is a master on the ISA Bus. MEMR# is an input when an ISA master, other than the SIO/SIO.A, owns the ISA Bus. This signal is also driven by the SIO/SIO.A during refresh cycles. For compatible timing mode DMA cycles, the SIO/SIO.A, as a master, asserts MEMR# if the address is less than 16 Mbytes. This signal is not generated for accesses to addresses greater than 16 Mbytes. For the 82378ZB, MEMR# is not driven active during DMA type "A", "B", or "F" cycles.
MEMW#	I/O	**MEMORY WRITE:** MEMW# is the command to a memory slave that it may latch data from the ISA data bus. MEMW# is an output when the SIO/SIO.A owns the ISA Bus. MEMW# is an input when an ISA master, other than the SIO/SIO.A, owns the ISA Bus. For compatible timing mode DMA cycles, the SIO/SIO.A, as a master, asserts MEMW# if the address is less than 16 Mbytes. This signal is not generated for accesses to addresses greater than 16 Mbytes. For the 82378ZB, MEMW# is not driven active during DMA type "A", "B", or "F" cycles.

Signal Name	Type	Description
SMEMW#	O	**SYSTEM MEMORY WRITE:** The SIO/SIO.A asserts SMEMW# to request a memory slave to accept data from the data lines. If the access is below the 1 Mbyte range (00000000–000FFFFFh) during DMA compatible, SIO/SIO.A master, or ISA master cycles, the SIO/SIO.A asserts SMEMW#. SMEMW# is a delayed version of MEMW#. SMEMW# is driven high upon reset.
SMEMR#	O	**SYSTEM MEMORY READ:** The SIO/SIO.A asserts SMEMR# to request a memory slave to accept data from the data lines. If the access is below the 1 Mbyte range (00000000–000FFFFFh) during DMA compatible, SIO/SIO.A master, or ISA master cycles, the SIO/SIO.A asserts SMEMR#. SMEMR# is a delay version of MEMR#. Upon PCIRST# this signal is low. SMEMR# is driven high upon reset.
ZEROWS#	I	**ZERO WAIT STATES:** An ISA slave asserts ZEROWS# after its address and command signals have been decoded to indicate that the current cycle can be shortened. A 16-bit ISA memory cycle can be reduced to two SYSCLKs. An 8-bit memory or I/O cycle can be reduced to three SYSCLKs. ZEROWS# has no effect during 16-bit I/O cycles. If IOCHRDY and ZEROWS# are both asserted during the same clock, then ZEROWS# is ignored and wait states are added as a function of IOCHRDY (i.e., IOCHRDY has precedence over ZEROWS#).
OSC	I	**OSCILLATOR:** OSC is the 14.31818 MHz ISA clock signal. It is used by the internal 8254 Timer, counters 0, 1, and 2.
RSTDRV	O	**RESET DRIVE:** The SIO/SIO.A asserts RSTDRV to reset devices that reside on the ISA Bus. The SIO/SIO.A asserts this signal when PCIRST# (PCI Reset) is asserted. In addition, the SIO/SIO.A can be programmed to assert RSTDRV by writing to the ISA Clock Divisor Register. Software should assert the RSTDRV during configuration to reset the ISA Bus when changing the clock divisor. Note that when RSTDRV is generated via the ISA Clock Divisor Register, software must ensure that RSTDRV is driven active for a minimum of 1 µs.
SD[15:0]	I/O	**SYSTEM DATA:** SD[15:0] provide the 16-bit data path for devices residing on the ISA Bus. SD[15:8] correspond to the high order byte and SD[7:0] correspond to the low order byte. SD[15:0] are undefined during refresh. The SIO/SIO.A tri-states SD[15:0] during reset.

2.6. DMA Signals

Signal Name	Type	Description
DREQ [3:0,7:5]	I	**DMA REQUEST:** The DREQ lines are used to request DMA service from the SIO/SIO.A DMA controller or for a 16-bit master to gain control of the ISA expansion bus. The active level (high or low) is programmed via the DMA Command Register (bit 6). The request must remain active until the appropriate DACK signal is asserted.
DACK# [3:0,7:5]	O	**DMA ACKNOWLEDGE:** The DACK output lines indicate that a request for DMA service has been granted by the SIO/SIO.A or that a 16-bit master has been granted the bus. The active level (high or low) is programmed via the DMA Command Register (bit 7). These lines should be used to decode the DMA slave device with the IOR# or IOW# line to indicate selection. Upon PCIRST#, these lines are set inactive (high).

Signal Name	Type	Description
EOP (82378ZB)	I/O	**END OF PROCESS:** EOP is bi-directional, acting in one of two modes, and is directly connected to the TC line of the ISA Bus. DMA slaves assert EOP to the SIO/SIO.A to terminate DMA cycles. The SIO/SIO.A asserts EOP to DMA slaves as a terminal count indicator. **EOP-IN MODE:** For all transfer types during DMA, the SIO/SIO.A samples EOP. If it is sampled asserted, the transfer is terminated. **TC-OUT MODE:** The SIO asserts EOP after a new address has been output, if the byte count expires with that transfer. The EOP (TC) remains asserted until AEN is negated, unless AEN is negated during an autoinitialization. EOP (TC) is negated before AEN is negated during an autoinitialization. When all the DMA channels are not in use, the EOP signal is in output mode and negated (low). After PCIRST#, EOP is in output mode and inactive.
TC (82379AB)	O	**TERMINAL COUNT:** The SIO.A asserts TC after a new address has been output, if the byte count expires with that transfer. TC remains asserted until AEN is negated, unless AEN is negated during an autoinitialization. TC is negated before AEN is negated during an autoinitialization. After PCIRST#, EOP is in output mode and inactive.
REFRESH#	I/O	**REFRESH:** As an output, REFRESH# is used by the SIO/SIO.A to indicate when a refresh cycle is in progress. It should be used to enable the SA[15:0] address to the row address inputs of all banks of dynamic memory on the ISA Bus. Thus, when MEMR# is asserted, the entire expansion bus dynamic memory is refreshed. Memory slaves must not drive any data onto the bus during refresh. As an output, this signal is driven directly onto the ISA Bus. This signal is an output only when the SIO/SIO.A DMA refresh is a master on the bus responding to an internally generated request for refresh. As an input, REFRESH# is driven by 16-bit ISA Bus masters to initiate refresh cycles. Upon PCIRST#, this signal is tri-stated.

2.7. Timer Signal

Signal Name	Type	Description
SPKR (82378ZB)	O	**SPEAKER DRIVE:** The SPKR signal, in the 82378ZB and 82379AB, is the output of counter 2 and has a 24 mA drive capability. Upon reset, its output state is 0.
SPKR/TESTO (82379AB)		For the 82379AB in test mode, this pin is the output pin used during NAND tree testing.

2.8. Interrupt Controller Signals

Signal Name	Type	Description
IRQ[15,14,11:9, 7:3,1]	I	**INTERRUPT REQUEST:** The IRQ signals provide both system board components and ISA Bus I/O devices with a mechanism for asynchronously interrupting the CPU. The assertion mode of these inputs (edge or level triggered) is programmable via the ELCR Register. Upon PCIRST#, the IRQ lines are placed in edge-triggered mode. **NOTE:** 1. For the 82378ZB, an active IRQ input must remain asserted until after the interrupt is acknowledged. If the input goes inactive before this time, a DEFAULT IRQ7 occurs when the CPU acknowledges the interrupt. 2. For the 82379AB, A low to high transition on IRQ1 is latched by the 82379AB. The latch is cleared by a read of address 60h. 3. Refer to the Utility Bus Signal descriptions for IRQ12 and IRQ13 signal descriptions.
IRQ8#	I	**INTERRUPT REQUEST EIGHT SIGNAL:** IRQ8# is an active low interrupt input. The assertion mode of these inputs (edge or level triggered) is programmable via the ELCR Register. Upon PCIRST#, the IRQ lines are placed in edge-triggered mode. **NOTE:** 1. For the 82378ZB, IRQ8# must remain asserted until after the interrupt is acknowledged. If the input goes inactive before this time, a DEFAULT IRQ7 will occur when the CPU acknowledges the interrupt. 2. For the 82379AB, this pin requires an external 8.2 KΩ pull-up resistor.
INT	O	**CPU INTERRUPT:** INT is driven by the SIO/SIO.A to signal the CPU that an interrupt request is pending and needs to be serviced. It is asynchronous with respect to SYSCLK or PCICLK and is always an output. The interrupt controller must be programmed following a reset to ensure that INT is at a known state. Upon PCIRST#, INT is driven low.
NMI	O	**NON-MASKABLE INTERRUPT:** NMI is used to force a non-maskable interrupt to the CPU. The SIO/SIO.A generates an NMI when either SERR# or IOCHK# is asserted, depending on how the NMI Status and Control Register is programmed. The CPU detects an NMI when it detects a rising edge on NMI. After the NMI interrupt routine processes the interrupt, the NMI status bits in the NMI Status and Control Register are cleared by software. The NMI interrupt routine must read this register to determine the source of the interrupt. The NMI is reset by setting the corresponding NMI source enable/disable bit in the NMI Status and Control Register. To enable NMI interrupts, the two NMI enable/disable bits in the register must be set to 0, and the NMI mask bit in the NMI Enable/Disable and Real-Time Clock Address Register must be set to 0. Upon PCIRST#, this signal is driven low.

2.9. APIC Bus Signals (82379AB Only)

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. APICD1 has a weak pull-down resistor. These signals require external pull-ups (330 Ω recommended) to the 3.3V rail. These signals are tri-stated during a hard reset.

2.10. Utility Bus Signals

Signal Name	Type	Description
UBUSTR	O	**UTILITY DATA BUS TRANSMIT/RECEIVE:** UBUSTR is tied directly to the direction control of a 74F245 that buffers the utility data bus, UD[7:0]. UBUSTR is asserted for all I/O read cycles (regardless if a utility bus device has been decoded). UBUSTR is asserted for memory cycles only if BIOS space has been decoded. For PCI and ISA master-initiated read cycles, UBUSTR is asserted from the falling edge of either IOR# or MEMR#, depending on the cycle type (driven from MEMR# only if BIOS space has been decoded). When the rising edge of IOR# or MEMR# occurs, the SIO/SIO.A negates UBUSTR. For DMA read cycles from the Utility Bus, UBUSTR is asserted when DACKx# is asserted and negated when DACKx# is negated. At all other times, UBUSTR is negated. Upon PCIRST#, this signal is driven low.
UBUSOE#	O	**UTILITY DATA BUS OUTPUT ENABLE:** UBUSOE# is tied directly to the output enable of a 74F245 that buffers the utility data bus, UD[7:0], from the system data bus, SD[7:0]. UBUSOE# is asserted anytime a SIO/SIO.A supported Utility Bus device is decoded, and the devices decode is enabled in the Utility Bus Chip Select Enable Registers. UBUSOE# is asserted from the falling edge of the ISA commands (IOR#, IOW#, MEMR#, or MEMW#) for PCI and ISA master-initiated cycles. UBUSOE# is negated from the rising edge of the ISA command signals for SIO/SIO.A-initiated cycles and the SA[16:0] and LA[23:17] address for ISA master-initiated cycles. For DMA cycles, UBUSOE# is asserted when DACK2# is asserted and negated when DACK2# negated. UBUSOE# is not driven active under the following conditions: **NOTES:** 1. During an I/O access to the floppy controller, if DSKCHG is sampled low at reset. 2. If the Digital Output Register is programmed to ignore DACK2#. 3. During an I/O read access to floppy location 3F7h (primary) or 377h (secondary), if theIDE decode space is disabled (i.e., IDE is not resident on the Utility Bus). 4. During any access to a utility bus peripheral in which its decode space has been disabled. Upon a PCIRST#, this signal is driven inactive (high).
ECSADDR [2:0]	O	**ENCODED CHIP SELECTS:** ECSADDR[2:0] are the encoded chip selects and/or control signals for the Utility Bus peripherals supported by the SIO/SIO.A. The binary code formed by the three signals indicates which Utility Bus device is selected. These signals tie to the address inputs of two external 74F138 decoder chips and are driven valid/invalid from the SA[16:0] and LA[23:17] address lines. Upon PCIRST#, these signals are driven high.

<table>
<tr><th>Signal Name</th><th>Type</th><th>Description</th></tr>
<tr><td>ECSEN#</td><td>O</td><td>ENCODED CHIP SELECT ENABLE: ECSEN# is used to determine which of the two external 74F138 decoders is to be selected. ECSEN# is driven low to select decoder 1 and driven high to select decoder 2. This signal is driven valid/invalid from the SA[16:0] and LA[23:17] address lines (except for the generation of RTCALE#, in which case, ECSEN# is driven active based on IOW# falling, and remains active for two SYSCLKs). During a non-valid address or during an access not targeted for the Utility Bus, this signal is driven high. Upon PCIRST#, this signal is driven high.</td></tr>
<tr><td>ALT_RST#</td><td>O</td><td>ALTERNATE RESET: ALT_RST# is used to reset the CPU under program control. This signal is AND'ed together externally with the reset signal (KBDRST#) from the keyboard controller to provide a software means of resetting the CPU. This provides a faster means of reset than is provided by the keyboard controller. Writing a 1 to bit 0 in the Port 92 Register causes this signal to pulse low for approximately 4 SYSCLKs. Before another ALT_RST# pulse can be generated, bit 0 must be set to 0. Upon PCIRST#, this signal is driven inactive high (bit 0 in the Port 92 Register is set to 0).</td></tr>
<tr><td>ALT_A20</td><td>O</td><td>ALTERNATE A20: ALT_A20 is used to force A20M# to the CPU low for support of real mode compatible software. This signal is externally OR'ed with the A20GATE signal from the keyboard controller and CPURST to control the A20M# input of the CPU. Writing a 0 to bit 1 of the Port 92 Register forces ALT_A20 low. ALT_A20 low drives A20M# to the CPU low, if A20GATE from the keyboard controller is also low. Writing a 1 to bit 1 of the Port 92 Register force ALT_A20 high. ALT_A20 high drives A20M# to the CPU high, regardless of the state of A20GATE from the keyboard controller. Upon reset, this signal is driven low.</td></tr>
<tr><td>DSKCHG</td><td>I</td><td>DISK CHANGE: DSKCHG is tied directly to the DSKCHG signal of the floppy controller. This signal is inverted and driven on SD7 during I/O read cycles to floppy address locations 3F7h (primary) or 377h (secondary) as shown in the table below. Note that the primary and secondary locations are programmed in the Utility Bus Address Decode Enable/Disable Register "A".
<table>
<tr><th>FLOPPYCS# Decode</th><th>IDECSx# Decode</th><th>State of SD7 (output)</th><th>State of UBUSOE#</th></tr>
<tr><td>Enabled</td><td>Enabled</td><td>Tri-stated</td><td>Enabled</td></tr>
<tr><td>Enabled</td><td>Disabled</td><td>Driven via DSKCHG</td><td>Disabled</td></tr>
<tr><td>Disabled</td><td>Enabled</td><td>Tri-stated</td><td>Enabled (note)</td></tr>
<tr><td>Disabled</td><td>Disabled</td><td>Tri-stated</td><td>Disabled</td></tr>
</table>
NOTE:
For this mode to be supported, extra logic is required to disable the U-Bus transceiver for accesses to 3F7/377. This is necessary because of potential contention between the Utility Bus buffer and a floppy on the ISA Bus driving the system bus at the same time during shared I/O accesses.
This signal is also used to determine if the floppy controller is present on the Utility Bus. It is sampled on the trailing edge of PCIRST#, and if high, the floppy is present. For systems that do not support a floppy via the SIO/SIO.A, this pin should be strapped low. If sampled low, the SD7 function, UBUSOE#, and ECSADDR[2:0] signals will not be enabled for DMA or programmed I/O accesses to the floppy disk controller. This condition overrides the floppy decode enable bits in the Utility Bus Chip Select A.</td></tr>
</table>

Signal Name	Type	Description
FERR#/ IRQ13	I	**NUMERIC COPROCESSOR ERROR/IRQ13:** This signal has two separate functions, depending on bit 5 in the ISA Clock Divisor Register. This pin functions as a FERR# signal supporting coprocessor errors, if this function is enabled (bit 5=1), or as an external IRQ13, if the coprocessor error function is disabled (bit 5=0). If programmed to support coprocessor error reporting, this signal is tied to the coprocessor error signal on the CPU. If FERR# is asserted by the coprocessor inside the CPU, the SIO/SIO.A generates an internal IRQ13 to its interrupt controller unit. The SIO/SIO.A then asserts the INT output to the CPU. Also, in this mode, FERR# gates the IGNNE# signal to ensure that IGNNE# is not asserted to the CPU unless FERR# is active. When FERR# is asserted, the SIO/SIO.A asserts INT to the CPU as an IRQ13. IRQ13 continues to be asserted until a write to F0h has been detected. If the coprocessor error reporting is disabled, FERR# can be used by the system as IRQ13. Upon PCIRST#, this signal provides the standard IRQ13 function. This signal should be pulled high with an external 8.2K Ω pull-up resistor if the IRQ13 mode is used or the pin is left floating.
IGNNE#	O	**IGNORE ERROR:** This signal is connected to the ignore error pin of the CPU. IGNNE# is only used if the SIO/SIO.A coprocessor error reporting function is enabled in the ISA Clock Divisor Register (bit 5=1). If FERR# is active, indicating a coprocessor error, a write to the Coprocessor Error Register (F0h) causes the IGNNE# to be asserted. IGNNE# remains asserted until FERR# is negated. If FERR# is not asserted when the Coprocessor Error Register is written, the IGNNE# is not asserted. IGNNE# is driven high upon a reset.
IRQ12/M	I	**INTERRUPT REQUEST/MOUSE INTERRUPT:** In addition to providing the standard interrupt function as described in the pin description for IRQ[15,14, 11:9, 7:3, 1], this pin also provides a mouse interrupt function. Bit 4 in the ISA Clock Divisor Register determines the functionality of IRQ12/M. When the mouse interrupt function is selected, a low-to-high transition on this signal is latched by the SIO/SIO.A and an INT is generated to the CPU as IRQ12. An interrupt will continue to be generated until a PCIRST# or an I/O read access to address 60h (falling edge of IOR#) is detected. After a PCIRST#, this pin provides the standard IRQ12 function.

2.11. Test Signals

Signal Name	Type	Description
TEST	I	**TEST:** The TEST signal is used to tri-state all of the SIO/SIO.A outputs. During normal operation, this input should be tied to ground.
TESTO (82378ZB) SPKR/TESTO (82379AB)	O	**TEST OUTPUT:** For both the 82378ZB and 82379AB, this is the output pin used during NAND tree testing. For the 82379AB, when not in test mode, this pin is the speaker output.

3.0. REGISTER DESCRIPTION

The SIO/SIO.A contains PCI configuration Registers and ISA-Compatible Registers. In addition, the SIO.A contains I/O APIC Registers. Some of the SIO/SIO.A configuration and ISA-Compatible Registers contain reserved bits. These bits are labeled "Reserved". Software must take care to deal correctly with bit-encoded fields that are reserved. On reads, software must use appropriate masks to extract the defined bits and not rely on reserved bits being any particular value. On writes, software must ensure that the values of reserved 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 the data then written back.

In addition to reserved bits within a register, the SIO/SIO.A contains address locations in the PCI configuration space that are marked "Reserved" (Table 3). The SIO/SIO.A responds to accesses to these address locations by completing the PCI cycle. However, reads of reserved address locations yield all zeroes and writes have no effect on the SIO/SIO.A.

The SIO/SIO.A, upon receiving a hard reset (PCIRST# signal), sets its internal registers to pre-determined **default** states. The default values are indicated in the individual register descriptions.

Configuration Registers

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

ISA-Compatible Registers

The ISA-Compatible Registers include DMA Registers, Timer Registers, Interrupt Controller Registers, and Non-Maskable Interrupt and Utility Bus Support Registers (Table 6). All of these registers are accessible from the PCI Bus. In addition, some of the registers are accessible from the ISA Bus. Except for the BIOS timer Registers, the ISA-Compatible Registers can only be accessed as byte quantities. If a PCI master attempts a multi-byte access (i.e., more than one Byte Enable signal asserted), the SIO/SIO.A responds with a target-abort. The BIOS Timer Register can be accessed as Byte, Word, or Dword quantities.

In general, PCI accesses to the ISA-Compatible Registers will not be broadcast to the ISA Bus. However, PCI accesses to addresses 70h, 60h, 92h, 3F2h, 372h, and F0h are exceptions. Read and write accesses to these SIO/SIO.A locations are broadcast onto the ISA Bus. PCI master accesses to SIO/SIO.A Registers will be retried if the ISA Bus is owned by an ISA master or the DMA controller. Accesses to the BIOS Timer Register are broadcast to the ISA Bus only if this register is disabled. If this register is enabled, the cycle is not broadcast to the ISA Bus.

I/O APIC Registers (82379AB Only)

The APIC 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 1). 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 2). 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 1. Memory Address For Accessing APIC Registers

Memory Address	Mnemonic	Register Name	Type
FEC0 xy00h	IOREGSEL	I/O Register Select	R/W
FEC0 xy10h	IOWIN	I/O Window	R/W

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

Table 2. I/O APIC Registers

Address Offset	Mnemonic	Register Name	Type
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].

Table 3. Configuration Registers

Configuration Offset	Register	Register Access	Bus Access
00–01h	Vendor Identification	RO	PCI Only
02–03h	Device Identification	RO	PCI Only
04–05h	Command	R/W	PCI Only
06–07h	Device Status	R/W	PCI Only
08h	Revision Identification	RO	PCI Only
09–3Fh	Reserved	--	PCI Only
40h	PCI Control	R/W	PCI Only
41h	PCI Arbiter Control	R/W	PCI Only
42h	PCI Arbiter Priority Control	R/W	PCI Only
43h	PCI Arbiter Priority Control Extension Register	R/W	PCI Only
44h	MEMCS# Control	R/W	PCI Only
45h	MEMCS# Bottom of Hole	R/W	PCI Only
46h	MEMCS# Top of Hole	R/W	PCI Only
47h	MEMCS# Top of Memory	R/W	PCI Only
48h	ISA Address Decoder Control	R/W	PCI Only

Configuration Offset	Register	Register Access	Bus Access
49h	ISA Address Decoder ROM Block Enable	R/W	PCI Only
4Ah	ISA Address Decoder Bottom of Hole	R/W	PCI Only
4Bh	ISA Address Decoder Top of Hole	R/W	PCI Only
4Ch	ISA Controller Recovery Timer	R/W	PCI Only
4Dh	ISA Clock Divisor	R/W	PCI Only
4Eh	Utility Bus Chip Select Enable A	R/W	PCI Only
4Fh	Utility Bus Chip Select Enable B	R/W	PCI Only
50–53h	Reserved	--	PCI Only
54h	MEMCS# Attribute Register #1	R/W	PCI Only
55h	MEMCS# Attribute Register #2	R/W	PCI Only
56h	MEMCS# Attribute Register #3	R/W	PCI Only
57h	Reserved (82379AB)	--	PCI Only
57h	S/G Relocation Base Address (82378ZB)	R/W	PCI Only
58–5Fh	Reserved	--	PCI Only
60h	PIRQ0# Route Control	R/W	PCI Only
61h	PIRQ1# Route Control	R/W	PCI Only
62h	PIRQ2# Route Control	R/W	PCI Only
63h	PIRQ3# Route Control	R/W	PCI Only
64–6Fh	Reserved	--	PCI Only
70h	Reserved (82378ZB)	--	PCI Only
70h	PIC/APIC Configuration Control (82379AB)	WO	PCI Only
71h	Reserved (82378ZB)	--	PCI Only
71h	APIC Base Address Relocation (82379AB)	WO	PCI Only
72–7Fh	Reserved	--	PCI Only
80–81h	BIOS Timer Base Address	R/W	PCI Only
82–9Fh	Reserved	--	PCI Only
A0h	SMI Control (SMICNTL)	R/W	PCI Only
A1h	Reserved	--	PCI Only
A2h - A3h	SMI Enable (SMIEN)	R/W	PCI Only
A4h - A7h	System Event Enable (SEE)	R/W	PCI Only

Configuration Offset	Register	Register Access	Bus Access
A8h	Fast-Off Timer (FTMR)	R/W	PCI Only
A9h	Reserved	--	PCI Only
AAh - ABh	SMI Request (SMIREQ)	R/W	PCI Only
ACh	Clock Throttle STPCLK# Low Timer (CTLTMRL)	R/W	PCI Only
ADh	Reserved	--	PCI Only
AEh	Clock Throttle STPCLK# High Timer (CTLTMRH)	R/W	PCI Only
AF–FFh	Reserved	--	PCI Only

Table 4. ISA-Compatible Registers

Address	Address				Type	Name	Block
	FEDC	BA98	7654	3210			
0000h	0000	0000	000x	0000	r/w	DMA1 CH0 Base and Current Address	DMA
0001h	0000	0000	000x	0001	r/w	DMA1 CH0 Base and Current Count	DMA
0002h	0000	0000	000x	0010	r/w	DMA1 CH1 Base and Current Address	DMA
0003h	0000	0000	000x	0011	r/w	DMA1 CH1 Base and Current Count	DMA
0004h	0000	0000	000x	0100	r/w	DMA1 CH2 Base and Current Address	DMA
0005h	0000	0000	000x	0101	r/w	DMA1 CH2 Base and Current Count	DMA
0006h	0000	0000	000x	0110	r/w	DMA1 CH3 Base and Current Address	DMA
0007h	0000	0000	000x	0111	r/w	DMA1 CH3 Base and Current Count	DMA
0008h	0000	0000	000x	1000	r/w	DMA1 Status(r) Command(w) Register	DMA
0009h	0000	0000	000x	1001	wo	DMA1 Write Request Register	DMA
000Ah	0000	0000	000x	1010	wo	DMA1 Write Single Mask Bit	DMA
000Bh	0000	0000	000x	1011	wo	DMA1 Write Mode Register	DMA
000Ch	0000	0000	000x	1100	wo	DMA1 Clear Byte Pointer	DMA
000Dh	0000	0000	000x	1101	wo	DMA1 Master Clear	DMA
000Eh	0000	0000	000x	1110	wo	DMA1 Clear Mask Register	DMA
000Fh	0000	0000	000x	1111	r/w	DMA1 Read/Write All Mask Register Bits	DMA
0020h	0000	0000	001x	xx00	r/w	INT 1 Control Register	PIC
0021h	0000	0000	001x	xx01	r/w	INT 1 Mask Register	PIC
0040h	0000	0000	010x	0000	r/w	Timer Counter 1 - Counter 0 Count	TC
0041h	0000	0000	010x	0001	r/w	Timer Counter 1 - Counter 1 Count	TC

Address	Address				Type	Name	Block
	FEDC	BA98	7654	3210			
0042h	0000	0000	010x	0010	r/w	Timer Counter 1 - Counter 2 Count	TC
0043h	0000	0000	010x	0011	wo	Timer Counter 1 Command Mode Register	TC
0060h[2]	0000	0000	0110	0000	ro	Reset UBus IRQ12	Control
0061h	0000	0000	0110	0xx1	r/w	NMI Status and Control	Control
0070h[2]	0000	0000	0111	0xx0	wo	CMOS RAM Address and NMI Mask Register	Control
0078–007Bh[3,4,5]	0000	0000	0111	10xx	r/w	BIOS Timer	TC
0080h[1]	0000	0000	100x	0000	r/w	DMA Page Register (Reserved)	DMA
0081h	0000	0000	100x	0001	r/w	DMA Channel 2 Page Register	DMA
0082h	0000	0000	1000	0010	r/w	DMA Channel 3 Page Register	DMA
0083h	0000	0000	100x	0011	r/w	DMA Channel 1 Page Register	DMA
0084h[1]	0000	0000	100x	0100	r/w	DMA Page Register (Reserved)	DMA
0085h[1]	0000	0000	100x	0101	r/w	DMA Page Register (Reserved)	DMA
0086h[1]	0000	0000	100x	0110	r/w	DMA Page Register (Reserved)	DMA
0087h	0000	0000	100x	0111	r/w	DMA Channel 0 Page Register	DMA
0088h[1]	0000	0000	100x	0100	r/w	DMA Page Register (Reserved)	DMA
0089h	0000	0000	100x	1001	r/w	DMA Channel 6 Page Register	DMA
008Ah	0000	0000	100x	1010	r/w	DMA Channel 7 Page Register	DMA
008Bh	0000	0000	100x	1011	r/w	DMA Channel 5 Page Register	DMA
008Ch[1]	0000	0000	100x	1100	r/w	DMA Page Register (Reserved)	DMA
008Dh[1]	0000	0000	100x	1101	r/w	DMA Page Register (Reserved)	DMA
008Eh[1]	0000	0000	100x	1110	r/w	DMA Page Register (Reserved)	DMA
008Fh	0000	0000	100x	1111	r/w	DMA Low Page Register Refresh	DMA
0090h[6]	0000	0000	100x	0000	r/w	DMA Page Register (Reserved)	DMA
0092h[2]	0000	0000	1001	0010	r/w	System Control Port	Control
0094h[6]	0000	0000	100x	0100	r/w	DMA Page Register (Reserved)	DMA
0095h[6]	0000	0000	100x	0101	r/w	DMA Page Register (Reserved)	DMA
0096h[6]	0000	0000	100x	0110	r/w	DMA Page Register (Reserved)	DMA
0098h[6]	0000	0000	100x	1000	r/w	DMA Page Register (Reserved)	DMA
009Ch[6]	0000	0000	100x	1100	r/w	DMA Page Register (Reserved)	DMA

Address	Address				Type	Name	Block
	FEDC	BA98	7654	3210			
009Dh[6]	0000	0000	100x	1101	r/w	DMA Page Register (Reserved)	DMA
009Eh[6]	0000	0000	100x	1110	r/w	DMA Page Register (Reserved)	DMA
009Fh	0000	0000	100x	1111	r/w	DMA low page Register Refresh	DMA
00A0h	0000	0000	101x	xx00	r/w	INT 2 Control Register	PIC
00A1h	0000	0000	101x	xx01	r/w	INT 2 Mask Register	PIC
00B2h	0000	0000	1011	0010	r/w	Advanced Power Management Control Port	PM
00B3h	0000	0000	1011	0011	r/w	Advanced Power Management Status Port	PM
00C0h	0000	0000	1100	000x	r/w	DMA2 CH0 Base and Current Address	DMA
00C2h	0000	0000	1100	001x	r/w	DMA2 CH0 Base and Current Count	DMA
00C4h	0000	0000	1100	010x	r/w	DMA2 CH1 Base and Current Address	DMA
00C6h	0000	0000	1100	011x	r/w	DMA2 CH1 Base and Current Count	DMA
00C8h	0000	0000	1100	100x	r/w	DMA2 CH2 Base and Current Address	DMA
00CAh	0000	0000	1100	101x	r/w	DMA2 CH2 Base and Current Count	DMA
00CCh	0000	0000	1100	110x	r/w	DMA2 CH3 Base and Current Address	DMA
00CEh	0000	0000	1100	111x	r/w	DMA2 CH3 Base and Current Count	DMA
00D0h	0000	0000	1101	000x	r/w	DMA2 Status(r) Command(w) Register	DMA
00D2h	0000	0000	1101	001x	wo	DMA2 Write Request Register	DMA
00D4h	0000	0000	1101	010x	wo	DMA2 Write Single Mask Bit	DMA
00D6h	0000	0000	1101	011x	wo	DMA2 Write Mode Register	DMA
00D8h	0000	0000	1101	100x	wo	DMA2 Clear Byte Pointer	DMA
00DAh	0000	0000	1101	101x	wo	DMA2 Master Clear	DMA
00DCh	0000	0000	1101	110x	wo	DMA2 Clear Mask Register	DMA
00DEh	0000	0000	1101	111x	r/w	DMA2 Read/Write All Mask Register Bits	DMA
00F0h[2]	0000	0000	1111	0000	wo	Coprocessor Error	Control
0372h[2]	0000	0011	0111	0010	wo	Secondary Floppy Disk Digital Output Reg.	Control
03F2h[2]	0000	0011	1111	0001	wo	Primary Floppy Disk Digital Output Reg.	Control
040Ah[3]	0000	0100	0000	1010	ro	S/G Interrupt Status (82378ZB Only)	DMA
040Bh	0000	0100	0000	1011	wo	DMA1 Extended Mode (82378ZB Only)	DMA
0410h[3,4]	0000	0100	0001	0000	wo	CH0 S/G Command (82378ZB Only)	DMA
0411h[3,4]	0000	0100	0001	0001	wo	CH1 S/G Command (82378ZB Only)	DMA

Address	Address				Type	Name	Block
	FEDC	BA98	7654	3210			
0412h[3,4]	0000	0100	0001	0010	wo	CH2 S/G Command (82378ZB Only)	DMA
0413h[3,4]	0000	0100	0001	0011	wo	CH3 S/G Command (82378ZB Only)	DMA
0415h[3,4]	0000	0100	0001	0101	wo	CH5 S/G Command (82378ZB Only)	DMA
0416h[3,4]	0000	0100	0001	0110	wo	CH6 S/G Command (82378ZB Only)	DMA
0417h[3,4]	0000	0100	0001	0111	wo	CH7 S/G Command (82378ZB Only)	DMA
0418h[3,4]	0000	0100	0001	1000	ro	CH0 S/G Status (82378ZB Only)	DMA
0419h[3,4]	0000	0100	0001	1001	ro	CH1 S/G Status (82378ZB Only)	DMA
041Ah[3,4]	0000	0100	0001	1010	ro	CH2 S/G Status (82378ZB Only)	DMA
041Bh[3,4]	0000	0100	0001	1011	ro	CH3 S/G Status (82378ZB Only)	DMA
041Dh[3,4]	0000	0100	0001	1101	ro	CH5 S/G Status (82378ZB Only)	DMA
041Eh[3,4]	0000	0100	0001	1110	ro	CH6 S/G Status (82378ZB Only)	DMA
041Fh[3,4]	0000	0100	0001	1111	ro	CH7 S/G Status (82378ZB Only)	DMA
0420h[3,4]	0000	0100	0010	00xx	r/w	CH0 S/G Descriptor Table Pointer (82378ZB Only)	DMA
0424h[3,4]	0000	0100	0010	01xx	r/w	CH1 S/G Descriptor Table Pointer (82378ZB Only)	DMA
0428h[3,4]	0000	0100	0010	10xx	r/w	CH2 S/G Descriptor Table Pointer (82378ZB Only)	DMA
042Ch[3,4]	0000	0100	0010	11xx	r/w	CH3 S/G Descriptor Table Pointer (82378ZB Only)	DMA
0434h[3,4]	0000	0100	0011	01xx	r/w	CH5 S/G Descriptor Table Pointer (82378ZB Only)	DMA
0438h[3,4]	0000	0100	0011	10xx	r/w	CH6 S/G Descriptor Table Pointer (82378ZB Only)	DMA
043Ch[3,4]	0000	0100	0011	11xx	r/w	CH7 S/G Descriptor Table Pointer (82378ZB Only)	DMA
0481h	0000	0100	1000	0001	r/w	DMA CH2 High Page Register	DMA
0482h	0000	0100	1000	0010	r/w	DMA CH3 High Page Register	DMA
0483h	0000	0100	1000	0011	r/w	DMA CH1 High Page Register	DMA
0487h	0000	0100	1000	0111	r/w	DMA CH0 High Page Register	DMA
0489h	0000	0100	1000	1001	r/w	DMA CH6 High Page Register	DMA
048Ah	0000	0100	1000	1010	r/w	DMA CH7 High Page Register	DMA
048Bh	0000	0100	1000	1011	r/w	DMA CH5 High Page Register	DMA

Address	Address				Type	Name	Block
	FEDC	BA98	7654	3210			
04D0	0000	0100	1101	0000	r/w	INT CNTRL-1 Edge Level Control Register	Control
04D1	0000	0100	1101	0001	r/w	INT CNTRL-2 Edge Level Control Register	Control
04D6h	0000	0100	1101	0010	wo	DMA2 Extended Mode Register (82378ZB Only)	DMA

NOTES:

1. PCI write cycles to these address locations flow through to the ISA Bus. PCI read cycles to these address locations do not flow through to the ISA Bus.
2. PCI read and write cycles to these address locations flow through to the ISA Bus.
3. The I/O address of this register is relocatable. The value shown in this table is the default address location.
4. This register can be accessed as a Byte, Word, or Dword quantity.
5. If this register location is enabled, PCI accesses to the BIOS Timer Register do not flow through to the ISA Bus. If disabled, accesses to this address location flow through to the ISA Bus.
6. When the DMAAC bit in the PCI Control Register is '0', the SIO/SIO.A aliases I/O accesses in the 80h-8Fh range to the 90h-9Fh range. Write accesses to these address locations flow through to the ISA Bus. Read cycles to these address locations do not flow through to the ISA Bus. When DMAAC=1, the SIO/SIO.A will only respond to the 80h-8Fh range and read and write accesses to these addresses in the 90h-9Fh range will be forwarded from the PCI Bus to the ISA Bus (I/O Port 92h is always a distinct register in the 90–9Fh range and is always fully decoded, regardless of the setting of the DMAAC bit).

3.1. SIO Configuration Register Description

This section describes the SIO/SIO.A Configuration Registers. These registers include the Mandatory Header Registers (located in the first 64 bytes of configuration space) and the SIO/SIO.A specific registers (located from configuration offset 40–56h).

3.1.1. VID—VENDOR IDENTIFICATION REGISTER

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

The VID Register contains the vendor identification number. This 16-bit register combined with the Device Identification Register uniquely identifies any PCI device. Writes to this register have no effect.

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

3.1.2. DID—DEVICE IDENTIFICATION REGISTER

Address Offset: 02–03h
Default Value: 0484h
Attribute: Read Only

The DID Register contains the device identification number. This register, along with the Vendor ID, uniquely identifies the SIO/SIO.A. Writes to this register have no effect.

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

3.1.3. COM—COMMAND REGISTER

Address Offset: 04–05h
Default Value: 0007h
Attribute: Read/Write

Bit	Description
15:5	**Reserved:** Read as 0.
4	**PMWE (Postable Memory Write Enable):** Enable postable memory write, memory write and invalidate, and memory read pre-fetch commands. The SIO/SIO.A does not support these commands as a master or slave so this bit is not implemented. This bit will always be read as a 0.
3	**SCE (Special Cycle Enable):** 1=Enable (the SIO/SIO.A will recognize PCI Special Cycles); 0=Disable (the SIO/SIO.A will ignore all PCI special cycles). This bit MUST be enabled if the STPCLK feature is being used.
2	**BME (Bus Master Enable):** Since the SIO/SIO.A always requests the PCI Bus on behalf of ISA masters, DMA, or line buffer PCI requests, this bit is hardwired to a 1 and will always be read as a 1.
1	**MSE (Memory Space Enable):** Enables SIO/SIO.A to accept a PCI-originated memory cycle. Since the SIO/SIO.A always responds to PCI-originated memory cycles (and ISA-bound cycles) by asserting DEVSEL#, this bit is hardwired to a 1 and will always be read as a 1.
0	**IOSE (I/O Space Enable):** Enable SIO/SIO.A to accept a PCI-originated I/O cycle. Since the SIO/SIO.A always responds to a master I/O cycle, this bit is hardwired to a 1 and will always be read as a 1.

3.1.4. DS—DEVICE STATUS REGISTER

Address Offset: 06–07h
Default Value: 0200h
Attribute: Read/Write

DSR is a 16-bit status register that reports the occurrence of a PCI master-abort by the SIO/SIO.A or a PCI target-abort when the SIO/SIO.A is a master. The register also indicates the SIO/SIO.A DEVSEL# signal timing that is hardwired in the SIO/SIO.A.

Bit	Description
15	**Reserved.** Read as 0.
14	**SERRS (SERR# Status)—Not Implemented:** This bit is set by the PCI devices that assert the SERR# signal. Since SERR# is only an input to the SIO/SIO.A, this bit is not implemented and will always be read as 0.
13	**MA (Master-Abort Status)—R/W:** When the SIO/SIO.A, as a master, generates a master-abort, MA is set to a 1. Software sets MA to 0 by writing a 1 to this bit location.
12	**RTA (Received Target-Abort Status)—R/W:** When the SIO/SIO.A is a master on the PCI Bus and receives a target-abort, this bit is set to a 1. Software sets RTA to 0 by writing a 1 to this bit location.
11	**STA (Signaled Target-Abort Status)—RO:** This bit is set to a 1 by the SIO/SIO.A when it generates a target-abort.
10:9	**DEVT (SIO DEVSEL# Timing Status)—RO:** This 2-bit field defines the timing for DEVSEL# assertion. These read only bits indicate the SIO/SIO.A DEVSEL# timing when performing a positive decode. Since the SIO/SIO.A always generates DEVSEL# with medium timing, DEVT=01. This DEVSEL# timing does not include Configuration cycles.
8:0	**Reserved:** Read as 0's.

3.1.5. RID—REVISION IDENTIFICATION REGISTER

Address Offset: 08h
Default Value: xxh (dependent on Part Revision)
Attribute: Read Only

This register contains the revision number for the SIO/SIO.A. This number indicates the stepping number of the component. Additionally, the upper nibble of the value is reserved. BIOS should mask the upper nibble when reading this register.

Bit	Description
7:4	**Reserved**
3:0	**Revision Identification Number:** This is a 4-bit value that indicates the revision identification number for the SIO/SIO.A. **82378ZB** 3h: 82378ZB A0-Stepping **82379AB** 88h: 82379AB A0 Stepping

3.1.6. PCICON—PCI CONTROL REGISTER

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

This 8-bit register controls the Line Buffer operation, the SIO/SIO.A PCI Posted Write Buffer enabling, and the DEVSEL# signal sampling point. The PCICON Register also controls how the SIO/SIO.A responds to INTA cycles on the PCI Bus and if the reserved DMA page registers are aliased from 80h-8Fh to 90h-9Fh.

Bit	Description
7	**Reserved:** Read as 0.
6	**DMAAC (DMA Reserved Page Register Aliasing Control):** This bit controls whether the SIO/SIO.A aliases I/O accesses in the 80h-8Fh to the 90h-9Fh range. When DMAAC=0, the SIO/SIO.A aliases I/O accesses in the 80h-8Fh to the 90h-9Fh range (AD4 is not used for decoding the DMA reserved page registers). When DMAAC=1, the SIO/SIO.A only respond to the 80h-8Fh range (AD4 is used for decoding the DMA reserved page registers). Read and write accesses to the 90h-9Fh range will be forwarded from the PCI Bus to the ISA Bus. **NOTE** I/O Port 92h is always a distinct register in the 90h-9Fh range and is always fully decoded, regardless of the setting of this bit.
5	**IAE (Interrupt Acknowledge Enable):** When IAE=0, the SIO/SIO.A ignores INTA cycles generated on the PCI Bus. However, when disabled, the SIO/SIO.A still responds to accesses to the 8259's register set and allows poll mode functions. When IAE=1, the SIO/SIO.A responds to INTA cycles in the normal fashion.
4:3	**SDSP (Subtractive Decoding Sample Point):** The SDSP field determines the DEVSEL# sample point, after which an inactive DEVSEL# results in the SIO/SIO.A forwarding the unclaimed PCI cycle to the ISA Bus (subtractive decoding). This setting should match the slowest device in the system. **Bit[4:3]** **Operation** **Bit[4:3]** **Operation** 00 Slow sample point 10 Fast sample point 01 Typical sample point 11 Reserved
2	**PPBE (PCI Posted Write Buffer Enable):** When PPBE=0 (default), the PCI posted write buffer is disabled. When PPBE=1, the PCI posted write buffer is enabled.
1	**ILBC (ISA Master Line Buffer Configuration):** When ILBC=0 (default), the Line Buffer is in single transaction mode. When ILBC=1, the Line Buffer is in 8-byte mode. This bit applies only to ISA Master transfers.
0	**DLBC (DMA Line Buffer Configuration):** When DLBC=0 (default), the Line Buffer is in single transaction mode. When DLBC=1, the Line Buffer is in 8-byte mode. This bit applies only to DMA transfers.

3.1.7. PAC—PCI ARBITER CONTROL REGISTER

Address Offset: 41h
Default Value: 00h
Attribute: Read/Write

This 8-bit register controls the operation of the PCI arbiter. The PAC Register enables/disables the guaranteed access time mode, controls bus lock cycles, enables/disables CPU bus parking, and controls the master retry timer.

Bit	Description
7:5	**Reserved:** Read as 0's.
4:3	**MRT (Master Retry Timer):** This 2-bit field determines the number of PCICLKs after the first retry that a PCI initiator's Bus request will be unmasked. **Bit[4:3]** — **Operation** 00 — Timer disabled, retries never masked. 01 — Retries unmasked after 16 PCICLK's. 10 — Retries unmasked after 32 PCICLK's. 11 — Retries unmasked after 64 PCICLK's.
2	**BP (Bus Park):** Set to a 1 the SIO/SIO.A will park CPUREQ# on the PCI Bus when it detects the PCI Bus idle. If Bus Park is disabled, the SIO/SIO.A takes responsibility for driving AD, C/BE# and PAR upon detection of bus idle state if the internal arbiter is enabled.
1	**BL (Bus Lock):** 1=Bus Lock (The arbiter considers the entire PCI Bus locked upon initiation of any locked transaction.); 0=resource lock (A locked agent is considered a locked resource and other agents may continue normal PCI transactions.)
0	**GAT (Guaranteed Access Time):** This bit enables/disables the guaranteed access time mode. When GAT=1, the SIO/SIO.A is configured for Guaranteed Access Time mode. This mode is available in order to guarantee the 2.5 µs CHRDY time-out specification for the ISA Bus. When the SIO/SIO.A is an Initiator on behalf of an ISA master, the PCI and memory busses are arbitrated for in serial and must be owned before the ISA master is given ownership of the ISA Bus. When GAT=0, the guaranteed access time mode is disabled. When guaranteed access time mode is disabled, the ISA master is first granted the ISA Bus and then the SIO/SIO.A arbitrates for the PCI Bus.

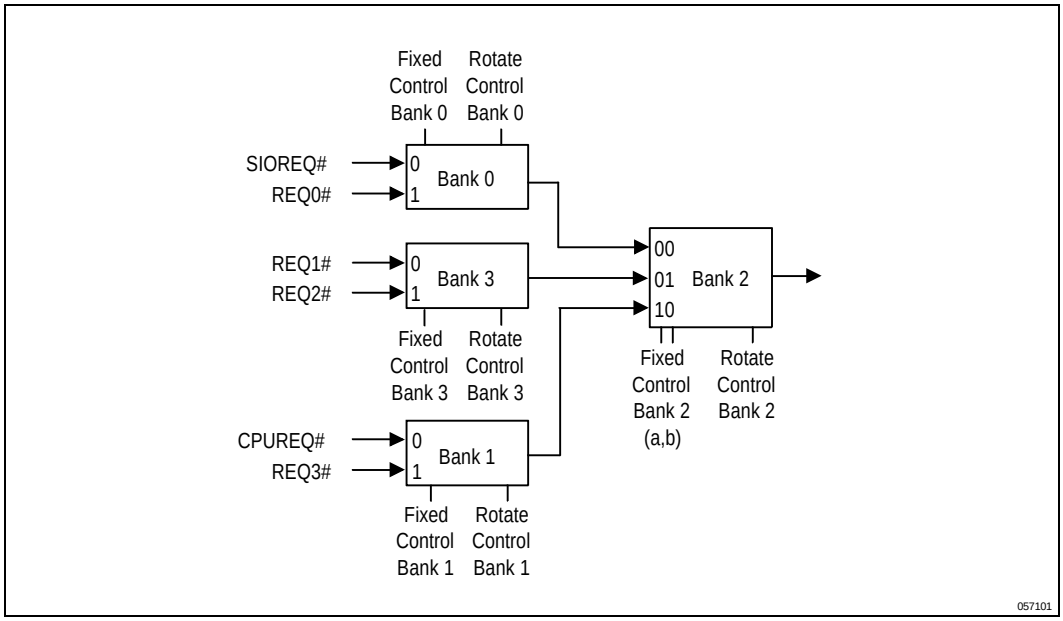
3.1.8. PAPC—PCI ARBITER PRIORITY CONTROL REGISTER

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

This register controls the PCI arbiter priority scheme. The arbiter supports six masters arranged through four switching banks. This permits the six masters to be arranged in a purely rotating priority scheme, one of 24 fixed priority schemes, or a hybrid combination of the fixed and rotating priority schemes. If both fixed and rotate modes are enabled for the same bank, the bank will be in rotate mode. For example, if both bits 0 and 4 are set to a 1, bank 0 will be in rotate mode. For each bit, 1=Enable and 0=Disable.

Bit	Description
7	**Bank 3 Rotate Control**
6	**Bank 2 Rotate Control**
5	**Bank 1 Rotate Control**
4	**Bank 0 Rotate Control**

Bit	Description
3	**Bank 2 Fixed Priority Mode Select B**
2	**Bank 2 Fixed Priority Mode Select A**
1	**Bank 1 Fixed Priority Mode Select**
0	**Bank 0 Fixed Priority Mode Select**



NOTE: SIOREQ#/SIOGNT# are SIO/SIO.A internal signals.

Figure 1. Arbiter Configuration Diagram

Fixed Priority Mode

The fixed bank control bits select which requester is the highest priority device within that particular bank.

Table 5. Fixed Mode Bank Control Bits

	Bank					Priority					
Mode	3	2b	2a	1	0	Highest					Lowest
00	0	0	0	0	0	SIOREQ#	REQ0#	REQ2#	REQ3#	CPUREQ#	REQ1#
01	0	0	0	0	1	REQ0#	SIOREQ#	REQ2#	REQ3#	CPUREQ#	REQ1#
02	0	0	0	1	0	SIOREQ#	REQ0#	REQ2#	REQ3#	REQ1#	CPUREQ#
03	0	0	0	1	1	REQ0#	SIOREQ#	REQ2#	REQ3#	REQ1#	CPUREQ#
04	0	0	1	0	0	CPUREQ#	REQ1#	SIOREQ#	REQ0#	REQ2#	REQ3#
05	0	0	1	0	1	CPUREQ#	REQ1#	REQ0#	SIOREQ#	REQ2#	REQ3#
06	0	0	1	1	0	REQ1#	CPUREQ#	SIOREQ#	REQ0#	REQ2#	REQ3#
07	0	0	1	1	1	REQ1#	CPUREQ#	REQ0#	SIOREQ#	REQ2#	REQ3#
08	0	1	0	0	0	REQ2#	REQ3#	CPUREQ#	REQ1#	SIOREQ#	REQ0#
09	0	1	0	0	1	REQ2#	REQ3#	CPUREQ#	REQ1#	REQ0#	SIOREQ#
0A	0	1	0	1	0	REQ2#	REQ3#	REQ1#	CPUREQ#	SIOREQ#	REQ0#
0B	0	1	0	1	1	REQ2#	REQ3#	REQ1#	CPUREQ#	REQ0#	SIOREQ#
0C-0F	0	1	1	x	x	Reserved					
10	1	0	0	0	0	SIOREQ#	REQ0#	REQ3#	REQ2#	CPUREQ#	REQ1#
11	1	0	0	0	1	REQ0#	SIOREQ#	REQ3#	REQ2#	CPUREQ#	REQ1#
12	1	0	0	1	0	SIOREQ#	REQ0#	REQ3#	REQ2#	REQ1#	CPUREQ#
13	1	0	0	1	1	REQ0#	SIOREQ#	REQ3#	REQ2#	REQ1#	CPUREQ#
14	1	0	1	0	0	CPUREQ#	REQ1#	SIOREQ#	REQ0#	REQ3#	REQ2#
15	1	0	1	0	1	CPUREQ#	REQ1#	REQ0#	SIOREQ#	REQ3#	REQ2#
16	1	0	1	1	0	REQ1#	CPUREQ#	SIOREQ#	REQ0#	REQ3#	REQ2#
17	1	0	1	1	1	REQ1#	CPUREQ#	REQ0#	SIOREQ#	REQ3#	REQ2#
18	1	1	0	0	0	REQ3#	REQ2#	CPUREQ#	REQ1#	SIOREQ#	REQ0#
19	1	1	0	0	1	REQ3#	REQ2#	CPUREQ#	REQ1#	REQ0#	SIOREQ#
1A	1	1	0	1	0	REQ3#	REQ2#	REQ1#	CPUREQ#	SIOREQ#	REQ0#
1B		1	0	1	1	REQ3#	REQ2#	REQ1#	CPUREQ#	REQ0#	SIOREQ#
1C-1F	1	1	1	x	x	Reserved					

Rotating Priority Mode

When any Bank Rotate Control bit is set to a 1, that particular bank rotates between the two requesting inputs. Any or all banks can be set in rotate mode. If, within a rotating bank, the highest priority input does not have an active request, then the lower priority input will be granted the bus. However, this does not change the roration scheme. When the bank toggles, the previously lowest priority input will become the highest priority input. Because of this, the maximum latency a device may encounter would be two complete rotations.

3.1.9. ARBPRIX—PCI ARBITER PRIORITY CONTROL EXTENSION REGISTER

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

This register provides the Fixed Priority Mode select for Bank 3 of the arbiter. The ARBPRIX Register fields are shown.

Bit	Description
7:1	**Reserved:** Read as 0.
0	**Bank 3 Fixed Priority Mode Select:** 0=REQ2# higher priority; 1=REQ3# higher priority.

3.1.10. MCSCON-MEMCS# CONTROL REGISTER

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

Bits[2:0] of this register enable MEMCS# blocks. PCI addresses within the enabled blocks result in the generation of MEMCS#. Note that the 0–512-Kbyte segment does not have RE and WE bits. The 0–512-Kbyte segment can only be turned off with the MEMCS# Master Enable bit (bit 4). Note also, that when the RE and WE bits are both 0 for a particular segment, the PCI master can not access the segment.

Bit	Description
7:5	**Reserved:** Read as 0's.
4	**MEMCS# Master Enable—R/W:** When the MEMCS# master enable bit is set to a 1, the SIO/SIO.A asserts MEMCS# for all accesses to the defined MEMCS# region (that have been programmed in this register and the MAR1, MAR2, and MAR3 Registers). Also, when this bit is a 1, the positive decoding functions enabled by having the ISA Clock Divisor Register bit 6=1 and the Utility Bus Chip Select Register "A" bit 6=1 are ignored. Subtractive decoding is provided for these memory areas, instead. When the MEMCS# master enable bit is set to a 0, the entire MEMCS# function is disabled. When this bit is 0, MEMCS# will never be asserted.
3	**Write Enable For 0F0000h–0FFFFFh (Upper 64 Kbyte BIOS):** 1=Enable; 0=Disable.
2	**Read Enable For 0F0000h–0FFFFFh (Upper 64 Kbyte BIOS):** 1=Enable; 0=Disable.
1	**Write Enable For 080000h–09FFFFh (512–640 Kbyte):** 1=Enable; 0=Disable.
0	**Read Enable For 080000h–09FFFFh (512–640 Kbyte):** 1=Enable; 0=Disable.

3.1.11. MCSBOH—MEMCS# BOTTOM OF HOLE REGISTER

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

This register defines the bottom of the MEMCS# hole. MEMCS# is not generated for accesses to addresses within the hole defined by this register and the MCSTOH Register. The hole is defined by the following equation: TOH ≥ address ≥ BOH. TOH is the top of the MEMCS# hole defined by the MCSTOH Register and BOH is the bottom of the MEMCS# hole defined by this register. For example, to program the BOH at 1 Mbyte, the value of 10h should be written to this register. To program the BOH at 2 Mbytes + 64 Kbytes this register should be programmed to 21h. To program the BOH at 8 Mbytes, this register should be programmed to 80h.

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

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

3.1.12. MCSTOH—MEMCS# TOP OF HOLE REGISTER

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

This register defines the top of the MEMCS# hole. MEMCS# is not generated for accesses to addresses within the hole defined by this register and the MCSBOH Register. The hole is defined by the following equation: TOH ≥ address ≥ BOH. TOH is the top of the MEMCS# hole defined by this register and BOH is the bottom of the MEMCS# hole defined by the MCSBOH Register. For example, to program the TOH at 1 Mbyte + 64 Kbytes, this register should be programmed to 10h. To program the TOH at 2 Mbytes + 128 Kbytes, this register should be programmed to 21h. To program the TOH at 12 Mbytes, this register should be programmed to BFh.

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

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

3.1.13. MCSTOM—MEMCS# TOP OF MEMORY REGISTER

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

This register determines MEMCS# top of memory boundary. The top of memory boundary ranges up to 512 Mbytes, in 2-Mbyte increments. This register is typically set to the top of main memory. Accesses ≥ 2 Mbytes and ≤ top of memory boundary results in the assertion of the MEMCS# signal (unless the address resides in the hole programmed by the MCSBOH and MCSTOH Registers). A value of 00h disables this 2 Mbyte-to-top memory region. A value of 00h assigns the top of memory to include 2 Mbyte - 1. A value of FFh assigns the top of memory to include 512 Mbytes - 1.

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

3.1.14. IADCON—ISA ADDRESS DECODER CONTROL REGISTER

Address Offset: 48h
Default value: 01h
Attribute: Read/Write

This register enables the forwarding of ISA or DMA memory cycles to the PCI Bus. In addition, this register sets the top of the "1 Mbyte to top of main memory" region.

Bit	Description
7:4	**ISA Memory Cycle Forwarding To PCI:** The top can be assigned in 1 Mbyte increments from 1 Mbyte up to 16 Mbytes. ISA master or DMA accesses within this region are forwarded to PCI unless they are within the hole. **Bits[7:4] / Top of Memory / Bits[7:4] / Top of Memory** 0000 / 1 Mbyte / 1000 / 9 Mbytes 0001 / 2 Mbytes / 1001 / 10 Mbytes 0010 / 3 Mbytes / 1010 / 11 Mbytes 0011 / 4 Mbytes / 1011 / 12 Mbytes 0100 / 5 Mbytes / 1100 / 13 Mbytes 0101 / 6 Mbytes / 1101 / 14 Mbytes 0110 / 7 Mbytes / 1110 / 15 Mbytes 0111 / 8 Mbytes / 1111 / 16 Mbytes
3:0	**ISA and DMA Memory Cycle To PCI Bus Enables:** The memory block is enabled by writing a 1 to the corresponding bit position. Setting the bit to 0 disables the corresponding block. ISA or DMA memory cycles to the enabled blocks result in the ISA cycle being forwarded to the PCI Bus. The BIOSCS# enable bit (bit 6 in the UBCSA Register) for the 896K-960K region overrides the function of bit 3 of this register. If the BIOSCS# bit is set to a 1, the ISA or DMA memory cycle is always contained to ISA, regardless of the setting of bit 3 in this register. If the BIOSCS# bit is disabled, the cycle is forwarded to the PCI Bus if bit 3 in this register is enabled. **Bit / Memory Block / Bit / Memory Block** 0 / 0–512-Kbyte Memory / 2 / 640–768-Kbyte VGA Memory 1 / 512–640-Kbyte Memory / 3 / 896–960-Kbyte Low BIOS

3.1.15. IADRBE—ISA ADDRESS DECODER ROM BLOCK ENABLE REGISTER

Address Offset: 49h
Default value: 00h
Attribute: Read/Write

ISA addresses within the enabled ranges result in the ISA memory cycle being forwarded to the PCI Bus. If the memory block is disabled, the ISA cycle is not forwarded to the PCI Bus. For each bit, 1=Enable and 0=Disable.

Bit	Description
7	880-896K Memory Enable
6	864-880K Memory Enable
5	848-864K Memory Enable
4	832-848K Memory Enable

Bit	Description
3	816-832K Memory Enable
2	800-816K Memory Enable
1	784-800K Memory Enable
0	768-784K Memory Enable

3.1.16. IADBOH—ISA ADDRESS DECODER BOTTOM OF HOLE REGISTER

Address Offset: 4Ah
Default value: 10h
Attribute: Read/Write

This register defines the bottom of the ISA Address Decoder hole. The hole is defined by the following equation: TOH ≥ address ≥ BOH, where BOH is the bottom of the hole address programmed into this register and TOH is the top of the hole address programmed into the IADTOH Register. ISA master or DMA addresses falling within the hole will not be forwarded to the PCI Bus. The hole can be sized in 64-Kbyte increments and placed anywhere between 1 Mbyte and 16 Mbytes on any 64-Kbyte boundary. When TOH < BOH, the hole is effectively disabled. The default value for the BOH and TOH disables the hole.

For example, to program the BOH at 1 Mbyte, this register should be set to 10h. To program the BOH at 2 Mbytes, this register should be set to 20h. To program the BOH at 8 Mbytes, this register should be set to 80h.

Bit	Description
7:0	**ISA Bottom of Memory Hole Address:** Bits[7:0] correspond to address lines A[23:16], respectively.

3.1.17. IADTOH—ISA ADDRESS DECODER TOP OF HOLE REGISTER

Address Offset: 4Bh
Default value: 0Fh
Attribute: Read/Write

This register defines the top of the ISA Address Decoder hole. The hole is defined by the following equation: TOH ≥ address ≥ BOH, where BOH is the bottom of the hole address programmed into the LADBOH Register and TOH is the top of the hole address programmed into this Register. ISA master or DMA addresses within the hole will not be forwarded to the PCI Bus. The hole can be sized in 64-Kbyte increments and placed anywhere between 1 Mbyte and 16 Mbytes on any 64-Kbyte boundary. When TOH < BOH, the hole is disabled. The default value for the BOH and TOH disables the hole.

For example, to program the TOH at 1 Mbyte + 64 Kbytes, this register should be set to 10h. To program the TOH at 2 Mbytes + 128 Kbytes, this register should be set to 21h. To program the TOH at 12 Mbytes, this register should be set to BFh.

Bit	Description
7:0	**ISA Top of Memory Hole Address:** Bits[7:0] correspond to address lines A[23:16], respectively.

3.1.18. ICRT—ISA CONTROLLER RECOVERY TIMER REGISTER

Address Offset: 4Ch
Default Value: 56h
Attribute: Read/Write

The I/O recovery mechanism in the SIO/SIO.A is used to add additional recovery delay between PCI originated 8-bit and 16-bit I/O cycles to the ISA Bus. The SIO/SIO.A automatically forces a minimum delay of five SYSCLKs between back-to-back 8- and 16-bit I/O cycles to the ISA Bus. The delay is measured from the rising edge of the I/O command (IOR# or IOW#) to the falling edge of the next BALE. If a delay of greater than five SYSCLKs is required, the ISA I/O Recovery Time Register can be programmed to increase the delay in increments of SYSCLKs. Note that no additional delay is inserted for back-to-back I/O "sub cycles" generated as a result of byte assembly or disassembly. This register defaults to 8- and 16-bit recovery enabled with two clocks added to the standard I/O recovery.

Bit	Description
7	**Reserved:** Read as 0.
6	**8-Bit I/O Recovery Enable:** 1=Enable delay programmed via bits[5:3]; 0=Disable Delay.
5:3	**8-Bit I/O Recovery Times:** This 3-bit field defines the recovery times for 8-bit I/O. Programmable delays between back-to-back 8-bit PCI cycles to ISA I/O slaves is shown in terms of ISA clock cycles (SYSCLK) added to the five minimum. The selected delay programmed into this field is enabled/disabled via bit 6 of this register. **Bit[5:3] / SYSCLK Added / Total SYSCLKs / Bit[5:3] / SYSCLK Added / Total SYSCLKs** 001 / +1 / 6 / 101 / +5 / 10 010 / +2 / 7 / 110 / +6 / 11 011 / +3 / 8 / 111 / +7 / 12 100 / +4 / 9 / 000 / +8 / 13
2	**16-Bit I/O Recovery Enable:** 1=Enable delay programmed via bits[1:0]; 0=Disable Delay.
1:0	**16-Bit I/O Recovery Times:** This 2-bit field defines the recovery time for 16-bit I/O. Programmable delays between back-to-back 16-bit PCI cycles to ISA I/O slaves is shown in terms of ISA clock cycles (SYSCLK) added to the five minimum. The selected delay programmed into this field is enabled/disabled via bit 2 of this register. **Bit[5:3] / SYSCLK Added / Total SYSCLKs / Bit[5:3] / SYSCLK Added / Total SYSCLKs** 01 / +1 / 6 / 11 / +3 / 8 10 / +2 / 7 / 00 / +4 / 9

3.1.19. ICD—ISA CLOCK DIVISOR REGISTER

Address Offset: 4Dh
Default Value: 40h
Attribute: Read/Write

This register selects the integer value used to divide the PCI clock (PCICLK) to generate the ISA clock (SYSCLK). In addition, this register provides an ISA Reset bit to software control RSTDRV, a bit to enable/disable the MOUSE function, a bit to enable/disable the coprocessor error support, and a bit to disable the positive decode for the upper 64 Kbytes of BIOS at the top of 1 Mbyte (F0000–FFFFFh) and aliased regions.

Bit	Description
7	**Reserved:** Read as 0.
6	**Positive Decode of Upper 64 Kbyte BIOS Enable:** This bit enables (bit 6=1) and disables (bit 6=0) the positive decode of the upper 64 Kbytes of BIOS area at the top of 1 Mbyte (F0000–FFFFFh) and the aliased regions at the top of 4 Gbytes (FFFF0000–FFFFFFFFh) and 4 Gbytes-1 Mbyte (FFEF0000–FFEFFFFFh). When bit 6=1, these address regions are positively decoded, unless bit 4 in the MEMCS# Control Register is set to a 1 in which case these regions are subtractively decoded. When bit 6=0, these address regions are subtractively decoded. The encoded chip selects for BIOSCS# and the UBUSOE# signal will always be generated when these locations are accessed, regardless of the state of this bit.
5	**Coprocessor Error Enable:** This bit is used to enable and disable the Coprocessor error support. When enabled (bit 5=1), the FERR# input, when driven active, triggers an IRQ13 to the SIO/SIO.A interrupt controller. FERR# is also used to gate the IGNNE# output. When disabled (bit 5=0), the FERR# signal can be used as IRQ13 and the coprocessor support is disabled.
4	**IRQ12/M Mouse Function Enable:** When this bit is set to 1, IRQ12/M provides the mouse function. When this bit is set to 0, IRQ12/M provides the standard IRQ12 interrupt function.
3	**RSTDRV Enable:** This bit is used to enable RSTDRV on the ISA Bus. When this bit is set to 1, RSTDRV is asserted and remains asserted until this bit is set to a 0. When set to 0, normal operation of RSTDRV is provided. This bit should be used during configuration to reset the ISA Bus when changing the clock divisor. Note that the software must ensure that RSTDRV is asserted for a minimum of 1 µs.
2:0	**PCICLK-to-ISA SYSCLK Divisor:** These bits are used to select the integer that is used to divide the PCICLK down to generate the ISA SYSCLK. Upon reset, these bits are set to 000 (divisor of 4 selected). For PCI frequencies less than 33 MHz (not including 25 MHz), a clock divisor value must be selected that ensures that the ISA Bus frequency does not violate the 6 MHz to 8.33 MHz SYSCLK specification.

Bit[2:0]	Divisor	SYSCLK	Bit[2:0]	Divisor	SYSCLK
000	4 (33 MHz)	8.33 MHz	100	Reserved	
001	3 (25 MHz)	8.33 MHz	101	Reserved	
010	Reserved		110	Reserved	
011	Reserved		111	Reserved	

3.1.20. UBCSA—UTILITY BUS CHIP SELECT A REGISTER

Address Offset: 4Eh
Default Value: 07h
Attribute: Read/Write

This register enables/disables accesses to the RTC, keyboard controller, floppy disk controller, IDE, and BIOS locations E0000–EFFFFh and FFF80000–FFFDFFFFh. Disabling any of these bits prevents the encoded chip select bits (ECSADDR[2:0]) and Utility Bus transceiver control signal (UBUSOE#) for that device from being generated.

This register is also used to select which address range (primary or secondary) will be decoded for the resident floppy controller and IDE. This ensures that there is no contention with the Utility Bus transceiver driving the system data bus during read accesses to these devices.

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>7</td><td>Extended BIOS Enable: When bit 7=1 (enabled), PCI accesses to locations FFF80000–FFFDFFFFh result in the generation of the encoded signals (ECSADDR[2:0]) for BIOS. When enabled, PCI master accesses to this area are positively decoded and UBUSOE# is generated. When this bit is disabled (bit 7=0), the SIO/SIO.A does not generate the encoded (ECSADDR[2:0]) signals or UBUSOE#.</td></tr>
<tr><td>6</td><td>Lower BIOS Enable: When bit 6=1 (enabled), PCI or ISA accesses to the lower 64 Kbyte BIOS block (E0000–EFFFFh) at the top of 1 Mbyte, or the aliases at the top of 4 Gbyte and 4 Gbyte - 1 Mbyte results in the generation of the encoded (ECSADDR[2:0]) signals for BIOS. When enabled, PCI master accesses to this area are positively decoded to the ISA Bus, unless bit 4 in the MEMCS# Control Register is set to a 1 in which case these regions are subtractively decoded. Also, when enabled, ISA master or DMA master accesses to this region are not forwarded to the PCI Bus. When this bit is disabled (bit 6=0), the SIO/SIO.A does not generate the encoded (ECSADDR[2:0]) signals. Also, when this bit is disabled, ISA master or DMA accesses to this region are forwarded to PCI, if bit 3 in the IADCON Register is set to 1.</td></tr>
<tr><td>4</td><td>IDE Decode Enable: Bit 4 enables/disables IDE locations 1F0–1F7h (primary) or 170–177h (secondary) and 3F6h, 3F7h (primary) or 376h, 377h (secondary). When bit 4=1, the IDE encoded chip select signals and the Utility Bus transceiver signal (UBUSOE#) are generated for these addresses. When bit 4=0, these signals are not generated for these addresses.</td></tr>
<tr><td>5,3:2</td><td>Floppy Disk Address Locations Enable: Bits 2 and 3 are used to enable or disable the floppy locations as indicated below. A PCIRST# sets bit 2 to 1 and bit 3 to 0. Bit 5 is used to select between the primary and secondary address range used by the floppy controller and the IDE. Only primary or only secondary can be programmed at any one time. A PCIRST# sets this bit to 0 (primary). The following table shows how these bits are used to select the floppy controller:
<table>
<tr><th>Address</th><th>Bit 5</th><th>Bit 3</th><th>Bit 2</th><th>DSKCHG</th><th>ECSADDR[2:0]</th><th>FLOPPYCS#</th></tr>
<tr><td>X</td><td>X</td><td>X</td><td>X</td><td>0</td><td>1 1 1</td><td>1</td></tr>
<tr><td>3F0h, 3F1h</td><td>0</td><td>1</td><td>X</td><td>1</td><td>1 0 0</td><td>0</td></tr>
<tr><td>3F2–3F7h</td><td>0</td><td>X</td><td>1</td><td>1</td><td>1 0 0</td><td>0 (note)</td></tr>
<tr><td>370h, 371h</td><td>1</td><td>1</td><td>X</td><td>1</td><td>1 0 0</td><td>0</td></tr>
<tr><td>372–37Fh</td><td>1</td><td>X</td><td>1</td><td>1</td><td>1 0 0</td><td>0 (note)</td></tr>
</table>
NOTE
If IDE decode is enabled (bit 4=1), all accesses to locations 03F6h and 03F7h (primary) or 0376h and 0377h (secondary) result in the ECSADDR[2:0] signals generating a decode for IDECS1# (FLOPPYCS# is not generated). An external AND gate can be used to tie IDECS1# and FLOPPYCS# together to insure that the floppy is enabled for these accesses. If IDE decode is disabled (bit 4=0), and the decode for the floppy is enabled, then the encoded chip selects for the floppy locations are generated.</td></tr>
</table>

Bit	Description
1	**Keyboard Controller Address Location Enable:** 1=Enable. 0=Disable. When disabled, the keyboard controller encoded chip select signals (ECSADDR[2:0]) and the Utility Bus transceiver signal (UBUSOE#) are not generated for the address locations below. For the 82378ZB, the enabled address locations are 60h, 62h, 64h, and 66h. For the 82379AB, the enabled address locations are 60h, and 64h.
0	**RTC Address Location Enable:** Enables (1) or disables (0) the RTC address locations 70–77h. When this bit is set to 0, the RTC encoded chip select signals (ECSADDR[2:0]), RTCALE#, RTCCS#, and UBUSOE# signals are not generated for these addresses.

3.1.21. UBCSB—UTILITY BUS CHIP SELECT B REGISTER

Address Offset: 4Fh
Default Value: 4Fh
Attribute: Read/Write

This register is used to enable/disable accesses to the serial ports and parallel port locations supported by the SIO/SIO.A. When disabled, the ECSADDR(2:0) encoded chip select bits and Utility Bus Transceiver control signal (UBUSOE#), for that device, are not generated. This register is also used to disable accesses to Port 92 and enable or disable configuration RAM decode.

Bit	Description
7	**Configuration RAM Decode Enable:** This bit is used to enable (bit 7=1) or disable (bit 7=0) I/O write accesses to location 0C00h and I/O read/write accesses to locations 0800–08FFh. When enabled, the encoded chip select signals for generating an external configuration page chip select (CPAGECS#) are generated for accesses to 0C00h. The encoded chip select signals for generating an external configuration memory chip select (CFIGMEMCS#) are generated for accesses to 0800–08FFh. When bit 7=0, configuration RAM decode is disabled and the CPAGECS# and CFIGMEMCS# are not generated for the corresponding accesses.
6	**Port 92 Enable:** 1=Enable; 0=Disable.
5:4	**Parallel Port Enable:** These bits are used to select the parallel port address range: (LPT1, LPT2, LPT3, or disable). When a PCIRST# occurs, this field is set to 00 (LPT1). **Bit[5:4] Function** — 00: 3BC–3BFh (LPT1); 01: 378–37Fh (LPT2); **Bit[5:4] Function** — 10: 278–27Fh (LPT3); 11: Disabled
3:2	**Serial Port B Enable:** These bits are used to assign Serial Port B address range: (COM1, COM2, or disable). If either COM1 or COM2 address ranges are selected, the encoded chip select signals [ECSADDR(2:0)] for Port B will be generated. A PCIRST# sets bit[3:2] to 11 (Port B disabled). Note that, If Serial Port A and B are programmed for the same I/O address, the encoded chip select signals, ECSADDR(2:0), for Port B are disabled. **Bit[3:2] Function** — 00: 3F8–3FFh (COM1); 01: 2F8–2FFh (COM2); **Bit[3:2] Function** — 10: Reserved; 11: Port B Disabled

Bit	Description
1:0	**Serial Port A Enable:** These bits are used to assign serial port A address range (COM1, COM2, or disable). If either COM1 or COM2 address ranges are selected, the encoded chip select signals (ECSADDR[2:0]) for Port A will be generated. A PCIRST# sets bits[1:0] to 11 (Port A disabled). Note that, If Serial Port A and B are programmed for the same I/O address, the encoded chip select signals, ECSADDR[2:0], for Port B are disabled. **Bit[1:0] Function** — 00: 3F8–3FFh (COM1); 01: 2F8–2FFh (COM2); 10: Reserved; 11: Port A disabled

3.1.22. MAR1—MEMCS# ATTRIBUTE REGISTER #1

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

RE—Read Enable. When RE=1 (bit 6, 4, 2, 0), the SIO/SIO.A generates MEMCS# for PCI master, DMA, or ISA master memory read accesses to the corresponding segment. When RE=0, the SIO/SIO.A does not generate MEMCS# for PCI master memory read accesses to the corresponding segment. When RE=WE=0 (or bit 4=0 in the MEMCS# Control Register - disabled), the PCI master, DMA, or ISA master can not access the segment.

WE—Write Enable. When WE=1 (bit 7, 5, 3, 1), the SIO/SIO.A generates MEMCS# for PCI master, DMA, or ISA master memory write accesses to the corresponding segment. When WE=0, the SIO/SIO.A does not generate MEMCS# for PCI master memory write accesses to the corresponding segment. When the RE=WE bits=0 (or bit 4=0 in the MEMCS# Control Register - disabled), the PCI master, DMA, or ISA master can not access the segment.

Bit	Description
7	0CC000–0CFFFFh Exp. ROM: WE
6	0CC000–0CFFFFh Exp. ROM: RE
5	0C8000–0CBFFFh Exp. ROM: WE
4	0C8000–0CBFFFh Exp. ROM: RE

Bit	Description
3	0C4000–0C7FFFh Exp. ROM: WE
2	0C4000–0C7FFFh Exp. ROM: RE
1	0C0000–0C3FFFh Exp. ROM: WE
0	0C0000–0C3FFFh Exp. ROM: RE

3.1.23. MAR2—MEMCS# ATTRIBUTE REGISTER #2

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

RE—Read Enable. When RE=1 (bit 6, 4, 2, 0), the SIO/SIO.A generates MEMCS# for PCI master, DMA, or ISA master memory read accesses to the corresponding segment. When RE=0, the SIO/SIO.A does not generate MEMCS# for PCI master memory read accesses to the corresponding segment. When RE= WE=0 (or bit 4=0 in the MEMCS# Control Register - disabled), the PCI master, DMA, or ISA master can not access the segment.

WE—Write Enable. When WE=1 (bit 7, 5, 3, 1), the SIO/SIO.A generates MEMCS# for PCI master, DMA, or ISA master memory write accesses to the corresponding segment. When WE=0, the SIO/SIO.A does not generate MEMCS# for PCI master memory write accesses to the corresponding segment. When RE=WE=0 (or bit 4=0 in the MEMCS# Control Register - disabled), the PCI master, DMA, or ISA master can't access the segment.

Bit	Description
7	0DC000–0DFFFFh Exp. ROM : WE
6	0DC000–0DFFFFh Exp. ROM : RE
5	0D8000–0DBFFFh Exp. ROM : WE
4	0D8000–0DBFFFh Exp. ROM : RE

Bit	Description
3	0D4000–0D7FFFh Exp. ROM : WE
2	0D4000–0D7FFFh Exp. ROM : RE
1	0D0000–0D3FFFh Exp. ROM : WE
0	0D0000–0D3FFFh Exp. ROM : RE

3.1.24. MAR3—MEMCS# ATTRIBUTE REGISTER #3

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

RE—Read Enable. When RE=1 (bit 6, 4, 2, 0), the SIO/SIO.A generates MEMCS# for PCI master, DMA, ISA master memory read accesses to the corresponding segment. When RE=0, the SIO/SIO.A does not generate MEMCS# for PCI master memory read accesses to the corresponding segment. When the RE=WE=0 (or bit 4=0 in the MEMCS# Control Register - disabled), the PCI master can not access the segment.

WE—Write Enable. When WE=1 (bit 7, 5, 3, 1), the SIO/SIO.A generates MEMCS# for PCI master, DMA, ISA master memory write accesses to the corresponding segment. When WE=0, the SIO/SIO.A does not generate MEMCS# for PCI master memory write accesses to the corresponding segment. When RE=WE=0 (or bit 4=0 in the MEMCS# Control Register - disabled), the PCI master can not access the segment.

Bit	Description
7	0EC000–0EFFFFh Lower 64-Kbyte BIOS: WE
6	0EC000–0EFFFFh Lower 64-Kbyte BIOS: RE
5	0E8000–0EBFFFh Lower 64-Kbyte BIOS: WE
4	0E8000–0EBFFFh Lower 64-Kbyte BIOS: RE

Bit	Description
3	0E4000–0E7FFFh Lower 64-Kbyte BIOS: WE
2	0E4000–0E7FFFh Lower 64-Kbyte BIOS: RE
1	0E0000–0E3FFFh Lower 64-Kbyte BIOS: WE
0	0E0000–0E3FFFh Lower 64-Kbyte BIOS: RE

3.1.25. PIRQ[3:0]#—PIRQ ROUTE CONTROL REGISTERS

Address Offset: 60h, 61h, 62h, 63h
Default Value: 80h
Attribute: Read/Write

These four 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. Note that two or more PCI interrupts (PIRQ[3:0]#) can be steered into the same IRQ signal (the interrupts are level sensitive and can be shared). In addition, each IRQ to which a PCI Interrupt is steered into must have its interrupt set to level sensitive in the Edge\Level Control Register.

Bit	Description
7	**Routing of Interrupts:** When enabled, this bit routes the PCI Interrupt signal to the PC compatible interrupt signal specified in bits[3:0]. At reset, this bit is disabled (set to 1).

Bit	Description
6:4	**Reserved:** Read as 0's.
3:0	**IRQx# Routing Bits:** These bits specify which IRQ signal to generate.

Bits[3:0]	IRQx#	Bits[3:0]	IRQx#	Bits[3:0]	IRQx#
0000	Reserved	0110	IRQ6	1100	IRQ12
0001	Reserved	0111	IRQ7	1101	Reserved
0010	Reserved	1000	Reserved	1110	IRQ14
0011	IRQ3	1001	IRQ9	1111	IRQ15
0100	IRQ4	1010	IRQ10		
0101	IRQ5	1011	IRQ11		

3.1.26. PACC—PIC/APIC CONFIGURATION CONTROL REGISTER (82379AB Only)

Address Offset: 70h
Default Value: 00h
Attribute: Write Only

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

Bit	Description
7:2	Reserved
1	SMI Routing Control (SMIRC): 1=SMI routed via the APIC; 0=SMI routed via the SMI# signal. When SMRC=1, INT can not be routed through the APIC, since it is sharing the APIC interrupt input with SMI.
0	INT Routing Control (INTRC): When APIC is enabled (in mixed or pure APIC mode), this bit allows the 82379AB's external INT signal to be masked (forces INT to the inactive state but does not tri-states the signal). Thus, the CPU's INT pin can be used (by providing a simple -gate) for the APIC Local Interrupt (LINTRx). However, INT must not be masked via this bit when APIC is disabled and INT is the only mechanism to signal the 8259 recognized interrupts to the CPU. When INTRC=1, INT is disabled (APIC must be enabled). When INTRC=0, INT is enabled.

3.1.27. APICBASE—APIC BASE ADDRESS RELOCATION (82379AB Only)

Address Offset: 71h
Default Value: 00h
Attribute: Write Only

The APICBASE Register provides the modifier for the APIC base address. 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:** Bits[5:2] are compared to host address bits A[15:12], respectively.
1:0	**Y-Base Address:** Bits[1:0] are compared to host address bits A[11:10], respectively.

3.1.28. BIOS TIMER BASE ADDRESS REGISTER

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

This register determines the base address for the BIOS Timer Register located in the I/O space. The base address can be set at Dword boundary anywhere in the 64 Kbyte I/O space. This register also provides the BIOS Timer access enable/disable control bit.

Bit	Description
15:2	**BIOS Timer Base Address:** Bits[15:2] correspond to PCI address lines A[15:2], respectively.
1	**Reserved.** Read as 0.
0	**BIOS Timer Access Enable:** When bit 0=1, access to the BIOS Timer is enabled. When bit 0=0, access to the BIOS Timer is disabled. The default value is 0 (disabled).

3.1.29. SMICNTL—SMI CONTROL REGISTER

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

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	**82378ZB: Reserved.** **82379AB: CWSGNTDI—R/W:** When this bit is set to a 0, the 82379AB requires a Stop Grant bus cycle before negating the STPCLK# signal. When this bit is set to a 1, the 82379AB does not wait for Stop Grant before negating STPCLK#.
5:4	**Reserved**
3	**Fast-Off Timer Freeze (CTMRFRZ)—R/W:** 1=Fast-Off timer stops counting (prevents time-outs from occurring while executing SMM code); 0=Fast-Off timer counts.
2	**STPCLK# Scaling Enable (CSTPCLKSC)—R/W:** 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)—R/W:** 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.

Bit	Description
0	**SMI# Gate (CSMIGATE)—R/W:** 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 affect the detection/recording of SMI events (i.e., This bit does not affect 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.

3.1.30. SMIEN—SMI ENABLE REGISTER

Address Offset: A2–A3h
Default Value: 0000h
Attribute: Read/Write

This 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:** 1=Enable; 0=Disable
6	**EXTSMI# SMI Enable:** 1=Enable; 0=Disable
5	**Fast-Off Timer SMI Enable:** 1=Enable; 0=Disable
4	**IRQ12 SMI Enable (PS/2 Mouse Interrupt):** 1=Enable; 0=Disable
3	**IRQ8 SMI Enable (RTC Alarm Interrupt):** 1=Enable; 0=Disable
2	**IRQ4 SMI Enable (COM2/COM4 Interrupt or Mouse):** 1=Enable; 0=Disable
1	**IRQ3 SMI Enable (COM1/COM3 Interrupt or Mouse):** 1=Enable; 0=Disable
0	**IRQ1 SMI Enable (Keyboard Interrupt):** 1=Enable; 0=Disable

3.1.31. SEE—SYSTEM EVENT ENABLE REGISTER

Address Offsset: A4–A7h
Default Value: 00000000h
Attribute: Read/Write

This register enables hardware events as system events or break events for power management control. For I/O locations that can cause a system event via bits[27:24], refer to the Power Management section.

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 or an access within the defined range), 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 SIO/SIO.A.

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):** 1=Enable; 0=Disable. System and Break events.
30	**82378ZB: Reserved.** **82379AB: Fast-Off Interrupt Enable (FINTREN).** 1=Enable; 0=Disable. Break event only.
29	**Fast-Off NMI Enable (FNMIEN):** 1=Enable; 0=Disable. System and Break events.
28	**Reserved.**
27	**82378ZB: Reserved.** **82379AB: Fast-Off COM Enable (FCOMEN):** 1=Enable; 0=Disable. System events only.
26	**82378ZB: Reserved.** **82379AB: Fast-Off LPF Enable (FLPTEN):** 1=Enable; 0=Disable. System events only.
25	**82378ZB: Reserved.** **82379AB: Fast-Off Drive Enable (FDRVEN):** 1=Enable; 0=Disable. System events only.
24	**82378ZB: Reserved.** **82379AB: Fast-Off DMA Enable (FDMAEN):** 1=Enable; 0=Disable. System events only.
23:16	**Reserved.**
15:3	**Fast-Off IRQ[15:3] Enable (FIRQ[15:3]EN):** 1=Enable; 0=Disable. System and Break events.
2	**Reserved.**
1:0	**Fast-Off IRQ[1:0] Enable (FIRQ[1:0]EN):** 1=Enable; 0=Disable. System and Break events.

3.1.32. FTMR—FAST OFF TIMER REGISTER

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

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.

3.1.33. SMIREQ—SMI REQUEST REGISTER

Address Offset: AA–ABh
Default Value: 00h
Attribute: Read/Write

The SMIREQ Register contains status bits indicating the cause of an SMI. When an enabled event causes an SMI, the SIO/SIO.A automatically sets the corresponding event's status bit to 1. Note that, if software attempts to set a status bit to 0 at the same time that the SIO/SIO.A is setting it to 1, the bit is set to 1. If the SMI event is still active when the corresponding SMIREQ bit is set to 0, the SIO/SIO.A 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 SIO/SIO.A 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 SIO/SIO.A 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 SIO/SIO.A 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 SIO/SIO.A 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 SIO/SIO.A 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 SIO/SIO.A 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 SIO/SIO.A 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.

Bit	Description
2	**IRQ4 Request SMI Status:** The SIO/SIO.A 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 SIO/SIO.A 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 SIO/SIO.A 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.

3.1.34. CTLTMR—CLOCK SCALE STPCLK# LOW TIMER

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

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.

3.1.35. CTLTMRH—CLOCK SCALE STPCLK# HIGH TIMER

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

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 SIO/SIO.A contains DMA circuitry that incorporates the functionality of two 82C37 DMA controllers (DMA1 and DMA2). The DMA registers control the operation of the DMA controllers and are all accessible from the PCI Bus via PCI I/O space. In addition, some of the registers are accessed from the ISA Bus via ISA I/O space. Table 4, at the beginning of Section 4.0. lists the bus access for each register.

This section describes the DMA registers. Unless otherwise stated, a PCIRST# sets each register to its default value.

3.2.1. DCOM—DMA COMMAND REGISTER

Address Offset: Channels 0–3—08h; Channels 4–7—0D0h
Default Value: 00h
Attribute: Write Only

This 8-bit register controls the configuration of the DMA. It is programmed by the microprocessor in the Program Condition and is cleared by PCIRST# or a Master Clear instruction. Note that disabling channels 4-7 also disables channels 0-3, since channels 0-3 are cascaded onto channel 4.

Bit	Description
7	**DACK# Assert Level (DACK#[3:0], [7:5]):** 1=Active high; 0=Active low.
6	**DREQ Sense Assert Level (DREQ[3:0], [7:5]):** 1=Active low; 0=Active high.
5	**Reserved:** Must be 0.
4	**DMA Group Arbitration Priority:** 1=Rotating priority; 0=Fixed priority.
3	**Reserved:** Must be 0.
2	**DMA Channel Group Enable:** 1=Disable; 0=Enable.
1:0	**Reserved:** Must be 0.

3.2.2. DCM—DMA CHANNEL MODE REGISTER

Address Offset: Channels 0–3—0Bh; Channels 4–7—0D6h
Default Value: Bits[7:2]=0, Bits[1:0]=undefined
Attribute: Write Only

Each channel has a DMA Channel Mode Register. The Channel Mode Registers provide control over DMA Transfer type, transfer mode, address increment/decrement, and autoinitialization.

Bit	Description
7:6	**DMA Transfer Mode:** Each DMA channel can be programmed in one of four modes. Channel 4 defaults to cascade mode and cannot be programmed for any mode other than cascade mode. **Bits[7:6]** — **Transfer Mode** 00 — Demand mode 01 — Single mode 10 — Block mode 11 — Cascade mode
5	**Address Increment/Decrement Select:** 0=Increment; 1=Decrement.
4	**Autoinitialize Enable:** 1=Enable; 0=Disable.

Bit	Description
3:2	**DMA Transfer Type:** When bits[7:6]=11, the transfer type bits are irrelevant. **Bits[3:2] Transfer Type** — 00 Verify transfer; 01 Write transfer; 10 Read Transfer; 11 Illegal
1:0	**DMA Channel Select:** Bits[1:0] select the DMA Channel Mode Register written by bits[7:2]. **Bits[1:0] Channel** — 00 0 (4); 01 1 (5); 10 2 (6); 11 3 (7)

3.2.3. DCEM—DMA CHANNEL EXTENDED MODE REGISTER (82378ZB Only)

Address Offset: Channels 0-3 - 040Bh
Channels 4-7 - 04D6h
Default Value: Bits[1:0]=undefined, Bits[3:2]=00 for DMA1, Bits[3:2]=01 for DMA2, Bits[7:4]=0
Attribute: Write Only

For the SIO, each channel has a DMA Channel Extended Mode Register. The register is used to program the DMA device data size, timing mode, EOP input/output selection, and Stop Register selection. Bits [1:0] select the appropriate Channel Extend Mode Register and are not stored. Only bits[7:2] are stored in the register. Four timing modes are available: ISA-compatible, "A", "B", and "F". Timings "A", "B", and "F" are extended timing modes and can only be run to main memory. DMA cycles to ISA expansion bus memory defaults to compatible timing if the channel is programmed in an extended timing mode.

The default bit values for each DMA group are selected upon PCIRST#. A Master Clear or any other programming sequence will not set the default register settings. The default programmed values for DMA1 channels 0-3 are 8-bit I/O count by bytes, compatible timing, and EOP output. The default values for DMA2 channels 4-7 are 16-bit I/O count by words with shifted address, compatible timing, and EOP output.

Bit	Description
7	**Reserved.** Must be 0.
6	**EOP Input/Output Selection:** Bit 6 selects whether the EOP signal is to be used as an output during DMA transfers on this channel or an input. EOP is typically used as an output, as was available on the PC/AT. The input function was added to support data communication and other devices that would like to trigger an autoinitialize when a collision or some other event occurs. The direction of EOP is switched when DACK is changed (when a different channel is granted the bus). There may be some overlap of the SIO driving the EOP signal along with the DMA slave. However, during this overlap, both devices drive the signal to a low level (inactive). For example, assume channel 2 is about to go inactive (DACK negating) and channel 1 is about to go active. In addition, assume that channel 2 is programmed for "EOP OUT" and channel 1 is programmed for "EOP IN". When channel 2's DACK is negated and channel 1's DACK is asserted, the SIO may be driving EOP to a low value on behalf of channel 2. At the same time the device connected to channel 1 is driving EOP in to the SIO, also at an inactive level. This overlap only lasts until the SIO EOP output buffer is tri-stated, and does not effect the DMA operation. Upon PCIRST#, bit 6 is set to 0 - EOP output selected.

Bit	Description
5:4	**DMA Cycle Timing Mode.** The SIO supports four DMA transfer timings: ISA-compatible, type "A", "B", and "F". Each timing and its corresponding code are described below. Upon PCIRST#, compatible timing is selected and the value of these bits is "00". The cycle timings noted below are for a SYSCLK (8.33 MHz, maximum SYSCLK frequency). DMA cycles to ISA expansion bus memory defaults to compatible timing if the channel is programmed in one of the performance timing modes (type "A", "B", or "F"). **Bits[5:4]=00: Compatible Timing** Compatible timing is provided for DMA slave devices that, due to some design limitation, cannot support one of the faster timings. Compatible timing runs at 9 SYSCLKs (1080 ns/single cycle) and 8 SYSCLKs (960 ns/cycle) during the repeated portion of a BLOCK or DEMAND mode transfer. **Bits[5:4]=01: Type "A" Timing** Type "A" timing is provided to allow shorter cycles to main memory (via the PCI Bus). Type "A" timing runs at 6 SYSCLKs (720 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 main memory. The I/O portion of the cycle (data setup on write, I/O read access time) is the same as with compatible cycles. The actual active command time is shorter. However, it is expected that the DMA devices that provide the data access time or write data setup time should not require excess IOR# or IOW# command active time. Because of this, most ISA DMA devices should be able to use type "A" timing. **Bits[5:4]=10: Type "B" Timing** Type "B" timing is provided for 8/16-bit ISA DMA devices that can accept faster I/O timing. Type "B" only works with fast main memory. Type "B" timing runs at 5 SYSCLKs (600 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 Type "B" timing the cycles are shortened so that the data setup time on I/O write cycles is shortened and the I/O read access time is required to be faster. **Bits[5:4]=11: Type "F" Timing** Type "F" timing provides high performance DMA transfer capability. Type "F" timing runs at 3 SYSCLKs (360 ns/single cycle) during the repeated portion of a BLOCK or DEMAND mode transfer, resulting in a maximum data transfer rate of 8.33 Mbytes/second.

Bit	Description
3:2	**Addressing Mode.** The SIO supports both 8 and 16 bit DMA device data sizes. Three data size options are programmable with bits[3:2]. Both the 8-bit I/O, "count by bytes" mode and the 16-bit I/O, "count by words" (address shifted) mode are ISA compatible. The 16-bit I/O, "count by bytes" mode is offered as an extension of the ISA compatible modes. Bits[3:2]=10 is reserved. Byte assembly/disassembly is performed by the ISA control unit. Each of the data transfer size modes is discussed below. **Bits[3:2]=00: 8-bit I/O, "Count By Bytes" Mode** In 8-bit I/O, "count by bytes" mode, the Current Address Register can be programmed to any address. The Current Byte/Word Count Register is programmed with the "number of bytes minus 1" to transfer. **Bits[3:2]=01: 16-bit I/O, "Count By Words" (Address Shifted) Mode** In "count by words" mode (address shifted), the Current Address Register can be programmed to any even address, but must be programmed with the address value shifted right by one bit. The Low Page and High Page Registers are not shifted during DMA transfers. Thus, the least significant bit of the Low Page Register is ignored when the address is driven out onto the bus. The Current Byte/Word Count Register is programmed with the number of words minus 1 to be transferred. **Bits[3:2]=10: Reserved** **Bits[3:2]=11: 16-Bit I/O, "Count By Bytes" Mode** In 16-bit "count by bytes" mode, the Current Address Register can be programmed to any byte address. For most DMA devices, however, it should be programmed only to even addresses. If the address is programmed to an odd address, the DMA controller does a partial word transfer during the first and last transfer, if necessary. The bus controller does the Byte/Word assembly necessary to write any size memory device. In this mode, the Current Address Register is incremented or decremented by two and the byte count is decremented by the number of bytes transferred during each bus cycle. The Current Byte/Word Count Register is programmed with the "number of bytes minus 1" to be transferred. This mode is offered as an extension of the two ISA compatible modes discussed above. This mode should only be programmed for 16-bit ISA DMA slaves.
1:0	**DMA Channel Select.** Bits[1:0] select the particular channel that will have its DMA Channel Extend Mode Register programmed with bits[7:2]. **Bits [1:0]** — **Channel** 00 — 0 (4) 01 — 1 (5) 10 — 2 (6) 11 — 3 (7)

3.2.4. DR—DMA REQUEST REGISTER

Address Offset: Channels 0–3—09h; Channels 4–7—0D2h
Default Value: Bits[1:0]=undefined, Bits[7:2]=0
Attribute: Write Only

Each channel has a request bit in one of the two DMA Request Registers. The Request Register is used by software to initiate a DMA request. The DMA responds to the software request as though DREQ[x] is asserted. These requests are non-maskable and subject to prioritization by the priority encoder network. The entire register is set to 0 upon PCIRST# or a Master Clear. It is not affected upon a RSTDRV output. 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:** 0=Resets the individual software DMA channel request bit. 1=Sets the request bit. Generation of a TC also sets this bit to 0.
1:0	**DMA Channel Select:** Bits[1:0] select the DMA channel mode register to program with bit 2. **Bits[1:0]** **Channel** **Bits[1:0]** **Channel** 00 0 10 2 (6) 01 1 (5) 11 3 (7)

3.2.5. MASK REGISTER—WRITE SINGLE MASK BIT

Address Offset: Channels 0–3—0Ah; Channels 4–7—0D4h
Default Value: Bits[1:0]=undefined, Bit 2=1, Bits[7:3]=0
Attribute: Write Only

Each DMA channel has a mask bit that enables/disables an incoming DMA channel service request DREQ[x]. Two Mask Registers store the current mask status for DMA1 and DMA2. Setting the mask bit disables the incoming DREQ[x] for that channel. Clearing the mask bit enables the incoming DREQ[x]. A channel's mask bit is automatically set when the Current Byte/Word Count Register reaches terminal count (unless the channel is programmed for autoinitialization). Each mask bit may also be set or cleared under software control. The entire register is also set by a PCIRST# or a Master Clear. Setting the entire register disables all DMA requests until a clear mask register instruction allows them to occur. This instruction format is similar to the format used with the DMA Request Register. Masking DMA channel 4 (DMA controller 2, channel 0) automatically masks DMA channels [3:0]

Bit	Description
7:3	**Reserved:** Must be 0.
2	**Channel Mask Select:** When bit 2 is set to a 1, DREQ is disabled for the selected channel. When bit 2 is set to a 0, DREQ is enabled for the selected channel.
1:0	**DMA Channel Select:** Bits[1:0] select the DMA Channel Mode Register for bit 2. **Bits[1:0]** **Channel** **Bits[1:0]** **Channel** 00 0 (4) 10 2 (6) 01 1 (5) 11 3 (7)

3.2.6. MASK REGISTER—WRITE ALL MASK BITS

Address Offset: Channels 0–3—0Fh; Channels 4–7—0DEh
Default Value: Bit[3:0]=1, Bit[7:4]=0
Attribute: Read/Write

A channel's mask bit is automatically set to 1 when the Current Byte/Word Count Register reaches terminal count (unless the channel is programmed for autoinitialization). Bits[3:0] are set to 1 by a PCIRST# or a Master Clear. Setting bits[3:0] to 1 disables all DMA requests until a clear mask register instruction enables the requests. Note that, masking DMA channel 4 (DMA controller 2, channel 0) will automatically mask DMA channels [3:0]. In addition, masking DMA controller 2 with a write to Port 0DEh will also mask DREQ assertions from DMA controller 1.

Bit	Description
7:4	**Reserved:** Must be 0.
3:0	**Channel Mask Bits:** 1=Disable the corresponding DREQ(s); 0=Enable the corresponding DREQ(s). **Bit** **Channel** **Bit** **Channel** 0 0 (4) 2 2 (6) 1 1 (5) 3 3 (7)

3.2.7. DS—DMA STATUS REGISTER

Address Offset: Channels 0–3—08h; Channels 4–7—0D0h
Default Value: 00h
Attribute: Read Only

Each DMA controller has a read-only DMA Status Register indicating which channels have reached terminal count and which channels have a pending DMA request.

Bit	Description
7:4	**Channel Request Status:** When a valid DMA request is pending for a channel (on its DREQ signal line), the corresponding bit is set to 1. When a DMA request is not pending for a particular channel, the corresponding bit is set to 0. The source of the DREQ may be hardware, 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** **Bit** **Channel** 4 0 6 2 (6) 5 1 (5) 7 3 (7)
3:0	**Channel Terminal Count Status:** 1=TC reached; 0=TC is not reached. **Bit** **Channel** **Bit** **Channel** 0 0 2 2 (6) 1 1 (5) 3 3 (7)

3.2.8. DMA BASE AND CURRENT ADDRESS REGISTERS (8237 COMPATIBLE SEGMENT)

Address Offset: DMA Channel 0—000h; DMA Channel 4—0C0h
DMA Channel 1—002h; DMA Channel 5—0C4h
DMA Channel 2—004h; DMA Channel 6—0C8h
DMA Channel 3—006h; DMA Channel 7—0CCh
Default Value: All bits undefined
Attribute: Read/Write

Each channel has a 16-bit Current Address Register. For the 82378ZB, this register contains the value of the 16 least significant bits of the full 32-bit address used during DMA transfers. For the 82379AB, this register contains the value of the 16 least significant bits of the full 27-bit address used during DMA transfers.

The address is automatically incremented or decremented after each transfer and the intermediate values of the address are stored in the Current Address Register during the transfer. This register is written to or read from by the PCI Bus or ISA Bus master in successive 8-bit bytes. The programmer must issue the "Clear Byte Pointer Flip-Flop" command to reset the internal byte pointer and correctly align the write prior to programming the Current Address Register. Autoinitialize takes place only after a TC or EOP.

For the 82378ZB in S/G mode, these registers store the lowest 16 bits of the current memory address. During an S/G transfer, the DMA will load a reserve buffer into the base memory address register.

Bit	Description
15:0	**Base and Current Address [15:0].** These bits represent address bits[15:0] used when foming the address for DMA transfers. Upon PCIRST# or Master Clear, the value of these bits is 0000h.

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

Address Offset: DMA Channel 0—001h DMA Channel 4—0C2h
DMA Channel 1—003h DMA Channel 5—0C6h
DMA Channel 2—005h DMA Channel 6—0CAh
DMA Channel 3—007h DMA Channel 7—0CEh
Default Value: All bits undefined
Attribute: Read/Write

This register determines the number of transfers to be performed. The actual number of transfers is one more than the number programmed in the Current Byte/Word Count Register. The Byte/Word count is decremented after each transfer. When the value in the register goes from zero to 0FFFFh, a TC is generated. Following the end of a DMA service the register may also be re-initialized by an autoinitialization back to its original value. Autoinitialize can only occur when a TC occurs. If it is not autoinitialized, this register has a count of FFFFh after TC.

For transfers to/from an 8-bit I/O, the Byte/Word count indicates the number of bytes to be transferred. For transfers to/from a 16-bit I/O, with shifted address, the Byte/Word count indicates the number of 16-bit words to be transferred. When the Extended Mode Register is programmed for transfers to/from a 16-bit I/O, the Byte/Word Count indicates the number of bytes to be transferred. The number of bytes does not need to be a multiple of two or four in this case.

For the 82378AB in S/G mode, these registers store the 16 bits of the current Byte/Word count. During S/G transfer, the DMA will load a reserve buffer into the base Byte/Word Count register.

Bit	Description
15:0	**Base and Current Byte/Word Count:** These bits represent the 16 byte/word count bits used when counting down a DMA transfer. Upon PCIRST# or Master Clear, the value of these bits is 0000h.

3.2.10. DMA MEMORY BASE LOW PAGE AND CURRENT LOW PAGE REGISTERS

Address Offset: DMA Channel 0—087h; DMA Channel 5—08Bh
DMA Channel 1—083h; DMA Channel 6—089h
DMA Channel 2—081h; DMA Channel 7—08Ah
DMA Channel 3—082h;
Default Value: All bits undefined

The DMA Memory Low Page Register contains the eight second most-significant bits of the address (32-bit address for the 82378ZB and 27-bit address for the 82379AB). The register works in conjunction with the DMA controller's High Page Register and Current Address Register to define the complete address for the DMA channel. This register may be re-initialized by an autoinitialize back to its original value. Autoinitialize takes place only after a TC or EOP.

For the 82378ZB in S/G, these registers store the 8 bits from the third byte of the current memory address. During a Scater-Gather transfer, the DMA will load a reserve buffer into the base memory address register.

Bit	Description
7:0	**DMA Low Page and Base Low Page [23:16].** These bits represent the eight second most significant address bits when forming the full address for a DMA transfer. Upon PCIRST# or Master Clear, the value of these bits is 00h.

3.2.11. DMA MEMORY BASE HIGH PAGE AND CURRENT HIGH PAGE REGISTERS

Address Offset: DMA Channel 0—0487h; DMA Channel 5—048Bh
DMA Channel 1—0483h; DMA Channel 6—0489h
DMA Channel 2—0481h; DMA Channel 7—048Ah
DMA Channel 3—0482h;
Default Value: All bits undefined

This register works in conjunction with the DMA controller's Current Low Page Register and Current Address Register to define the complete address for the DMA channels and corresponds to the Current Address Register for each channel. This register may be autoinitialized back to its original value. Autoinitialize takes place only after a TC or EOP.

For the 82378ZB, this register contains the eight most significant bits of the 32-bit address. For the 82379AB, this register contains the three most significant bits of the 27-bit address.

This register is set to 0 during the programming of both the Current Low Page Register and the Current Address Register. Thus, if this register is not programmed after the other address and Low Page Registers are programmed, then its value is 00h. In this case, the DMA channel operates the same as an 82C37 (from an addressing standpoint). This is the address compatibility mode.

If the high 8 bits (3-bits for the 82379AB) of the address are programmed after the other addresses, then the channel modifies its operation to increment (or decrement) the entire 32-bit (27-bit for the 82379AB) address. This is unlike the 82C37 "Page" register in the original PCs which could only increment to a 64 Kbyte boundary

for 8-bit channels or 128 Kbyte boundary for 16-bit channels. This is extended address mode. In this mode, the ISA Bus controller generates the signals MEMR# and MEMW# only for addresses below 16 Mbytes.

For the 82378ZB in S/G, these registers store the 8 bits from the highest byte of the current memory address. During a S/G transfer, the DMA will load a reserve buffer into the base memory address register.

82378ZB:

Bit	Description
7:0	**DMA High Page and Base High Page [31:24].** These bits represent the eight most-significant address bits when forming the full 32-bit address for a DMA transfer. Upon PCIRST# or Master Clear, the value of these bits is 00h.

82379AB:

Bit	Description
7:3	**Reserved.**
2:0	**DMA High Page and Base High Page [26:24].** These bits represent the three most-significant address bits when forming the full 27-bit address for a DMA transfer. Upon PCIRST# or Master Clear, the value of these bits is 00h.

3.2.12. DMA CLEAR BYTE POINTER REGISTER

Address Offset: Channels 0–3—00Ch; Channels 4–7—0D8h
Default Value: All bits undefined
Attribute: Write Only

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

The Master Clear command clears the internal latch used to address the upper or lower byte of the 16-bit Address and Word Count Register. 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.

Bit	Description
7:0	**Clear Byte Pointer:** No specific pattern. Command enabled with a write to the I/O port address.

3.2.13. DMC—DMA MASTER CLEAR REGISTER

Address Offset: Channel 0–3—00Dh; Channel 4–7—0DAh
Default Value: All bits undefined
Attribute: Write Only

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

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

3.2.14. DCM—DMA CLEAR MASK REGISTER

Address Offset: Channel 0–3—00Eh; Channel 4–7—0DCh
Default Value: All bits undefined
Attribute: Write Only

This command clears the mask bits of all four channels, enabling them to accept DMA requests. 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 Register:** No specific pattern. Command enabled with a write to the I/O port address.

3.2.15. SCATTER/GATHER (S/G) COMMAND REGISTER (82378ZB Only)

Address Offset: Channels 0 default address—0410h; Channels 1 default address—0411h; Channels 2 default address—0412h; Channels 3 default address—0413h; Channels 5 default address—0415h Channels 6 default address—0416h Channels 7 default address—0417h
Default Value: 00h
Attribute: Write Only, Relocatable

The S/G Command Register controls operation of the descriptor table aspect of scatter/gather transfers. This register can be used to start and stop a scatter/gather transfer. The register can also be used to select between IRQ13 and EOP to be asserted following a terminal count. The current scatter/gather transfer status can be read in the scatter/gather channel's corresponding S/G Status Register. After a PCIRST# or Master Clear, IRQ13 is disabled and EOP is enabled.

Bit	Description
7	**IRQ13/EOP Select.** Bit 7, if enabled via bit 6 of this register, selects whether EOP or IRQ13 is asserted at termination caused by a last buffer expiring. The last buffer can be either the last buffer in the list or the last buffer loaded in the DMA while it is suspended. If bit 7=1 (and bit 6=1), EOP is asserted when the last buffer is completed. If bit 7=0 (and bit 6=1), IRQ13 is asserted when the last buffer is completed. EOP can be used to alert an expansion bus I/O device that a scatter/gather termination condition was reached. The I/O device, in turn, can assert its own interrupt request line to invoke a dedicated interrupt handling routine. IRQ13 should be used when the CPU needs to be notified directly. Following PCIRST#, or Master Clear, the value stored for this bit is 1, and EOP is selected. Bit 6 must be set to a 1 to enable this bit during a S/G Command register write. When bit 6 is a 0 during the write, bit 7 will not have any effect on the current EOP/IRQ13 selection.
6	**IRQ13/EOP Programming Enable.** Enabling IRQ13/EOP programming allows initialization or modification of the S/G termination handling bits. When bit 6=0, bit 7 does not affect the state of IRQ13 or EOP assertion. When bit 6=1, bit 7 determines the termination handling following a terminal count.
5:2	**Reserved.** Must be 0.
1:0	**Scatter/Gather (S/G) Commands.** This 2-bit field is used to start and stop scatter/gather. **Bits[1:0]=00: No S/G operation** No S/G command operation is performed. Bits[7:6] may still be used to program IRQ13/EOP selection. **Bits[1:0]=01: Start S/G Command** The Start command initiates the scatter/gather process. Immediately after the start command is issued (setting bits[1:0] to 01), a request is issued to fetch the initial buffer from the descriptor table to fill the Base Register set in preparation for performing a transfer. The buffer prefetch request has the same priority with respect to other channels as the DREQ it is associated with. Within the channel, DREQ is higher in priority than a prefetch request. The Start command assumes the Base and Current Registers are both empty and will request a prefetch automatically. Note that this command also sets the S/G Status Register to S/G Active, Base Empty, Current Empty, not Terminated, and Next Null Indicator to 0. The EOP/IRQ13 bit will still reflect the last value programmed **Bits[1:0]=10: Stop S/G Command** The Stop command halts a S/G transfer immediately. When a Stop command is given, the Terminate bit in the S/G Status Register and the DMA channel mask bit are both set. **Bits[1:0]=11: Reserved**

3.2.16. SCATTER/GATHER (S/G) STATUS REGISTER (82378ZB Only)

Address Offset:	Channels 0 default address—0418h; Channels 1 default address—0419h; Channels 2 default address—041Ah; Channels 3 default address—041Bh;	Channels 5 default address—041Dh Channels 6 default address—041Eh Channels 7 default address—041Fh
Default Value:	00h	
Attribute:	Read Only, Relocatable	

The S/G Status Register contains information on the scatter/gather transfer status. This register provides dynamic status information on S/G transfer activity, the current and base buffer state, S/G transfer termination, and the End of the List indicator.

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

Bit	Description
7	**Next Link Null Indicator.** If the next scatter/gather descriptor fetched from memory during a fetch operation has the EOL value set to 1, the current value of the Next Link Register is not overwritten. Instead, bit 7 of the channel's S/G Status Register is set to a 1. If the fetch returns a EOL value set to 0, this bit is set to 0. This status bit is written after every fetch operation. Following PCIRST#, or Master Clear, this bit is set to 0. This bit is also cleared by an S/G Start Command write to the S/G Command Register.
6	**Reserved.**
5	**Issue IRQ13/EOP on Last Buffer.** When bit 5=0, EOP was either defaulted to at reset or selected through the S/G Command Register as the S/G process termination indicator. EOP is issued when a terminal count occurs or following the Stop Command. When bit 5=1, an IRQ13 is issued to alert the CPU of this same status.
4	**Reserved.**
3	**Scatter/Gather Base Register Status.** When bit 3=0, the Base Register is empty. When bit 3=1, the Base Register has a buffer link loaded. Note that the Base Register State may be set while the Current Register state is cleared. This condition occurs when the Current Register expires following a transfer. The Base Register will not be moved into the Current Register until the start of the next DMA transfer.
2	**Scatter/Gather Current Register Status.** When bit 2=0, the Current Register is empty. When bit 2=1, the Current Register has a buffer link loaded and is considered full. Following PCIRST#, bit 2 is set to 0.
1	**Reserved.**
0	**Scatter/Gather Active.** The S/G Active bit indicates the current S/G transfer status. Bit 0 is set to a 1 after an S/G Start Command is issued. Bit 0 is set to 0 before the Start Command is issued. Bit 0 is 0 after terminal count on the last buffer on the channel is reached. Bit 0 is also 0 after an S/G Stop Command has been issued. Following a PCIRST# or Master Clear, this bit is 0.

3.2.17. SCATTER/GATHER (S/G) DESCRIPTOR TABLE POINTER REGISTER (82378ZB Only)

Address Offset:	Channel 0 default address—0420–0423h; Channel 1 default address—0424–0424h; Channel 2 default address—0428–042Bh; Channel 3 default address—042C–042Ch;	Channel 5 default address—0434–0437h Channel 6 default address—0438–043Bh Channel 7 default address—043C–043Fh
Default Value:	All bits undefined	
Attribute:	Read/Write, Relocatable	

The S/G Descriptor Table Pointer Register contains the 32-bit pointer address to the first scatter/gather descriptor entry in the descriptor table in memory. Before the start of a S/G transfer, this register should be programmed to point to the first descriptor in the S/G Descriptor Table. Following a S/G Start command, the SIO reads the first SGD entry. Subsequently, at the end of the each buffer block transfer, the contents of the SGD Table pointer registers are incremented by 8 until the end of the SGD Table is reached.

The S/G Descriptor Table Pointer Registers can be programmed with a single 32-bit PCI write.

Following a prefetch to the address pointed to by the channel's S/G Descriptor Table Pointer Register, the new memory address is loaded into the Base Address Register, the new Byte Count is loaded into the Base Byte Count Register, and the newly fetched next scatter/gather descriptor replaces the current next scatter/gather value.

The end of the S/G Descriptor Table is indicated by an End of Table field having a MSB equal to 1. When this value is read during a scatter/gather descriptor fetch, the current scatter/gather descriptor value is not replaced. Instead, bit 7 of the channel's Status Register is set to a 1, when the EOL is read from memory.

Bit	Description
31:0	**Descriptor Table Pointer Address.** The S/G Descriptor Table Pointer Register contains a 32-bit pointer address to the main memory location where the software maintains the Scatter Gather Descriptors for the linked-list buffers. Bits[31:0] correspond to A[31:0] on the PCI.

3.2.18. SCATTER/GATHER (S/G) INTERRUPT STATUS REGISTER (82378ZB Only)

Address Offset:	040Ah
Default Value:	00h
Attribute:	Read Only, Relocatable
Size:	8 bits

The S/G Interrupt Status Register is a read only register and is used to indicate the source (channel) of a DMA S/G interrupt on IRQ13. The DMA controller drives IRQ13 active after reaching terminal count during a S/G transfer. It does not drive IRQ13 active during the initial programming sequence that loads the Base Registers.

Bit	Description
7	**Channel 7 Interrupt Status.** When this bit is set to a 1, Channel 7 has an interrupt due to a S/G Transfer; otherwise this bit is set to a 0.
6	**Channel 6 Interrupt Status.** When this bit is set to a 1, Channel 6 has an interrupt due to a S/G Transfer; otherwise this bit is set to a 0.
5	**Channel 5 Interrupt Status.** When this bit is set to a 1, Channel 5 has an interrupt due to a S/G Transfer; otherwise this bit is set to a 0.
4	**Reserved.** Read as 0.

Bit	Description
3	**Channel 3 Interrupt Status.** When this bit is set to a 1, Channel 3 has an interrupt due to a S/G Transfer; otherwise this bit is set to a 0.
2	**Channel 2 Interrupt Status.** When this bit is set to a 1, Channel 2 has an interrupt due to a S/G Transfer; otherwise this bit is set to a 0.
1	**Channel 1 Interrupt Status.** When this bit is set to a 1, Channel 1 has an interrupt due to a S/G Transfer; otherwise this bit is set to a 0.
0	**Channel 0 Interrupt Status.** When this bit is set to a 1, Channel 0 has an interrupt due to a S/G Transfer; otherwise this bit is set to a 0.

3.3. Timer Register Description

The SIO/SIO.A contains three counters that are equivalent to those found in the 82C54 Programmable Interval Timer. The Timer Registers control these counters and can be accessed from either the ISA Bus via ISA I/O space or the PCI Bus via PCI I/O space. This section describes the counter/timer registers on the SIO/SIO.A.

3.3.1. TCW—TIMER CONTROL WORD REGISTER

Address Offset: 043h
Default Value: All bits undefined
Attribute: Write Only

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

Following PCIRST#, the control words for each register are undefined. Each timer must be programmed to bring it into a known state. However, each counter OUT signal is set to 0 following PCIRST#. The SPKR output, interrupt controller input IRQ0 (internal), bit 5 of Port 061h, and the internally generated refresh request are each set to 0 following PCIRST#.

Bit	Description
7:6	**Counter Select:** **Bit[7:6]** **Function** — **Bit[7:6]** **Function** 00 Counter 0 select — 10 Counter 2 select 01 Counter 1 select — 11 Read Back Command
5:4	**Read/Write Select:** Bits[5:4] are the read/write control bits. The Counter Latch Command is selected when bits[5:4] are both 0. The read/write options include r/w least significant byte, r/w most significant byte, or r/w the LSB and then the MSB. The actual counter programming is done through the counter I/O port (040h, 041h, and 042h for counters 0, 1, and 2, respectively). **Bit[5:4]** **Function** — **Bit[5:4]** **Function** 00 Counter Latch Command — 10 R/W Most Significant Byte 01 R/W Least Significant Byte — 11 R/W LSB then MSB

Bit	Description
3:1	**Counter Mode Selection:** Bits[3:1] select one of six possible operating modes: **Bit[3:1] / Mode / Function** 000 / 0 / Out signal on end of count (=0) 001 / 1 / Hardware retriggerable one-shot X10 / 2 / Rate generator (divide by n counter) X11 / 3 / Square wave output 100 / 4 / Software triggered strobe 101 / 5 / Hardware triggered strobe
0	**Binary/BCD Countdown Select:** 0=Binary countdown. The largest possible binary count is 2^{16}. 1=Binary coded decimal (BCD) count is used. The largest BCD count allowed is 10^4.

Read Back Command

The Read Back Command is used to determine the count value, programmed mode, and current states of the OUT pin and Null count flag of the selected counter or counters. The Read Back Command is written to the Timer Control Word Register which latches the current states of the above mentioned variables. The value of the counter and its status may then be read by I/O access to the counter address. Note that the Timer Counter Register bit definitions are different during the Read Back Command than for a normal Timer Counter Register write.

Bit	Description
7:6	**Read Back Command:** When bits[7:6] are both 1, the Read Back Command is selected during a write to the Timer Control Word Register. Following the Read Back Command, I/O reads from the selected counter's I/O addresses produce the current latch status, the current latched count, or both if bits 4 and 5 are both 0.
5	**Latch Count of Selected Counters:** When bit 5=1, the current count value of the selected counters will be latched. When bit 4=0, the status will not be latched.
4	**Latch Status of Selected Counters:** When bit 4=1, the status of the selected counters will be latched. When bit 4=0, the status will not be latched.
3	**Counter 2 Select:** When bit 3=1, Counter 2 is selected for the latch command selected with bits 4 and 5. When bit 3=0, status and/or count will not be latched.
2	**Counter 1 Select:** When bit 2=1, Counter 1 is selected for the latch command selected with bits 4 and 5. When bit 2=0, status and/or count will not be latched.
1	**Counter 0 Select:** When bit 1=1, Counter 0 is selected for the latch command selected with bits 4 and 5. When bit 1=0, status and/or count will not be latched.
0	**Reserved:** Must be 0.

Counter Latch Command

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

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

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

NOTE

1. If a counter is programmed to read/wr ite two-byte counts, a program must not transfer control between reading the first and second byte to another routine that also reads from that same counter. Otherwise, an incorrect count will be read. Finish reading the latched two-byte count before transferring control to another routine.
2. The Timer Counter Register bit definitions are different during the Counter Latch Command than for a normal Timer Counter Register write.

Bit	Description
7:6	**Counter Selection:** Bits 6 and 7 are used to select the counter for latching. **Bit[7:6]** — **Function**: 00 latch counter 0 select; 01 latch counter 1 select; 10 latch counter 2 select; 11 Read Back Command select
5:4	**Counter Latch Command:** When bits[5:4]=00, the Counter Latch Command is selected during a write to the Timer Control Word Register. Following the Counter Latch Command, I/O reads from the selected counter's I/O addresses produce the current latched count.
3:0	**Reserved.** Must be 0.

3.3.2. INTERVAL TIMER STATUS BYTE FORMAT REGISTER

Address Offset: Counter 0—040h; Counter 1—041h; Counter 2—042h
Default Value: Bits[6:0]=X, Bit 7=0
Attribute: Read Only

Each counter's status byte can be read following an Interval Timer Read Back Command. If latch status is chosen (bit 4=0, Read Back Command) as a read back option for a given counter, the next read from the counter's Counter Access Ports Register returns the status byte.

Bit	Description
7	**Counter OUT Pin State:** 1=Pin is 1; 0=Pin is 0.
6	**Count Register Status:** This bit indicates when the last count written to the Count Register (CR) has been loaded into the counting element (CE). 0=Count has been transferred from CR to CE and is available for reading. 1=Count has not been transferred from CR to CE and is not yet available for reading.
5:4	**Read/Write Selection Status:** Bits[5:4] reflect the read/write selection made through bits[5:4] of the control register. The binary codes returned during the status read match the codes used to program the counter read/write selection. **Bit[5:4]** — **Function**: 00 Counter Latch Command; 01 R/W Least Significant Byte; 10 R/W Most Significant Byte; 11 R/W LSB then MSB

Bit	Description
3:1	**Mode Selection Status:** Bits[3:1] return the counter mode programming. **Bit[3:1] Mode Selected Bit[3:1] Mode Selected** 000 0 X11 3 001 1 100 4 X10 2 101 5
0	**Countdown Type Status:** 0=Binary countdown; 1=Binary coded decimal (BCD) countdown.

3.3.3. COUNTER ACCESS PORTS REGISTER

Address Offset: Counter 0—040h; Counter 1—041h; Counter 2—042h
Default Value: All bits undefined
Attribute: Read/Write

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

Bit	Description
7:0	**Counter Port Bit[x]:** Each counter I/O port address is used to program the 16-bit Count Register. The order of programming, either LSB only, MSB only, or LSB then MSB, is defined with the Interval Counter Control Register at I/O port address 043h. The counter I/O port is also used to read the current count from the Count Register, and return the status of the counter programming following a Read Back Command.

3.3.4. BIOS TIMER REGISTER

Register Location: Default=78–7Bh (Dword aligned)
Default Value: 0000xxxxh
Attribute: Read/Write, Programmable

A write to the BIOS Timer initiates a counting sequence. The timer can be initiated by writing either a 16-bit data portion or the entire 32-bit register (the upper 16 bits are don't cares). Bits[15:0] can be written with the initial count value to start the timer or read to check the current count value. It is the programmer's responsiblity to ensure that all 16 bits are written at the same time. After data is written into the BIOS timer, the timer starts decrementing until it reaches zero. It will "freeze" at zero until the new count value is written.

The BIOS Timer consists of a single 32-bit register mapped in the I/O space on the location determined by the value written into the BIOS Timer Base Address Register. Bit 0 of the BIOS Timer Base Address Register enables/disables accesses to the BIOS Timer and must be 1 to enable access to the BIOS Timer Register. When the BIOS Timer is enabled, PCI accesses to the BIOS Timer Register do not flow through to the ISA Bus. If the BIOS Timer is disabled, accesses to the addresses assigned to the BIOS Timer Register flow through to the ISA Bus. Note, however, that the counter continues to count normally.

Bit	Description
31:16	**Reserved.** Read as 0.
15:0	**Timer Count Value**

3.4. Interrupt Controller Register Description

The SIO/SIO.A contains an ISA compatible interrupt controller that incorporates the functionality of two 82C59 interrupt controllers. The interrupt registers control the operation of the interrupt controller and can be accessed from the PCI Bus via PCI I/O space. In addition, some of the registers can be accessed from the ISA Bus via ISA I/O space.

3.4.1. ICW1—INITIALIZATION COMMAND WORD 1 REGISTER

Register Location: INT CNTRL-1—020h; INT CNTRL-2—0A0h
Default Value: All bits undefined
Attribute: Write Only

A write to Initialization Command Word 1 starts the interrupt controller initialization sequence. Addresses 020h and 0A0h are referred to as the base addresses of CNTRL-1 and CNTRL-2, respectively. An I/O write to the CNTRL-1 or CNTRL-2 base address with bit 4 equal to 1 is interpreted as ICW1. For SIO/SIO.A-based ISA systems, three I/O writes to "base address + 1" must follow the ICW1. The first write to "base address + 1" performs ICW2, the second write performs ICW3, and the third write performs ICW4.

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

a. The edge sense circuit is reset. This means that following initialization, an interrupt request (IRQ) input must make a low-to-high transition to generate an interrupt.
b. The Interrupt Mask Register is cleared.
c. IRQ7 input is assigned priority 7.
d. The slave mode address is set to 7.
e. Special Mask Mode is cleared and Status Read is set to IRR.
f. If IC4 was set to 0, then all functions selected by ICW4 are set to 0. However, ICW4 must be programmed in the SIO/SIO.A implementation of this interrupt controller, and IC4 must be set to a 1.

Bit	Description
7:5	**ICW/OCW Select:** These bits should be 000 when programming the SIO/SIO.A.
4	**ICW/OCW Select:** Bit 4 must be a 1 to select ICW1. After the fixed initialization sequence to ICW1, ICW2, ICW3, and ICW4, the controller base address is used to write to OCW2 and OCW3. Bit 4 is a 0 on writes to these registers. A 1 on this bit at any time will force the interrupt controller to interpret the write as an ICW1. The controller will then expect to see ICW2, ICW3, and ICW4.
3	**LTIM (Edge/Level Bank Select):** This bit is ignored by the SIO/SIO.Aand is read as a 1.
2	**ADI—WO:** Ignored for the SIO/SIO.A.
1	**SNGL (Single or Cascade):** This bit must be programmed to a 0 to indicate that two interrupt controllers are operating in cascade mode on the SIO/SIO.A.
0	**IC4 (ICW4 Write Required):** This bit must be set to a 1.

3.4.2. ICW2—INITIALIZATION COMMAND WORD 2 REGISTER

Address Offset: INT CNTRL-1—021h; INT CNTRL-2—0A1h
Default Value: All bits undefined
Attribute: Write Only

ICW2 is used to initialize the interrupt controller with the five most significant bits of the interrupt vector address.

Bit	Description
7:3	**Interrupt Vector Base Address:** Bits[7:3] define the base address in the interrupt vector table for the interrupt routines associated with each interrupt request level input.
2:0	**Interrupt Request Level:** When writing ICW2, these bits should all be 0.

3.4.3. ICW3—INITIALIZATION COMMAND WORD 3 REGISTER

Address Offset: INT CNTRL-1—021h
Default Value: All bits undefined
Attribute: Write Only

The meaning of ICW3 differs between CNTRL-1 and CNTRL-2. On CNTRL-1, the master controller, ICW3 indicates which CNTRL-1 IRQ line physically connects the 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	**Not Used:** These bits must be programmed to 0.
2	**Cascaded Interrupt Controller IRQ Connection:** Bit 2 must always be programmed to a 1 selecting cascade mode.
1:0	**Not Used:** These bits must be programmed to 00.

3.4.4. ICW3—INITIALIZATION COMMAND WORD 3 REGISTER

Address Offset: INT CNTRL-2—0A1h
Default Value: All bits undefined
Attribute: Write Only

On CNTRL-2 (the slave controller), ICW3 is the slave identification code broadcast by CNTRL-1 from the trailing edge of the first INTA# pulse to the trailing edge of the second INTA# pulse.

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

Address Offset: INT CNTRL-1—021h; INT CNTRL-2—0A1h
Default Value: 01h
Attribute: Write Only
Size: 8 bits

Both SIO/SIO.A interrupt controllers must have ICW4 programmed as part of their initialization sequence.

Bit	Description
7:5	**Reserved.** Must be 0.
4	**Special Fully Nested Mode (SFNM):** Bit 4, SFNM, should normally be disabled by writing a 0 to this bit. If SFNM=1, the special fully nested mode is programmed.
3	**Buffered Mode (BUF):** Must be programmed to 0 selecting non-buffered mode.
2	**Master/Slave in Buffered Mode:** Bit 2 should always be programmed to 0. Bit not used.
1	**Automatic End of Interrupt (AEOI):** This bit should normally be programmed to 0. This is the normal end of interrupt. If this bit is 1, the automatic end of interrupt mode is programmed.
0	**Microprocessor Mode:** The Microprocessor Mode bit must be programmed to 1 indicating an 80x86-based system.

3.4.6. OCW1—OPERATIONAL CONTROL WORD 1 REGISTER

Address Offset: INT CNTRL-1—021hI; NT CNTRL-2—0A1h
Default Value: 00h
Attribute: Read/Write

OCW1 sets and clears the mask bits in the Interrupt Mask Register (IMR). Each interrupt request line may be selectively masked or unmasked any time after initialization. A single byte is written to this register. Each bit position in the byte represents the same-numbered channel: bit 0=IRQ[0], bit 1=IRQ[1] and so on. Setting the bit to a 1 sets the mask, and clearing the bit to a 0 clears the mask. Note that masking IRQ[2] on CNTRL-1 will also mask all of controller 2's interrupt requests (IRQ8-IRQ15). Reading OCW1 returns the controller's mask register status. The IMR stores the bits which mask the interrupt lines to be masked. The IMR operates on the IRR. Masking of a higher priority input will not affect the interrupt request lines of lower priority. Unlike status reads of the ISR and IRR, for reading the IMR, no OCW3 is needed. The output data bus will contain the IMR whenever I/O read is active and the I/O port address is 021h or 0A1h (OCW1). All writes to OCW1 must occur following the ICW1-ICW4 initialization sequence, since the same I/O ports are used for OCW1, ICW2, ICW3 and ICW4.

Bit	Description
7:0	**Interrupt Request Mask (Mask [7:0]):** When a 1 is written to any bit in this register, the corresponding IRQx line is masked. For example, if bit 4 is set to a 1, then IRQ4 will be masked. Interrupt requests on IRQ4 will not set channel 4's interrupt request register (IRR) bit as long is the channel is masked. When a 0 is written to any bit in this register, the corresponding IRQx mask bit is cleared, and interrupt requests will again be accepted by the controller. Note that masking IRQ2 on CNTRL-1 will also mask the interrupt requests from CNTRL-2, which is physically cascaded to IRQ2.

3.4.7. OCW2—OPERATIONAL CONTROL WORD 2 REGISTER

Address Offset: INT CNTRL-1—020h; INT CNTRL-2—0A0h
Default Value: Bit[4:0]=undefined, Bit[7:5]=001
Attribute: Write Only

OCW2 controls both the Rotate Mode and the End of Interrupt Mode, and combinations of the two. The three high order bits in an OCW2 write represent the encoded command. The three low order bits are used to select individual interrupt channels during three of the seven commands. The three low order bits (labeled L2, L1 and L0) are used when bit 6 is set to a 1 during the command. Following a PCIRST# and ICW initialization, the controller enters the fully nested mode of operation. Non-specific EOI without rotation is the default. Both rotation mode and specific EOI mode are disabled following initialization.

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>7:5</td><td>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</td></tr>
<tr><td>4:3</td><td>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.</td></tr>
<tr><td>2:0</td><td>Interrupt Level Select (L2, L1, L0): L2, L1, and L0 determine the interrupt level acted upon when the SL bit is active (bit 6). When the SL bit is inactive 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)</td></tr>
</table>

3.4.8. OCW3—OPERATIONAL CONTROL WORD 3 REGISTER

Address Offset: INT CNTRL-1—020h; INT CNTRL-2—0A0h
Default Value: Bit[6,0]=0, Bit[7,4:2]=undefined, Bit[5,1]=1
Attribute: Read/Write

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

Bit	Description
7	**Reserved:** Must be 0.
6	**SMM (Special Mask Mode):** If ESMM=1 and SMM=1 the interrupt controller enters Special Mask Mode. If ESMM=1 and SMM=0, the interrupt controller is in normal mask mode. When ESMM=0, SMM has no effect.
5	**ESMM (Enable Special Mask Mode):** 1=Enable SMM bit; 0=Disable SMM bit.
4:3	**OCW3 Select:** Must be programmed to 01 selecting OCW3.
2	**Poll Mode Command:** 0=Disable Poll command. When bit 2=1, the next I/O read to the interrupt controller is treated as an interrupt acknowledge cycle representing the highest priority level requesting service.
1:0	**Register Read Command:** Bits[1:0] provide control for reading the In-Service Register (ISR) and the Interrupt Request Register (IRR). When bit 1=0, bit 0 will not affect the register read selection. When bit 1=1, bit 0 selects the register status returned following an OCW3 read. If bit 0=0, the IRR will be read. If bit 0=1, the ISR will be read. Following ICW initialization, the default OCW3 port address read will be "read IRR". To retain the current selection (read ISR or read IRR), always write a 0 to bit 1 when programming this register. The selected register can be read repeatedly without reprogramming OCW3. To select a new status register, OCW3 must be reprogrammed prior to attempting the read. **Bit[1:0]** **Function** **Bit[1:0]** **Function** 00 No Action 10 Read IRQ Register 01 No Action 11 Read IS Register

3.5. Control Registers

This section contains NMI Registers, a real-time clock register, Port 92 Register, and the Digital Output Register.

3.5.1. NMISC—NMI STATUS AND CONTROL REGISTER

Address Offset: 061h
Default Value: 00h
Attribute: Read/Write

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

Bit	Description
7	**SERR# Status—RO:** 1=SERR# pulsed. 0=No SERR#. Bit 7 is set if a system board agent (PCI devices or main memory) detects a system board error and pulses the PCI SERR# line. This interrupt is enabled by setting bit 2 to 0. To reset the interrupt, set bit 2 to 0 and then set it to 1. When writing to Port 061h, bit 6 must be a 0.
6	**IOCHK# NMI Source Status—RO:** Bit 6 is set if an expansion board asserts IOCHK# on the ISA/SIO Bus. This interrupt is enabled by setting bit 3 to 0. To reset the interrupt, set bit 3 to 0 and then set it to 1. When writing to Port 061h, bit 6 must be a 0.
5	**Timer Counter 2 OUT Status—RO:** The Counter 2 OUT signal state is reflected in bit 5. The value on this bit following a read is the current state of the Counter 2 OUT signal. Counter 2 must be programmed following a PCIRST# for this bit to have a determinate value. When writing to Port 061h, bit 5 must be a 0.
4	**Refresh Cycle Toggle—RO:** The Refresh Cycle Toggle signal toggles from either 0 to 1 or 1 to 0 following every refresh cycle. This read-only bit is a 0 following PCIRST#. When writing to Port 061h, bit 4 must be a 0.
3	**IOCHK# NMI Enable—R/W:** 1=Clear and Disable; 0=Enable IOCHK# NMI's.
2	**PCI SERR# Enable—R/W:** 1=Clear and Disable. 0=Enable.
1	**Speaker Data Enable—R/W:** 1=SPKR output is 0; 1=SPKR output is the Counter 2 OUT value.
0	**Timer Counter 2 Enable—R/W:** 0=Disable; 1=Enable

3.5.2. NMI ENABLE AND REAL-TIME CLOCK ADDRESS REGISTER

Address Offset: 070h
Default Value: Bit[6:0]=undefined, Bit 7=1
Attribute: Write Only

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

Bit	Description
7	**NMI Enable:** 1=Disable; 0=Enable.
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.

3.5.3. PORT 92 REGISTER

Address Offset: 92h
Default Value: 24h
Attribute: Read/Write

This register is used to support the alternate reset (ALT_RST#) and alternate A20 (ALT_A20) functions. This register is only accessible if bit 6 in the Utility Bus Chip Select B Register is set to a 1. Reads and writes to this register location flow through to the ISA Bus.

Bit	Description
7:6	**Reserved:** Read as 0s.
5	**Reserved:** Read as 1.
4:3	**Reserved:** Read as 0s.
2	**Reserved:** Read as 1.
1	**ALT_A20 Signal Control—R/W:** 0=ALT_A20 signal negated (low). 1=ALT_A20 signal asserted (high).
0	**Alternate System Reset—R/W:** This read/write bit provides an alternate system reset function. This function provides an alternate means to reset the system CPU to effect a mode switch from Protected Virtual Address Mode to the Real Address Mode. This provides a faster means of reset than is provided by the Keyboard controller. This bit is set to a 0 by a system reset. Writing a 1 to this bit will cause the ALT_RST# signal to pulse active (low) for approximately 4 SYSCLK's. Before another ALT_RST# pulse can be generated, this bit must be written back to a 0.

3.5.4. DIGITAL OUTPUT REGISTER

Address Offset: 03F2h (Primary), 0372h (Secondary)
Default Value: Bit[7:4,2:0]=undefined, Bit 3=0
Attribute: Write only

This register is used to prevent UBUSOE# from responding to DACK2# during a DMA read access to a floppy controller on the ISA Bus. If a second floppy (residing on the ISA Bus) is using DACK2# in conjunction with a floppy on the Utility Bus, this prevents the floppy on the Utility Bus and the Utility Bus transceiver from responding to an access targeted for the floppy on the ISA Bus. This register is also located in the floppy controller device. Reads and writes to this register location flow through to the ISA Bus.

Bit	Description
7:4	**Not Used:** These bits exist in the floppy controller.
3	**DMA Enable:** When this bit is a 1, the assertion of DACK# will result in UBUSOE# being asserted. If this bit is 0, DACK2# has no effect on UBUSOE#. This port bit also exists on the floppy controller. This bit defaults to disable (0).
2:0	**Not Used:** These bits exist in the floppy controller.

3.5.5. RESET UBUS IRQ1/IRQ12 REGISTER

Address Offset: 60h
Default Value: N/A
Attribute: Read only

This address location (60h) is used to clear the mouse and keyboard interrupt functions to the CPU. Reads to this address are monitored by the SIO/SIO.A. When the mouse interrupt function is enabled (bit 4 of the ISA Clock Divisor Register is 1), the mouse interrupt function is provided on the IRQ12/M input signal. In this mode, a mouse interrupt generates an interrupt through IRQ12/M to the Host CPU. A read of 60h releases IRQ12. If bit 4=0 in the ISA Clock Divisor Register, a read of address 60h has no effect on IRQ12/M. Reads and writes to this register flow through to the ISA Bus. For additional information, see the IRQ12/M description in Signal Description section. A read of 60h always releases IRQ1, regardless of the setting of bit 4 in the ISA Clock Divisor Register.

Bit	Description
7:0	**Reset IRQ1/IRQ12:** No specific pattern. A read of address 60h executes the command.

3.5.6. COPROCESSOR ERROR REGISTER

Address Offset: F0h
Default Value: N/A
Attribute: Write only

This address location (F0h) is used when the SIO/SIO.A is programmed for coprocessor error reporting (bit 5 of the ISA Clock Divisor Register is 1). Writes to this address are monitored by the SIO/SIO.A. In this mode, the SIO/SIO.A generates an interrupt (INT) to the CPU when it receives an error signal (FERR# asserted) from the CPU's coprocessor. Writing address F0h, when FERR# is asserted, causes the SIO/SIO.A to assert IGNNE# and negate IRQ13. IGNNE# remains asserted until FERR# is negated. If FERR# is not asserted, writing to address F0h does not effect IGNNE#. Reads and writes to this register flow through to the ISA Bus. For additional information, see the IGNNE# description in the Signal Description section

Bit	Description
7:0	**Reset IRQ12:** No specific pattern. A write to address F0h executes the command.

3.5.7. ELCR—EDGE/LEVEL CONTROL REGISTER

Address Offset: INT CNTRL-1—04D0h; INT CNTRL-2—04D1h
Default Value: 00h
Attribute: Read/Write

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. Each IRQ that a PCI interrupt is steered into (see the PIRQ Route Control Register) must have it's interrupt set to level sensitive.

Bit	
7:0	**Edge/Level Select:** 0=Edge sensitive interrupt; 1=Level sensitive.

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

* Must be 0 when written.

3.6. Power Management Registers

This section describes the two power management registers (APMS and APMC) that are located in normal I/O space. These registers are accessed via the CPU or PCI Bus with 8 bit accesses. Note that the rest of the power management registers are part of the SIO/SIO.A configuration registers.

3.6.1. APMC—ADVANCED POWER MANAGEMENT CONTROL PORT

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

This register passes data (APM Commands) between the OS and the SMI handler. In addition, writes can generate an SMI and reads can cause STPCLK# to be asserted. The SIO/SIO.A operation is not affected 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

This register passes status information between the OS and the SMI handler. The SIO/SIO.A operation is not affected 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 (82379AB Only)

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 at the beginning of the Register 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 (82379AB Only)

Memory Address: FEC0 xy00h (xy=See APICBASE Register)
Default Value: 00h
Attribute: Read/Write

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 (82379AB Only)

Memory Address: FEC0 xy10h (xy=See APICBASE Register)
Default Value: 00h
Attribute: Read/Write

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 (82379AB Only)

Address Offset: 00h
Default Value: 00000000h
Attribute: Read/Write

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 VERSION REGISTER (82379AB Only)

Address Offset: 01h
Default Value: 000F0011h
Attribute: Read Only

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 82379AB, this value is 0Fh.
15:8	**Reserved**
7:0	**APIC VERSION:** This 8 bit field identifies the implementation version. The version number for the 82379AB is 11h.

3.7.5. APICARB—I/O APIC ARBITRATION REGISTER (82379AB Only)

Address Offset: 02h
Default Value: 00000000h
Attribute: Read Only

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

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

3.7.6. IOREDTBL[15:0]—I/O REDIRECTION TABLE REGISTERS (82379AB Only)

Address Offset:

10-11h (IOREDTBL0)	1C-1Dh (IOREDTBL6)	26-27h (IOREDTBL11)
12-13h (IOREDTBL1)	1E-1Fh (IOREDTBL7)	28-29h (IOREDTBL12)
14-15h (IOREDTBL2)	20-21h (IOREDTBL8)	2A-2Bh (IOREDTBL13)
16-17h (IOREDTBL3)	22-23h (IOREDTBL9)	2C-2Dh (IOREDTBL14)
18-19h (IOREDTBL4)	24-25h (IOREDTBL10)	2E-2Fh (IOREDTBL15)
1A-1Bh (IOREDTBL5)		

Default Value: xx000000 00010xxxh
Attribute: Read/Write

Each of the 16 I/O Redirection Table entry registers 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—R/W:** 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—R/W:** 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 affect 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.

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>10:8</td><td>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 INT 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 INT 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 INT 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.</td></tr>
<tr><td>7:0</td><td>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–FEh.</td></tr>
</table>

4.0. FUNCTIONAL DESCRIPTION

This section describes the SIO/SIO.A functions and hardware interfaces including memory and I/O address mapping, PCI interface, ISA interface, system arbitration, DMA, interrupt controller, advanced interrupt controller, timer/counters, power management, and the Utility Bus.

4.1. Memory and I/O Address Map

The SIO/SIO.A interfaces to the PCI and ISA Buses. Positive decode is provided for certain PCI master memory and I/O accesses. The SIO/SIO.A also provides positive decode for certain DMA and ISA master memory and I/O accesses.

4.1.1. MEMORY ADDRESS MAP (GENERATING MEMCS#)

PCI Master Access

The SIO/SIO.A decodes accesses to main memory and generates a memory chip select (asserts MEMCS#), if enabled. Various memory regions listed below can be enabled/disabled by programming the MCSCON, MCSBOH, MCSTOH, MCSTOM, MAR1, MAR2, and MAR3 Registers. Accesses within these enabled regions generate a MEMCS# signal that can be used by the host bridge to know when to forward PCI cycles to main memory.

- 0–512 Kbytes memory (can only be disabled if MEMCS# is completely disabled)
- 512–640 Kbytes memory
- (1 Mbyte-64 Kbytes) to 1 Mbyte memory (BIOS Area)
- 768–918 Kbytes in 16-Kbyte sections (total of 8 sections)
- 918–983 Kbytes in 16-Kbyte sections (total of 4 sections)
- 1 Mbyte-to-programmable boundary on 2-Mbyte increments from 2–512 Mbytes
- Programmable memory hole in 64-Kbyte increments between from 1–16 Mbytes

The SIO/SIO.A generates MEMCS# from the PCI address. MEMCS# is generated from the clock edge after FRAME# is sampled active. MEMCS# will only go active for one PCI clock period. The SIO/SIO.A does not take any other action as a result of this decode other than generating MEMCS#. It is the responsibility of the device using the MEMCS# signal to generate DEVSEL#, TRDY# and any other cycle response. The device using MEMCS# will always generate DEVSEL# on the next clock. This fact can be used to avoid an extra clock delay in the subtractive decoder described in the next section.

ISA/DMA Access

ISA master or DMA accesses that are positively decoded are forwarded to the PCI Bus. Various memory regions listed below can be enabled/disabled by programming the IADCON, IADBOH, IADTOH, and IADRBE Registers. A memory address above 16 Mbytes will be forwarded to the PCI Bus automatically. This is possible only during DMA cycles in which the DMA has been programmed for addressing above 16 Mbytes.

- 0–512 Kbytes
- 512–640 Kbytes
- 640–768 Kbytes (Video buffer)
- 768–896 Kbytes in eight 16-Kbyte sections (Expansion ROM)
- 896–960 Kbytes (lower BIOS area)

- 1-to-X Mbytes (up to 16 Mbytes) within which a hole can be opened. Accesses to the hole are not forwarded to PCI. The top of the region can be programmed on 64 Kbyte boundaries up to 16 Mbytes. The hole can be between 64 Kbytes and 8 Mbytes in size in 64-Kbyte increments located on any 64-Kbyte boundary.
- Greater than 16 Mbytes are forwarded to PCI

4.1.2. BIOS MEMORY SPACE

PCI Master Access

The SIO/SIO.A supports 512 Kbytes of BIOS space. This includes the normal 128-Kbyte space plus an additional 384-Kbyte Bios space (known as the enlarged BIOS area). The 128-Kbyte BIOS memory space is located at 000E0000–000FFFFFh (top of 1 Mbyte), and is aliased at FFFE0000–FFFFFFFFh (top of 4 Gbytes) and FFEE0000–FFEFFFFFh (top of 4 Gbytes-1 Mbyte). This 128-Kbyte block is split into two 64-Kbyte blocks. The top 64 Kbytes is always enabled while the bottom 64 Kbytes can be enabled or disabled (the aliases automatically match). When the lower 64-Kbyte BIOS space (000E0000–000EFFFFh) is enabled, accesses to this space generate a BIOS chip select (asserts BIOSCS#). Access to the upper 64 Kbytes is controlled by bit 6 in the ISA Clock Divisor Register and bit 4 in the MEMCS# Control Register.

When PCI master accesses to the 128-Kbyte BIOS space at 4 Gbytes - 1 Mbyte are forwarded to the ISA Bus, the LA20 line is driven to a 1 to avoid aliasing at the 15-Mbyte area. The 4 Gbytes - 1 Mbyte BIOS region accounts for the condition when A20M# is asserted and an ALT-CTRL-DEL reset is generated. The CPU's reset vector will access 4 Gbyte - 1 Mbyte. When this gets forwarded to ISA, AD[32:24] are truncated and the access is aliased to 16 Mbytes - 1 Mbyte = 15 Mbyte space. If ISA memory is present at 15 Mbytes, there will be contention. Forcing LA20 high aliases this region to 16 Mbytes. The alias here is permissible since this is the 80286 reset vector location.

The additional 384-Kbyte region (FFF80000–FFFDFFFFh) can only be accessed by PCI masters. When enabled via the UBCSA Register, memory accesses within this region are forwarded to the ISA Bus and encoded BIOSCS# generated. When forwarded to the ISA Bus, the PCI AD[23:20] signals will be propagated to the ISA LA[23:20] lines as all 1s which will result in aliasing this 512-Kbyte region at the top of the 16-Mbyte space. To avoid contention, ISA add-in memory must not be present in this space.

All PCI accesses to enabled BIOS space are forwarded to the ISA Bus. Note that PCI burst reads from the BIOS space invoke "disconnect target termination", in order to meet the PCI incremental latency guidelines.

ISA/DMA Access

ISA masters can only access BIOS in the 000E0000–000FFFFFh region. ISA originated accesses to the enabled 64-Kbyte sections of the BIOS space (000E0000h–000FFFFFh) generate the encoded BIOSCS# signal. ISA originated cycles arel not forwarded to the PCI Bus. Encoded BIOSCS# is combinatorially generated from the ISA SA and LA address bus. Encoded BIOSCS# is disabled during refresh and DMA cycles.

4.1.3. I/O ACCESSES

For PCI master accesses, The SIO/SIO.A positively decodes I/O addresses for registers contained within the SIO/SIO.A (exceptions: 60h, 70h, 92h, 3F2h, 372h, and F0h). The SIO/SIO.A also provides positive decode for ISA masters to most of the ISA-Compatible Registers. Refer to the Register Description section for details on accessing the SIO/SIO.A internal registers. Note that for the SIO.A, the APIC registers are memory mapped and accesses to these registers are also described in the Register Description section.

4.1.4. SUBTRACTIVELY DECODED CYCLES TO ISA

The addresses that reside on the ISA Bus could be highly fragmented. For this reason, subtractive decoding is used to forward PCI cycles to the ISA Bus. An inactive DEVSEL# will cause the SIO/SIO.A to forward the PCI cycle to the ISA Bus. The DEVSEL# sample point can be configured for three different settings—fast, typical, or slow. Note that when unclaimed cycles are forwarded to the ISA Bus, the SIO/SIO.A asserts DEVSEL#.

Since an active MEMCS# will always result in an active DEVSEL# at the "Slow" sample point, MEMCS# is used as an early indication of DEVSEL#. In this case, if the device using MEMCS# is the only "slow" agent in the system, the sample point can be moved in to the "typical" edge.

Unclaimed PCI cycles with memory addresses above 16M and I/O addresses above 64K are not forwarded to the ISA Bus. To avoid the possibility of aliasing, the SIO/SIO.A does not respond with DEVSEL# (BIOS accesses are an exception to this).

4.1.5. UTILITY BUS ENCODED CHIP SELECTS

The SIO/SIO.A generates encoded chip selects for certain functions that are located on the utility bus (formerly X-Bus). The encoded chip selects are generated combinatorially from the ISA SA[15:0] address bus. Chip selects can be enabled or disabled via configuration registers. In general, the chip select addresses do not reside in the SIO/SIO.A itself. Write only addresses 70h, 372h, 3F2h are exceptions since particular bits from these registers reside in the SIO/SIO.A. For ISA master cycles, the SIO/SIO.A responds to writes to address 70h, 372h, and 3F2h by generating IOCHRDY and writing to the appropriate bits.

Note that the SIO/SIO.A monitors read accesses to address 60h to support the mouse function. In this case, IOCHRDY is not generated.

4.2. PCI Interface

4.2.1. PCI COMMAND SET

Bus commands indicate to the slave the type of transaction the master is requesting. Bus Commands are encoded on the C/BE[3:0]# lines during the address phase of a PCI cycle.

Table 6. PCI Commands

C/BE[3:0]#	Command Type As Slave	Supported As Slave	Supported As Master
0000	Interrupt Acknowledge	Yes	No
0001	Special Cycle[4]	No/Yes	No
0010	I/O Read	Yes	No
0011	I/O Write	Yes	No
0100	Reserved[3]	No	No
0101	Reserved[3]	No	No
0110	Memory Read	Yes	Yes
0111	Memory Write	Yes	Yes
1000	Reserved[3]	No	No

C/BE[3:0]#	Command Type As Slave	Supported As Slave	Supported As Master
1001	Reserved[3]	No	No
1010	Configuration Read	Yes	No
1011	Configuration Write	Yes	No
1100	Memory Read Multiple	No[2]	No
1101	Reserved[3]	No	No
1110	Memory Read Line	No[2]	No
1111	Memory Write and Invalidate	No[1]	No

NOTES:

1. Treated as Memory Write.
2. Treated as Memory Read.
3. Reserved Cycles are considered invalid by the SIO/SIO.A and are to be competely ignored. All internal address decoding is ignored and DEVSEL# is never to be asserted.
4. Special Cycles are considered invalid by the 82378IB and are completely ignored. The 82378ZB responds to a Stop Grant Special Cycle.

NOTE

During PCI reads of the ISA Bus the PCI AD signals can change asynchrounously. The system design should be careful to avoid any cross coupled noise that may be generated from the PCI AD signals onto the PCI control signals. Otherwise, system failures may occur.

4.2.2. TRANSACTION TERMINATION

The SIO/SIO.A supports both Master-initiated Termination as well as Target-initiated Termination. Two forms of master-initiated termination are supported—Normal Termination of a completed transaction and abnormal termation due to no slave response to the transaction (Abort). The SIO/SIO.A also supports three forms of Target-initiated Termination—Disconnect, Retry, and Abort.

NOTE

- The SIO/SIO.A always terminates burst cycles with a disconnect protocol.
- Except for accesses to the internal BIOS Timer Register, the SIO/SIO.A issues a target-abort when the internal SIO/SIO.A registers are the target of a PCI master I/O cycle and more than one byte enable is active.

The following lists the SIO/SIO.A response as a master to a target-termination:

1. For a target-abort, the SIO/SIO.A will not retry the cycle. If an ISA master or the DMA is waiting for the PCI cycle to complete (CHRDY negated), the target-abort condition will cause the SIO/SIO.A to assert CHRDY and end the cycle on the ISA Bus. If the ISA master or DMA device was reading from PCI memory, the SIO/SIO.A will drive all 1's on the data lines of the ISA Bus. The Received Target-abort Status bit in the PCI Status Register will be set indicating that the SIO/SIO.A experienced a target-abort condition.
2. If the SIO/SIO.A is retried as a master on the PCI Bus, it will remove it's request for 2 PCI clocks before asserting it again to retry the cycle.
3. If the SIO/SIO.A is disconnected as a master on the PCI Bus, it will respond very much as if it had been retried. The difference between retry and disconnect is that the SIO/SIO.A did not see any data phase for the retry.

4.3. PCI Arbitration Controller

The SIO/SIO.A contains a PCI Bus arbiter that supports six PCI masters; the Host Bridge, SIO/SIO.A, and two other masters. The SIO/SIO.A REQ#/GNT# lines are internal. The integrated arbiter can be disabled by asserting CPUREQ# during PCIRST#. When disabled, the SIO/SIO.A REQ#, GNT#, and RESUME# signals become visible for an external arbiter. The internal arbiter is enabled upon power-up.

The internal arbiter contains several features that contribute to system efficiency:

- Use of a RESUME# signal to re-enable a backed-off initiator in order to minimize PCI Bus thrashing when the SIO/SIO.A generates a retry.
- A programmable timer to re-enable retried initiators after a programmable number of PCICLKs.
- The CPU (host bridge) can be optionally parked on the PCI Bus.
- A programmable PCI Bus lock or PCI resource lock function.

The PCI arbiter is also responsible for control of the Guaranteed Access Time (GAT) mode signals.

4.3.1. ARBITRATION SIGNAL PROTOCOL

The internal arbiter follows the PCI arbitration method as outlined in the *Peripheral Component Interconnect (PCI) Specification*. The PCI arbitration priority scheme is programmable through the PCI Arbiter Priority Control and Arbiter Priority Control Extension Register. See the Register Description section for further discussions on arbitration priority.

NOTE

1. The SIO/SIO.A as a master does not generate fast back-to-back accesses.
2. As a target, the SIO/SIO.A does support back-to-back transactions. For back-to-back cycles, the SIO/SIO.A treats positively decoded accesses and subtractively decoded accesses as different targets. Therefore, masters can only run fast back-to-back cycles to positively decoded addresses or to subtractively decoded addresses.
3. Before an ISA master or the DMA can be granted the PCI Bus, it is necessary that all PCI system posted write buffers be flushed (including the SIO/SIO.A Posted Write Buffer). Also, since the ISA originated cycle could access memory on the host bridge, it's possible that the ISA master or the DMA could be held in wait states (via IOCHRDY) waiting for the host bridge arbitration for longer than the 2.5 µs ISA specification. The SIO/SIO.A has an optional mode called the Guaranteed Access Time Mode (GAT) that ensures that this timing specification is not violated.
4. An external arbiter in GAT mode will require special lo gic in the arbiter.

RETRY THRASHING RESOLVE

When a PCI initiator's access is retried, the initiator releases the PCI Bus for a minimum of two PCI clocks and will then normally request the PCI Bus again. To avoid thrashing the bus with retry after retry, the PCI arbiter provides REQ# masking. The REQ# masking mechanism differentiates between SIO/SIO.A target retries and all other retries.

For initiators which were retried by the SIO/SIO.A as a target, the masked REQ# is flagged to be cleared upon RESUME# active. All other retries trigger the Master Retry Timer, if enabled. When the timer expires, the mask is cleared.

The conditions under which the SIO/SIO.A forces a retry to a PCI master and will mask the REQ# are:

1. Any required buffer management

2. ISA Bus occupied by ISA master or DMA
3. The PCI to ISA Posted Write Buffer is full
4. The SIO/SIO.A is locked as a resource and LOCK# is asserted during the address process.

The RESUME# signal is pulsed whenever the SIO/SIO.A has retried a PCI cycle for one of the above reasons and that condition has passed. When RESUME# is asserted, the SIO/SIO.A will unmask the REQ#'s that are masked and flagged to be cleared by RESUME#.

If the internal arbiter is enabled, RESUME# is an internal signal. The RESUME# signal becomes visible as an output when the internal arbiter is disabled. This allows an external arbiter to optionally avoid retry thrashing associated with the SIO/SIO.A as a target. The RESUME# signal is asserted for one PCI clock.

BUS PARKING

The SIO/SIO.A provides PCI Bus parking (enabled via the Arbiter Control Register). Parking is only allowed for the device which is tied to CPUREQ# (typically the system CPU). When bus parking is enabled, CPUGNT# is asserted when no other agent is currently using or requesting the bus. When CPUGNT# is asserted due to bus parking enabled and the PCI Bus idle, the CPU (or the parked agent) must ensure that AD[31:0], C/BE[3:0], and (one PCICLK later) PAR are driven. If bus parking is disabled, the SIO/SIO.A takes responsibility for driving the bus when it is idle.

BUS LOCK MODE

As an option, the SIO/SIO.A arbiter can be configured to run in Bus Lock Mode or Resource Lock Mode. The Bus Lock Mode is used to lock the entire PCI Bus. This may improve performance in some systems that frequently run quick read-modify-write cycles. Bus Lock Mode emulates the LOCK environment found in today's PC by restricting bus ownership when the PCI Bus is locked. With Bus Lock enabled, the arbiter recognizes a LOCK# being driven by any initiator and does not allow any other PCI initiator to be granted the PCI Bus until LOCK# and FRAME# are both negated indicating the master released lock. When Bus Lock is disabled, the default resource lock mechanism is implemented (normal resource lock) and a higher priority PCI initiator could intervene between the read and write cycles and run non-exclusive accesses to any unlocked resource.

4.3.2. INTERNAL/EXTERNAL ARBITER CONFIGURATION

The SIO/SIO.A arbiter is enabled if CPUREQ# is sampled high on the trailing edge of PCIRST#. When enabled, the arbiter is set in fixed priority mode 4 with CPU bus parking turned off. Fixed mode 4 guarantees that the CPU will be able to run accesses to the BIOS in order to configure the system, regardless of the state of the other REQ#'s. Note that the Host Bridge should drive CPUREQ# high during the trailing edge of PCIRST#. When the arbiter is enabled, the SIO/SIO.A acts as the central resource responsible for driving the AD[31:0], C/BE[3:0]#, and PAR signals when no one is granted the PCI Bus and the bus is idle. The SIO/SIO.A is always responsible for driving AD[31:0], C/BE[3:0]#, and PAR when it is granted the bus and as appropriate when it is the master of a transaction. After reset, if the arbiter is enabled, CPUGNT#, GNT0#, GNT1#, and the internal SIOGNT# will be driven based on the arbitration scheme and the asserted REQ#'s.

If an external arbiter is present in the system, the CPUREQ# signal should be tied low. When CPUREQ# is sampled low on the trailing edge of PCIRST#, the internal arbiter is disabled. When the internal arbiter is disabled, the SIO/SIO.A does not drive AD[31:0], C/BE[3:0]#, and PAR as the central resource. In this case, the SIO/SIO.A is only responsible for driving AD[31:0], C/BE[3:0]#, and PAR when it is granted the bus. If the SIO/SIO.A arbiter is disabled, GNT0# becomes SIOREQ#, GNT1# becomes RESUME#, and REQ0# becomes SIOGNT#. This exposes the normally embedded SIO/SIO.A arbitration signals. Note that usage of an external arbiter in GAT mode will require special logic in the arbiter.

4.3.3. Guaranteed Access Time Mode

Guaranteed Access Time (GAT) Mode is enabled/disabled via the PCI Arbiter Control Register. When this mode is enabled, the MEMREQ# and MEMACK# signals are used to guarantee that the ISA 2.5 µs IOCHRDY specification is not violated.

When an ISA master or DMA slave requests the ISA Bus (DREQ# active), the ISA Bus, the PCI Bus, and the memory bus must be arbitrated for and all three must be owned before the ISA master or DMA slave is granted the ISA Bus. After receiving the DREQ# signal from the ISA master or DMA slave, MEMREQ# and FLSHREQ# are asserted (FLSHREQ# is driven active, regardless of GAT mode being enabled or disabled). MEMREQ# is a request for direct access to main memory. MEMREQ# and FLSHREQ# will be asserted as long as the ISA master or the DMA owns the ISA Bus. When MEMACK# is received by the SIO/SIO.A (all posted write buffers are flushed and the memory bus is dedicated to the PCI interface), it will request the PCI Bus. When it is granted the PCI Bus, it asserts the DACK signal releasing the ISA Bus to the requesting master or the DMA.

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

Note that usage of an external arbiter in GAT mode will require special logic in the arbiter.

4.3.3.1. DMA Latencies In GAT Mode Only (82378ZB Only)

In GAT mode, the system may have DREQ# to DACK# latencies of up to 53 msec. Two recommendations for reducing long latencies are:

1. **BIOS Implementation:** Use non-GAT mode. Systems may not meet the 2.1 µs IOCHRDY specification in non-GAT mode if a write back cycle is required. However, Intel has found no issues with non-GAT mode in compatibility testing. Because non-GAT mode allows concurrency on all three buses in the system, (host, PCI and ISA Buses) non-GAT mode is the preferred and higher performance mode of operation for a PCI system.
2. **BIOS and Hardware Implementation:** Use non-GAT mode and connect the FLUSHREQ# pin from the 82378ZB to the FLUSHREQ# and MEMREQ# pins of the host bridge (Neptune, Saturn or Mercury). MEMREQ# on the 82378ZB is unconnected. This avoids the DREQ# to DACK# latency and still meets the ISA 2.1 us IOCHRDY specification.

4.4. ISA Interface

The SIO/SIO.A incorporates a fully ISA Bus compatible master and slave interface. The SIO/SIO.A directly drives six ISA slots without external data or address buffers. The ISA interface also provides byte swap logic, I/O recovery support, wait-state generation, and SYSCLK generation.

The ISA interface supports the following types of cycles:

- PCI-initiated I/O and memory cycles to the ISA Bus.
- For the 82379AB, DMA compatible cycles between PCI memory and ISA I/O and between ISA I/O and ISA memory.
- For the 82378ZB, DMA compatible cycles between PCI memory and ISA I/O and between ISA I/O and ISA memory, DMA type "A", type "B", and type "F" cycles between PCI memory and ISA I/O.
- ISA Refresh cycles initiated by either the SIO/SIO.A or an external ISA master.
- ISA master-initiated memory cycles to the PCI Bus and ISA master-initiated I/O cycles to the internal SIO/SIO.A registers.

An ISA master can access PCI memory, but not I/O devices residing on the PCI Bus. If the SIO/SIO.A is programmed for GAT mode, the SIO/SIO.A arbiter will not grant the ISA Bus before gaining ownership of both the PCI Bus and system memory. However, if the SIO/SIO.A is not programmed in this mode, the SIO/SIO.A does not need to arbitrate for the PCI Bus before granting the ISA Bus to the ISA master.

All cycles forwarded to a PCI resource will run as 16-bit extended cycles (i.e., IOCHRDY will be held inactive until the cycle is completed).

Because the ISA Bus size is different from the PCI Bus size, the data steering logic inside the SIO/SIO.A is responsible for steering the data to the correct byte lanes on both buses, and assembling/disassemblying the data as necessary.

4.4.1. ISA CLOCK GENERATION

The SIO/SIO.A generates the ISA system clock (SYSCLK). SYSCLK is a divided down version of the PCICLK (see Table 9). The clock divisor value is programmed through the ISA Clock Divisor Register.

Table 7. SYSCLK Generation from PCICLK

PCICLK (MHz)	Divisor (Programmable)	SYSCLK (MHz)
25	3	8.33
33	4 (default)	8.33

NOTE:
For PCI frequencies less than 33 MHz (not including 25 MHz), a clock divisor value must be selected that ensures that the ISA Bus frequency does not violate the 6 MHz to 8.33 MHz SYSCLK specification.

NOTE

When the ISA Clock Divisor is programmed for PCICLK/3 (i.e., offset 4Dh, bits 2:0 are set to 001), the DMA line should be disabled. For proper operation, disable the DMA line buffers by setting bit 0 of the PCI Control Register (offset 40h) to 0 when PCICLK/3 is selected. Note that the ISA Master Line Buffer Enable (bit 1) can remain enabled.

4.5. DMA Controller

The 82378ZB/82379AB DMA circuitry incorporates the functionality of two 82C37 DMA controllers with seven independently programmable channels (Channels 0–3 and Channels 5–7). DMA Channel 4 is used to cascade the two controllers and will default to cascade mode in the DMA Channel Mode (DCM) Register. In addition to accepting requests from DMA slaves, the DMA controller also responds to requests that are initiated by software. Software may initiate a DMA service request by setting any bit in the DMA Channel Request Register to a 1. The DMA controller for Channels 0-3 is referred to as "DMA-1" and the controller for Channels 4–7 is referred to as "DMA-2".

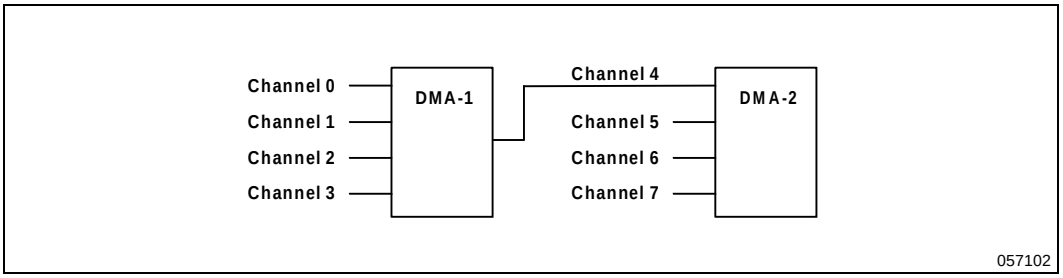


Figure 2. Internal DMA Controller

For the 82378ZB, the DMA supports programmable 8 and 16-bit device sizes per channel using ISA-compatible, Type "A", Type "B", or Type "F" transfer timing. Each DMA channel defaults to the compatible settings for DMA device size: channels [3:0] default to 8-bit, count-by-bytes transfers, and channels [7:5] default to 16-bit, count-by-words (address shifted) transfers. The SIO provides the timing control and data size translation necessary for the DMA transfer between the PCI and the ISA Bus. ISA-compatible is the default transfer timing. Full 32-bit addressing is supported as an extension of the ISA-compatible specification.

For the 82379AB, the DMA supports 8/16-bit device size using ISA-compatible timings. Each DMA channel is hardwired to the compatible settings for DMA device size—channels [3:0] are hardwired to 8-bit, count-by-bytes transfers and channels [7:5] are hardwired to 16-bit, count-by-words (address shifted) transfers. The 82379AB provides the timing control and data size translation necessary for the DMA transfer between the PCI and the ISA Bus. Full 27-bit addressing is supported as an extension of the ISA-compatible specification

For both the 82378ZB and 82379AB, a DMA device (I/O device) is always on the ISA Bus, but the memory referenced is located on either an ISA Bus device or in main memory. For compatible timing mode, the SIO/SIO.A drives the MEMR# or MEMW# strobes if the address is less than 16 Mbytes (00000000–00FFFFFFh). Note that the 82379AB always generates ISA-Compatible DMA memory cycles. The MEMR# and MEMW# memory strobes are generated, regardless of whether the cycle is decoded for PCI or ISA memory. The SMEMR# and SMEMW# is generated if the address is less than 1 Mbyte (00000000–000FFFFFh). To avoid aliasing problems when the address is greater than 16 Mbytes (1000000–7FFFFFFh), the MEMR# or MEMW# strobe is not generated.

For the 82378ZB, if the memory is decoded to be on the ISA Bus, the DMA cycle runs as a compatible cycle. If the memory is decoded to be on the PCI Bus, the cycle can run as compatible, "A", "B", or "F" type. The ISA controller does not drive a valid address for type "A", "B", and "F" DMA transfers on the ISA Bus. For type "A", "B", and "F" timing mode DMA cycles, the SIO only generates the MEMR# or MEMW# strobe when the address is decoded for ISA memory. When this occurs, the cycle converts to compatible mode timing.

For both the 82378ZB and 82379AB, the channels can be programmed for any of four transfer modes—single, block, demand, or cascade. Each of the three active transfer modes (single, block, and demand), can perform three different types of transfers (read, write, or verify). Note that memory-to-memory transfers are not supported by the 82379AB. The DMA supports fixed and rotating channel priorities. The DMA controller also features refresh address generation, and auto-initialization following a DMA termination.

4.5.1. DMA TIMINGS

ISA-Compatible timing is provided for DMA slave devices. For the 82378ZB, three additional timings are provided for I/O slaves capable of running at faster speeds. These timings are referred to as Type "A", Type "B", and Type "F".

4.5.1.1. Compatible Timing (82378ZB and 82379AB)

Compatible timing runs at 8 SYSCLKs during the repeated portion of a Block or Demand mode transfer.

4.5.1.2. Type "A" Timing (82378ZB)

Type "A" timing is provided to allow shorter cycles to PCI memory. Type "A" timing runs at 6 SYSCLKs (720 ns/cycle) during the repeated portion of a block or demand mode transfer. This timing assumes an 8.33 MHz SYSCLK. Type "A" timing varies from compatible timing primarily in shortening the memory operation to the minimum allowed by system memory. The I/O portion of the cycle (data setup on write, I/O read access time) is the same as with compatible cycles. The actual active command time is shorter, but it is expected that the DMA devices which provide the data access time or write data setup time should not require excess IOR# or IOW# command active time. Because of this, most ISA DMA devices should be able to use type "A" timing.

4.5.1.3. Type "B" Timing (82378ZB)

Type "B" timing is provided for 8/16-bit ISA DMA devices which can accept faster I/O timing. Type "B" only works with PCI memory. Type "B" timing runs at 5 SYSCLKs (600 ns/cycle) during the repeated portion of a Block or Demand mode transfer. This timing assumes an 8.33 MHz SYSCLK. Type "B" timing requires faster DMA slave devices than compatible timing in that the cycles are shortened so that the data setup time on I/O write cycles is shortened and the I/O read access time is required to be faster. Some of the current ISA devices should be able to support type "B" timing, but these will probably be more recent designs using relatively fast technology.

4.5.1.4. Type "F" Timing (82378ZB)

Type "F" timing provides high performance DMA transfer capability. These transfers are mainly for fast I/O devices (i.e., IDE devices). Type "F" timing runs at 3 SYSCLKs (360 ns/cycle) during the repeated portion of a Block or Demand mode transfer.

4.5.1.5. DREQ And DACK# Latency Control (82378ZB and 82379AB)

The SIO/SIO.A DMA arbiter maintains a minimum DREQ to DACK# latency on DMA channels programmed to operate in compatible timing mode. This is to support older devices such as the 8272A. The DREQs are delayed by eight SYSCLKs prior to being seen by the arbiter logic. Software requests will not have this minimum request to DACK# latency.

4.5.2. ISA REFRESH CYCLES (82378ZB and 82379AB)

Refresh cycles are generated by two sources: the refresh controller inside the SIO/SIO.A component or by ISA Bus masters other than the SIO/SIO.A. The ISA Bus controller will enable the address lines SA[15:0] so that when MEMR# goes active, the entire ISA system memory is refreshed at one time. Memory slaves on the ISA Bus must not drive any data onto the data bus during the refresh cycle.

4.5.3. SCATTER/GATHER (S/G) DESCRIPTION (82378ZB)

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

The S/G Command and Status Registers are used to control the operational aspect of S/G transfers. The SGD Table Pointer Register holds the address of the next buffer descriptor in the SGD Table.

The next buffer descriptor is fetched from the SGD Table by a DMA read transfer. DACK# will not be asserted for this transfer because the IO device is the SIO itself. The SIO will fetch the next buffer descriptor from either PCI memory or ISA memory, depending on where the SGD Table is located. If the SGD table is located in PCI memory, the memory read will use the line buffer to temporarily store the PCI read before loading it into the DMA S/G registers. The line buffer mode (8-byte or single transaction) for the S/G fetch operation will be the same as what is set for all DMA operations. If set in 8-byte mode, the SGD Table fetches will be PCI burst memory reads. The SGD Table PCI cycle fetches are subject to all types of PCI cycle termination (retry, disconnect, target-abort, master-abort). The fetched SGD Table data is subject to normal line buffer coherency management and invalidation. EOP will be asserted at the end of the complete link transfer.

To initiate a typical DMA S/G transfer between memory and an I/O device, the following steps are required:

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

   Each S/G Descriptor for the linked list contains the following information:

   a. Memory Address (buffer start) 4 bytes
   b. Transfer Size (buffer size) 2 bytes
   c. End of Link List 1 bit (MSB)

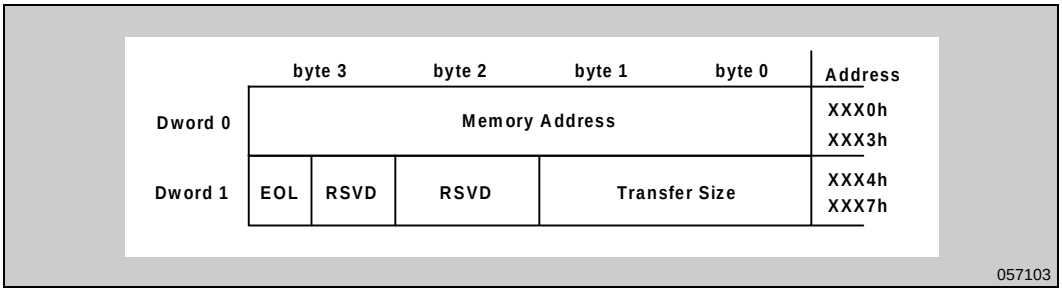


Figure 3. SGD Format

2. Initialize the DMA Channel Mode and DMA Channel Extended Mode Registers with transfer specific information like 8/16-bit I/O device, Transfer Mode, Transfer Type, etc.
3. Software provides the starting address of the SGD Table by loading the SGD Table Pointer Register.
4. Engage the S/G function by writing a Start command to the S/G Command Register.
5. The Mask register should be cleared as the last step of programming the DMA register set. This is to prevent the DMA from starting a transfer with a partially loaded command description.
6. Once the register set is loaded and the channel is unmasked, the DMA will generate an internal request to fetch the first buffer from the SGD Table.

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

The DMA controller will terminate a S/G cycle by detecting an End of List (EOL) bit in the SGD Table. After the EOL bit is detected, the channel transfers the buffers in the Base and Current register sets, if they are loaded. At terminal count the channel asserts EOP or IRQ13, depending on its programming and set the terminate bit in the S/G Status Register. If the channel asserted IRQ13, then the appropriate bit is set in the S/G Interrupt Status Register. The active bit in the S/G Status Register will be reset and the channel's Mask bit will be set.

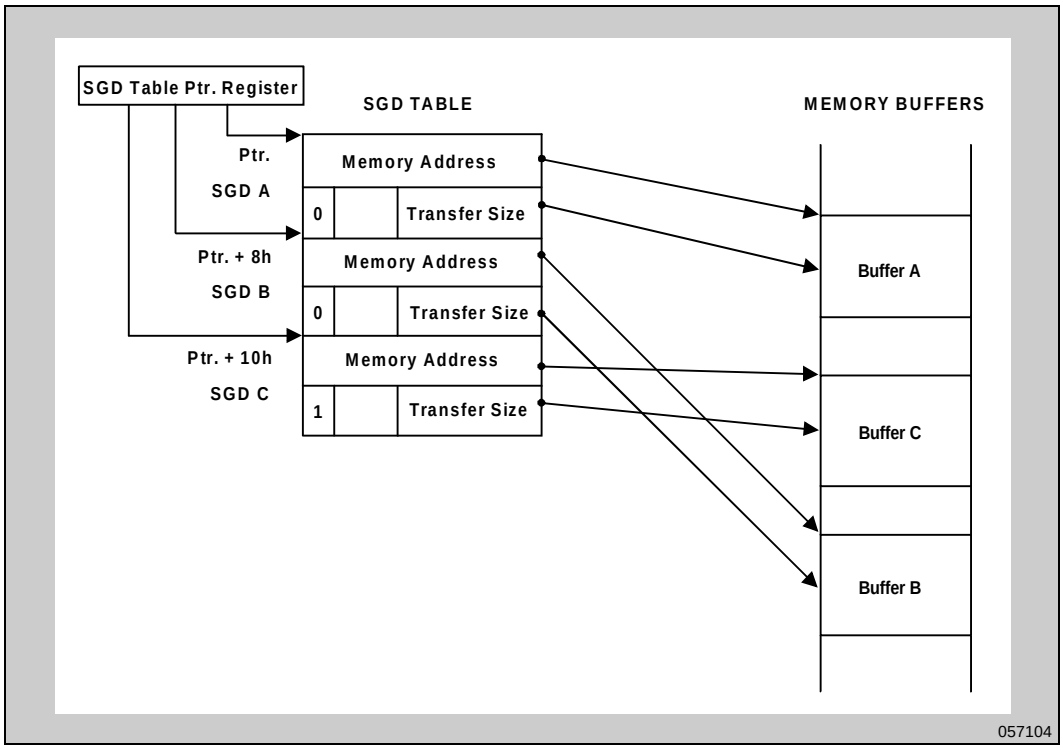


Figure 4. Link List Example

4.6. Data Buffering

The SIO/SIO.A contains data buffers to isolate the PCI Bus from the ISA Bus. The buffering is described from two perspectives: PCI master accesses to the ISA Bus (Posted Write Buffer) and DMA/ISA master accesses to the PCI Bus (Line Buffer). Temporarily buffering the data requires buffer management logic to ensure that the data buffers remain coherent.

4.6.1. DMA/ISA MASTER LINE BUFFER

An 8-byte Line Buffer is used to isolate the ISA Bus's slower I/O devices from the PCI Bus. The Line Buffer is bi-directional and is used by ISA masters and the DMA controller to assemble and disassemble data. Only memory data written to or read from the PCI Bus by an ISA master or DMA is assembled/disassembled using this 8-byte line buffer. I/O cycles do not use the buffer.

Bits 0 and 1 of the PCI Control Register set the buffer to operate in either single transaction mode (bit=0) or 8-byte mode (bit=1). Note that ISA masters and DMA controllers can have their buffer modes configured separately.

In single transaction mode, the buffer will store only one transaction. For DMA/ISA master writes, this single transaction buffer looks like a posted write buffer. As soon as the ISA cycle is complete, a PCI cycle is scheduled. Subsequent DMA/ISA master writes are held off in wait-states until the buffer is empty. For DMA/ISA master reads, only the data requested is read over the PCI Bus. For instance, if the DMA channel is programmed in 16-bit mode, 16 bits of data will be read from PCI. As soon as the requested data is valid on the PCI Bus, it is latched into the Line Buffer and the ISA cycle is then completed, as timing allows. Single transaction mode will guarantee strong read and write ordering through the buffers.

In 8-byte mode, for write data assembly, the Line Buffer acts as two individual 4 byte buffers working in ping pong fashion. For read data disassembly, the Line Buffer acts as one 8-byte buffer.

4.6.2. PCI MASTER POSTED WRITE BUFFER

PCI master memory write cycles destined to ISA memory are buffered in a 32-bit Posted Write Buffer. The PCI Memory Write and Memory Write and Invalidate commands are all treated as a memory write and can be posted, subject to the Posted Write Buffer status. The Posted Write Buffer has an address associated with it. A PCI master memory write can be posted any time the posted write buffer is empty and write posting is enabled (bit 2 of the PCI Control Configuration Register is set to a 1). Also, the ISA Bus must not be occupied. If the posted write buffer contains data, the PCI master write cycle is retried. If the posted write buffer is disabled, the SIO/SIO.A response to a PCI master memory write is dependent on the state of the ISA Bus. If the ISA Bus is available and the posted write buffer is disabled, the cycle will immediately be forwarded to the ISA Bus (TRDY# will not be asserted until the ISA cycle has completed). If the ISA Bus is busy and the posted write buffer is disabled, the cycle is retried. Memory read and I/O read and I/O write cycles do not use the 32-bit Posted Write Buffer.

4.7. SIO Timers

4.7.1. INTERVAL TIMERS

The SIO/SIO.A contains three counters that are equivalent to those found in the 82C54 programmable interval timer. The three counters are contained in one SIO/SIO.A timer unit, referred to as Timer-1. Each counter output provides a key system function. Counter 0 is connected to interrupt controller IRQ0 and provides a system timer interrupt for a time-of-day, diskette time-out, or other system timing functions. Counter 1 generates a refresh request signal and Counter 2 generates the tone for the speaker. Note that the 14.31818 MHz counters use OSC for a clock source.

Counter 0, System Timer: This counter functions as the system timer by controlling the state of IRQ0 and is typically programmed for Mode 3 operation. The counter produces a square wave with a period equal to the product of the counter period (838 ns) and the initial count value. The counter loads the initial count value one counter period after software writes the count value to the counter I/O address. The counter initially asserts IRQ0 and decrements the count value by two each counter period. The counter negates IRQ0 when the count value reaches 0. It then reloads the initial count value and again decrements the initial count value by two each counter period. The counter then asserts IRQ0 when the count value reaches 0, reloads the initial count value, and repeats the cycle, alternately asserting and negating IRQ0.

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

Counter 2, Speaker Tone: This counter provides the speaker tone and is typically programmed for Mode 3 operation. The counter provides a speaker frequency equal to the counter clock frequency (1.193 MHz) divided by the initial count value. The speaker must be enabled by a write to Port 061h (see Register Description section).

NOTE

1. In the PC-AT architecture, the three interval timers in the 82C54 perform the following functions— Timer 1 is the System timer, Timer 2 provides the refresh request, and Timer 3 is used for the speaker tone. The interval timer in the SIO/SIO.A, are intended to be used for these functions. Other operations are not supported and may produce unintended operations. In other than these operations, the timer may be in a state where a counter can be read at the same time it's count is changing and an incorrect value is latched. The readback and latch commands are also affected.
2. Hardware re-triggerable one-shot mode (timer mode 1) for timer 2. This mode uses a rising edge on the timer GATE input to start the timer. The timer GATE input can be toggled by writing to I/O port 61H. Because the GATE pin is asserted relative to PCICLK, the setup and hold time requirements of the GATE pin relative to the 82C54 OSC clock can be violated. Because GATE is edge triggered in timer mode 1, the 82C54 does not recognize the rising edge and the timer does not begin counting. In the PC-AT architecture, timer 2 is the only timer that has its GATE input connected to an I/O port. All the other GATE inputs are tied high. This condition was uncovered during 82C54 diagnostic testing. No application failures have been reported due to this issue.

4.7.2. BIOS TIMER

The SIO/SIO.A provides a system BIOS Timer that decrements at each edge of its 1.04 MHz clock (derived by dividing the 8.33 MHz SYSCLK by 8). Since the state of the counter is undefined at power-up, it must be programmed before it can be used. Accesses to the BIOS Timer are enabled and disabled through the BIOS Timer Base Address Register. The timer continues to count even if accesses are disabled.

A BIOS Timer Register is provided to start the timer counter by writing an initial clock value. The BIOS Timer Register can be accessed as a single 16-bit I/O port or as a 32-bit port with the upper 16-bits being "don't care" (reserved). It is up to the software to access the I/O Register in the most convenient way. The I/O address of the BIOS Timer Register is software relocatable. The I/O address is determined by the value programmed into the BIOS Timer Base Address Register.

The BIOS Timer clock has a value of 1.04 MHz using an 8.33 MHz SYSCLK input (an 8-to-1 ratio will always exist between SYSCLK and the timer clock). This allows the counting of time intervals from 0 to approximately 65 ms. Because of the PCI clock rate, it is possible to start the counter and read the value back in less than 1 µs. The expected value of the expired interval is 0, but depending on the state of the internal clock divisor, the BIOS Timer might indicate that 1 ms has expired. Therefore, accuracy of the counter is ± 1 µs.

A write operation to the BIOS Timer Register will initiate the counting sequence. The timer can be initiated by writing either the 16-bit data portion or the whole 32-bit register (upper 16 bits are "don't care"). After initialization, the BIOS timer will start decrementing until it reaches zero. Then it will stop decrementing (and hold a zero value) until initialized again.

After the timer is initialized, the current value can be read at any time and the timer can be reprogrammed (new initial value written), even before it reaches zero.

All write and read operations to the BIOS timer Register should include all 16 counter bits. Separate accesses to the individual bytes of the counter must be avoided since this can cause unexpected results (wrong count intervals).

4.8. Interrupt Controller

The SIO/SIO.A provides an ISA-compatible interrupt controller which incorporates the functionality of two 82C59 interrupt controllers. The two controllers are cascaded so that 14 external and two internal interrupts are possible. The master interrupt controller provides IRQ[7:0] and the slave interrupt controller provides IRQ[15:8] (see Figure 5). The two internal interrupts are used for internal functions only and are not available to the user. IRQ2 is used to cascade the two controllers together and IRQ0 is used as a system timer interrupt and is tied to Interval Timer 1, Counter 0. The remaining 14 interrupt lines (IRQ1, IRQ[15:3]) are available for external system interrupts. Edge or level sense selection is programmable on a by-controller basis.

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

Note that IRQ13 is generated internally (as part of the coprocessor error support) by the SIO/SIO.A when bit 5 in the ISA Clock Divisor Register is set to a 1. When this bit is set to a 0, then the FERR#/IRQ13 signal is used as an external IRQ13 signal and has the same functionality as the normal IRQ13 signal. IRQ12/M is generated internally (as part of the mouse support) by the SIO/SIO.A when bit 4 in the ISA Clock Divisor Register is set to a 1. When set to a 0, the standard IRQ12 function is provided.

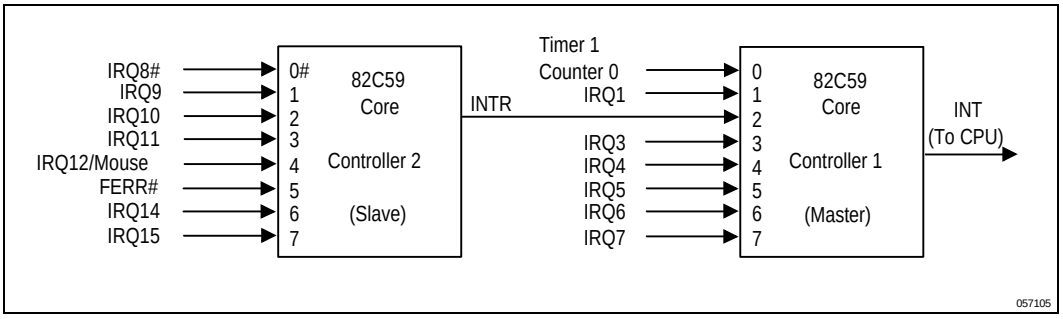


Figure 5. Block Diagram of the Interrupt Controller

4.8.1. 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[7:3,1:0]#) and 04D1h (for the Slave Bank, IRQ[15:8])#. They allow the edge and level sense selection to be made on an interrupt by interrupt basis instead of on a complete bank. Interrupts reserved for ISA use MUST be programmed for edge sensitivity (to ensure ISA compatibility). That is, IRQ (0,1,2,8#,13) must be programmed for edge sensitive operation. The LTIM bit (Edge/Level Bank select, offsets 20h, A0h) is disabled in the SIO/SIO.A. 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.

4.8.2. NON-MASKABLE INTERRUPT (NMI)

An NMI is an interrupt requiring immediate attention and has priority over the normal interrupt lines (IRQx). The SIO/SIO.A indicates error conditions by generating a non-maskable interrupt. NMI interrupts are caused by:

1. System Errors on the PCI Bus. SERR# is driven low by a PCI resource when this error occurs.
2. Parity errors on the add-in memory boards on the ISA expansion bus. IOCHK# is driven low when this error occurs.

The NMI logic incorporates two different 8-bit registers—the NMI Status and Control Register and the NMI Enable and Real-Time Clock Address Register. These registers are described in the Register Description Section.

All NMI sources can be enabled or disabled by setting Port 070h bit 7 to a 0 or 1. This disable function does not clear the NMI detect flip-flops. This means, if NMI is disabled then enabled via Port 070h, then an NMI will occur when Port 070h is re-enabled if one of the NMI detect flip-flops had been previously set.

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

1. NMI is detected by the processor on the rising edge of the NMI input.
2. The processor will read the status stored in Port 061h to determine what sources caused the NMI. The processor may then set to 0 the register bits controlling the sources that it has determined to be active. Between the time the processor reads the NMI sources and sets them to a 0, an NMI may have been generated by another source. The level of NMI will then remain active. This new NMI source will not be recognized by the processor because there was no edge on NMI.

3. The processor must then disable all NMI's by setting bit 7 of Port 070H to a 1 and then enable all NMI's by setting bit 7 of Port 070H to a 0. This will cause the NMI output to transition low then high if there are any pending NMI sources. The CPU's NMI input logic will then register a new NMI.

4.9. Advanced Programmable Interrupt Controller (APIC) (82379AB Only)

In addition to the standard ISA compatible interrupt controller described in the previous section, the 82379AB 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 ISA 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 6)—one residing in the I/O subsystem (**I/O APIC**) and the other in the CPU (**Local APIC**). The 82379AB contains an I/O APIC unit. The local APIC and the I/O APIC communicate over a dedicated APIC Bus. The 82379AB's I/O APIC Bus interface consists of two bi-directional data signals (APICD[1:0]) and a clock input (APICCLK).

NOTE

Any designs should strive to make the APICCLK as clean as possible at the input of any CPU. Further details can be found in the latest "PENTIUM PROCESSOR at iCOMP INDEX 735\90 (or 816\100) MHz STEPPING INFORMATION" document, errata 8AP. This document also provides routing and topology guidelines.

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.

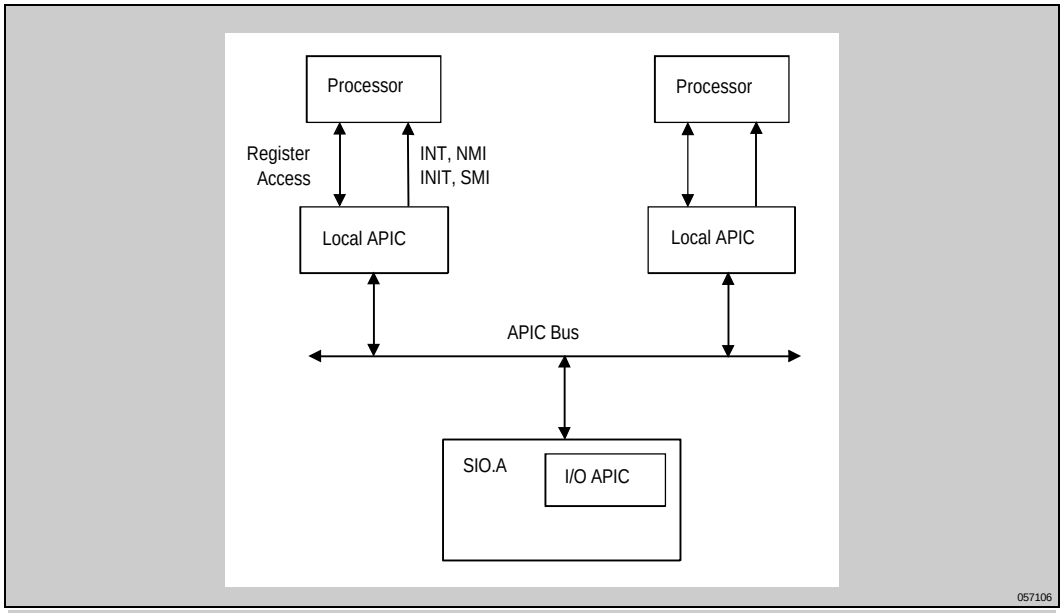


Figure 6. APIC System Structure

The 82379AB 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 7). I/O devices inject interrupts into the system by asserting one of the interrupt lines to the I/O APIC (Figure 8). 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 82379AB 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 the Register Description section. 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 82379AB I/O APIC receives an interrupt request, the 82379AB flushes its buffers and requests all system buffers pointing to PCI to be flushed (via the FLSHREQ#/MEMREQ# signals). The APIC does not send the interrupt message over the APIC Bus until the 82379AB receives confirmation (via the MEMACK# 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.

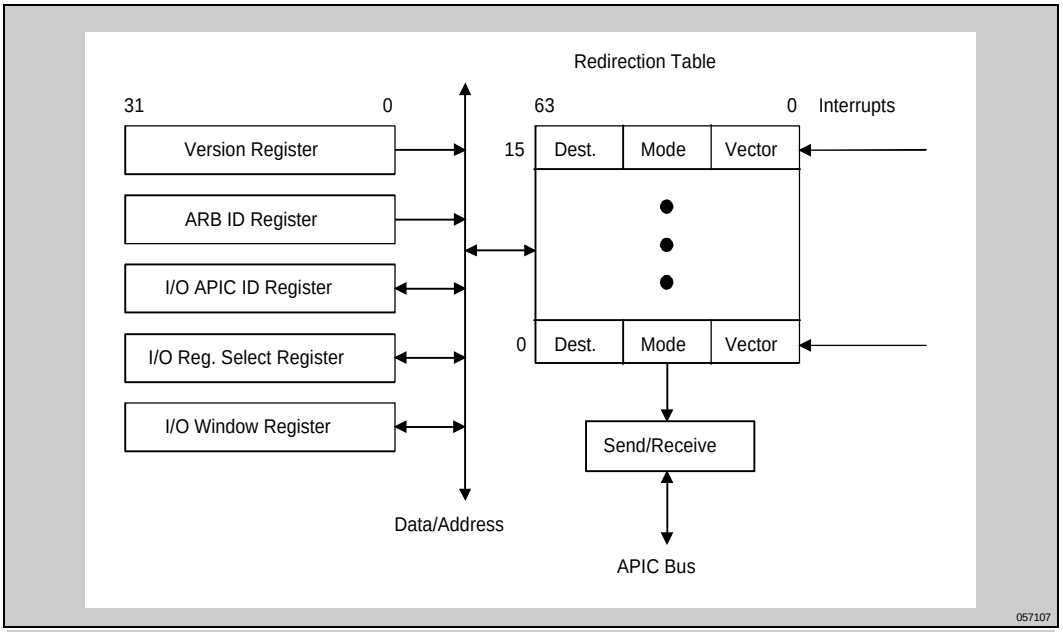


Figure 7. I/O APIC Register Block Diagram

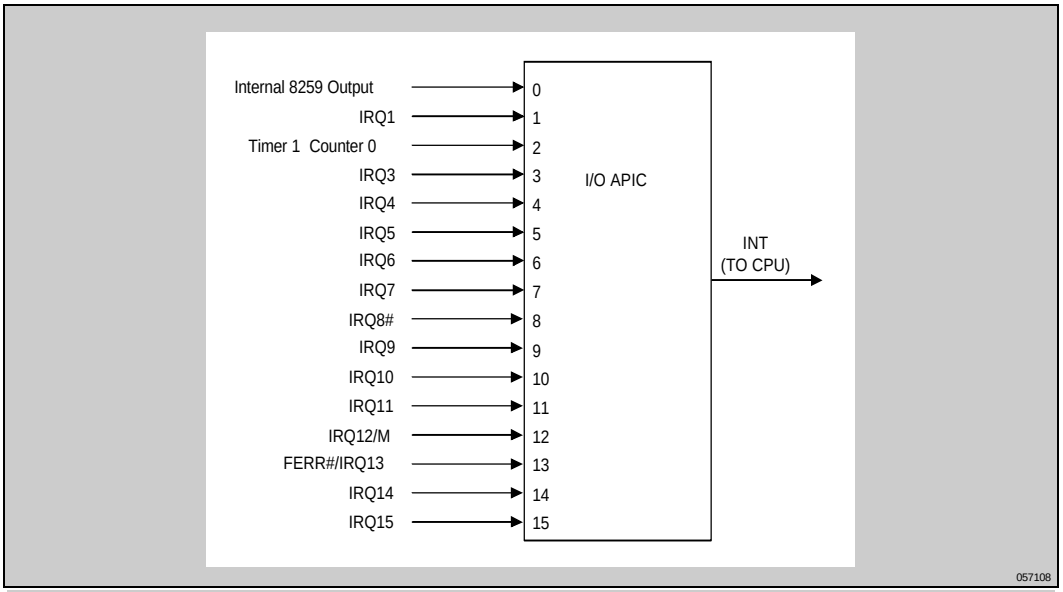


Figure 8. I/O APIC Interrupt Mapping

4.9.1. PHYSICAL CHARACTERISTICS OF APIC BUS (82379AB Only)

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.

4.9.2. ARBITRATION FOR APIC BUS (82379AB Only)

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 7). 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 7. TT_APICBusArbBus Arbitration Cycles

Cycle	Bit 1	Bit 0	
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	

4.9.3. INTR AND THE PENTIUM® PROCESSOR'S "THROUGH LOCAL MODE" (82379AB Only)

The B-1 thru B-5 steppings of the Pentium® processor at ICOMP index 735\90 (or 815\100) have a stepping publication information (10AP) when using the local APIC in Through Local Mode (virtual wire). The Pentium stepping information provides software solutions that permit the customer to use the local APIC in Through Local Mode. This section provides hardware solutions.

If a processor has its Local APIC setup in Through Local Mode (virtual wire), the 82379AB can potentially negate and then reassert INT before the interrupt acknowledge cycle for the first interrupt completes. When this occurs,

the CPU ignores the second interrupt and the system eventually “hangs” because no further interrupts are serviced.

The PCMC can wait 106 host clocks during an interrupt acknowledge cycle between FRAME# assertion on the PCI Bus and BRDY# being asserted on the host bus. Since the SIO.A can reassert INT as soon as 18 host clocks after receiving FRAME# of the interrupt acknowledge cycle, there is a “window” of 88 host clocks where INT can be reasserted by the SIO.A.

One solution uses an external circuit where LOCK# and M/IO# from the processor blocks the assertion of INT until the interrupt acknowledge cycle has completed. Loading and timing on the LOCK# and M/IO must be considered for this solution as they are host bus signals. The following is the logic equation used for INT:

INT_OUT = ((INT * INT_OUT + (INT*M/IO#) + (INT*LOCK#))

Another solution delays the assertion of INT by 88 hostclocks (1320 ns). An RC circuit on the output of the required external buffer on the INT signal can also accomplish this delay.

Any solution that prevents INT from being asserted for 106 host clocks after FRAME# goes low at the start of an interrupt acknowledge cycle will work. As an example, another way to implement the delay is to block the re-assertion of INT for 14 BCLKs after INT goes low. The logic uses INT going low as the start of the “block clock” as this transition signals the beginning of the blocking period. INT can, therefore, be directly passed through the circuit with no appreciable delay at all other times.

14 BCLKS was chosen for the following reasons :

FRAME# to BRDY# = 106 HOST CLOCKS
FRAME# to INT low (SIO) = 2 PCICLKs (4 HOST CLOCKS)
BLOCK needed from INT low to BRDY# = 104 HOST CLOCKS
= 13 BCLKS (chosen to minimize flip flops)
= 13 + 1 (needed due to asynchronous locks)
= 14 BCLKS

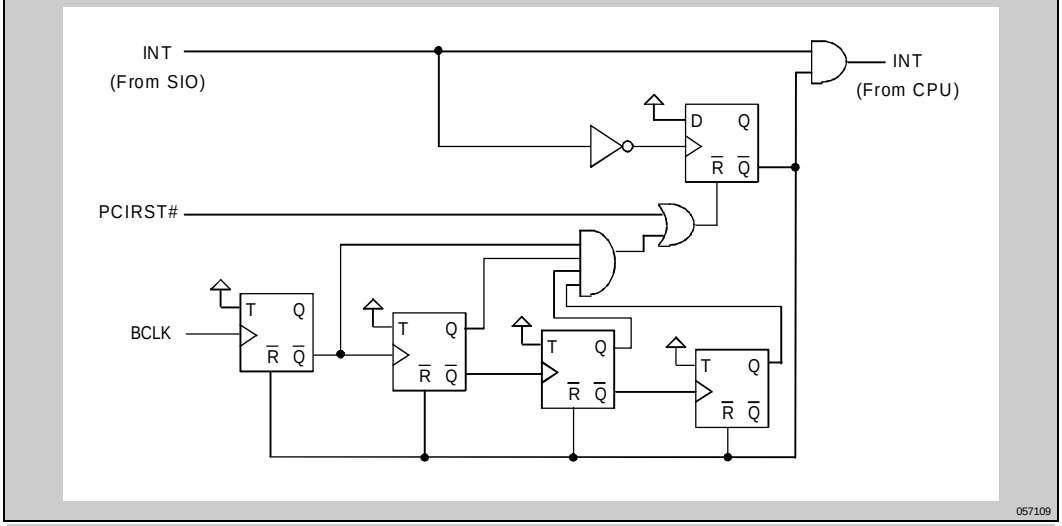


Figure 9. INTR - Through Local Mode Hardware Solution

4.9.4. PULSING OF APICD1 DURING CPU RESET (82379AB Only)

For dual-processing (DP) ISA systems using the Pentium® processor at ICOMP index 735\90 (or 815\100) and the A-1 stepping of the 82434NX PCMC host bridge an anomily can occur when the IOAPIC data line is pulsed during CPU reset. (Single processor designs are not affected. Also, Pentium Pro processor systems are not affected.) To avoid this issue, use one of the following workarounds:

The CPUs APIC data lines (APICD[0,1]), have secondary functions when CPU RESET (CPURST) is active. APICD1 is the APICEN (APIC enable) pin and APICD0 is the DPEN# (dual processor enable) pin. The local APICs are enabled if APICEN is sampled active high on the falling edge of CPURST. Also, during RESET, the second CPU drives DPEN# active low to indicate to the boot processor that it is present.

On the A-1 stepping of the 82434NX PCMC, there is approximately a 1-ms delay (specifically 66,672 external host clocks) between the negation of PCIRST# and the negation of CPURST following a cold system boot (i.e., the PWROK input of the PCMC transitions from low to high). During this 1-ms window, the I/O APIC in the SIO.A, which is no longer in reset, begins to monitor the APIC Bus for valid messages. APICEN (APICD1 = logic 1) and DPEN# (APICD0 = logic 0) remain asserted during this period. The I/O APIC interprets the states of these two bits as the beginning of a normal APIC message and continues to sample the APIC data lines. The I/O APIC responds, per the APIC protocol, by driving APICD1(APICEN) low once every 20 APICCLKs, indicating a checksum error has been detected on the current message. Under these circumstances, this is proper operation for the SIO.A.

From the time that PCIRST# negates until the time that CPURST negates, the I/O APIC periodically pulses APICD1 low for one out of every 20 APICCLKs. If the pulse on APICD1 (APICEN) aligns with the falling edge of CPU RESET (CPURST), the local APICs will not be enabled; this prevents proper DP or DP-ready functionality (Figure 10}.

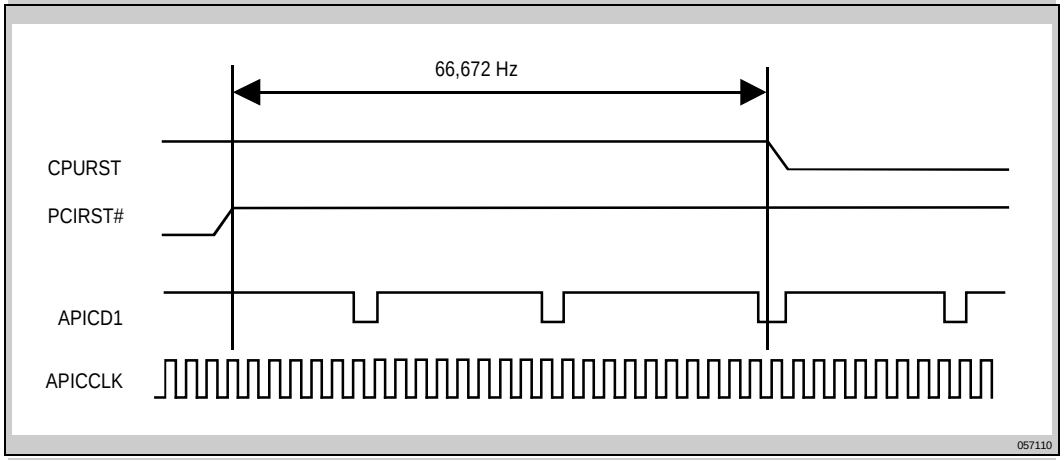


Figure 10. RESET/APIC Data Signl Timing (Not To Scale)

Hardware Solution #1:

Calculate the window for the APICD1 pulse relative to CPURST for a given host clock (HCLK) and APIC Clock (APICCLK) frequency. The following example demonstrates that by using a 60.0 MHz HCLK and a 15.0 MHz APICCLK (0.01% accuracy for both) the APICD1 pulse will be outside the sampling window of CPURST. With these equations, equally acceptable windows can be found for other values of HCLK and APICCLK.

Note that with the above frequency accuracies, a 15.0 MHz APICCLK with a 50-MHz or 66-MHz HCLK may not always function correctly under worst-case clock variations in a DP ISA environment. Also, a 16.0-MHz APICCLK may not function correction with 50MHz, 60MHz or 66MHz under worst-case clock variations.

Sample Calculation:

Two key variables in the system first must be identified:

1. The accuracy of the host clock oscillator input to the PCMC (HCLKOSC) additional inaccuracy in HCLK or PCI Clock (PCLK) is not induced by the internal PCMC PLLs.
2. The accuracy of the APIC clock oscillator.

The following example assumes a 60-MHz Host Clock (HCLK) frequency with an oscillator accuracy of 0.01% and a 15.0-MHz APIC Clock (APICCLK) frequency with an accuracy of 0.01%.

STEP 1: Determine the nominal clock frequency for HCLK, PCLK and APICCLK. Also, determine the accuracy for HCLK and APICCLK. Since PCLK is HCLK/2, no additional inaccuracy is incurred.

Signal	Frequency	Accuracy
HCLK	60 MHz	0.01%
APICCLK	15 MHz	0.01%

STEP 2: Calculate the maximum and minimum window from PCIRST# and CPURST. One less HCLK (66,671) was used to account for the uncertainty between HCLK and PCLK for the minimum case. This provides the minimum time window between the I/O APIC responding to the APIC messages and the falling edge of CPURST.

Convert these two extremes to absolute time (seconds). When converting the maximum HCLK window to real time be sure to use the max HCLK Period possible due to clock tolerances. Similarly, use the minimum HCLK period for the min HCLK window.

Equation #1:

(HCLK cycles) * Max HCLK Period = Max HCLK Window
(66,672) * (1.66683335E-08) = 1.111311 msec

(HCLK cycles) * Min HCLK Period = Min HCLK Window
(66,671) * (1.66650002E-08) = 1.111072 msec

	APICD1 Pulse Window (s)
Max HCLK window	1.111311E-03
Min HCLK window	1.111072E-03

STEP 3: Determine the equivalent number of APICCLKs contained in the maximum and minimum pulse windows calculated above. Account for the accuracy of the APICCLK source.

Also, since APICCLK and HCLK are completely asynchronous to each other, one APICCLK should be subtracted from the minimum window to account for any HCLK/APICCLK misalignment.

Equation #2:

(Max APICD1 pulse window) * (Max APICCLK frequency) = Max APICCLKs
1.111311E-03 * 15.0015 MHz = 16,671.33 APICCLKs

(Min APICD1 pulse window) * (Min APICCLK frequency) - 1 = Min APICCLKs
1.111072E-03 * 14.9985 MHz - 1 = 16,663.42 APICCLKs

APICCLK Frequency	Max window (APICCLKs)	Min Window (APICCLKs)
15.0 MHz + 0.01%	16671.33	
15.0 MHz - 0.01%		16663.42

STEP 4: Select the maximum and minimum number of APICCLKs contained within the pulse window. The two cases are underlined in the Step 3 calculation above. For each case, divide the number of APICCLKs by 20 to determine the minimum and maximum number of APICD1 pulses driven by the I/O APIC during this window. APICD1 is pulled to a logic 1 via the strong, external pullup resistor for 19 APICCLKs before pulsing low via the I/O APIC for 1 APICCLK.

Signal	Max APICD1 Pulses	Min APICD1 Pulses
APICD1 (APICEN) pulses	833.56	833.17

STEP 5: The fractional portion of each pulse calculation above provides the time (in APICCLKs) between the final APICD1 pulse and the falling edge of CPURST.

To calculate the APICEN setup time for the maximum pulse condition (833.56 pulses), multiply the fractional portion of the pulse calculation (0.56) by 20 and then divide by the maximum APICCLK frequency. This results in the actual setup time between the end of the last APICD1 pulse and the falling edge of CPURST. Subtracting this setup time from [(19 divided by the maximum APICCLK frequency)] results in the APICEN hold time from the falling edge of CPURST.

To calculate the setup and hold times for APICEN for the minimum pulse condition (833.56), the minimum APICCLK frequency is used.

Equation #3 - Maximum APICD1 Pulse Condition

[(fractional number of max APICD1 pulses) * 20]/(max APICCLK frequency) = APICEN setup time
[(0.56) * 20]/(15.0015 MHz) = 747 ns
[19/(max APICCLK frequency)] - APICEN setup time = APICEN hold time

Equation #3 - Minimum APICD1 Pulse Condition

[(fractional number of min APICD1 pulses) * 20]/(min APICCLK frequency) = APICEN setup time
[(0.17) * 20]/(14.9985 MHz) = 227 ns
[19/(min APICCLK frequency)] - APICEN setup time = APICEN hold time
[19/(14.9985 MHz)] - 975 ns = 1040 ns

STEP 6: If the whole number portion of the pulses calculated in Step 4 is identical in the minimum and maximum cases, then the minimum setup and hold time margins for APICEN easily fall out. Therefore, in this example, for this combination of oscillator frequencies and accuracies, worst-case APICEN setup and hold times clearly meet the minimum CPU setup and hold specifications of 2 HCLKs. APICEN will be properly sampled by the CPUs under all circumstances.

If the whole number portions are not equal for the minimum and maximum cases, then the accuracy of the HCLK and APICCLK frequencies is not sufficient to keep the APICD1 pulse from aliasing, i.e., shifting through the entire 20 APICCLK window under worst-case conditions.

Hardware Solution #2

Generate APICCLK by dividing the PCLK by two. In this case, there will not be an anomaly of APICD1 pulsing low during the negation of CPURST, as long as PCLK is generated by dividing HCLK by two and APICCLK is generated by dividing PCLK by two.

This is only viable for designs using the 82379AB. For EISA designs using the 82374EB(SB) ESC, this solution is invalid. The ESC begins pulsing the APICD1 pin later in time, causing the APICD1 pulse to align with the falling edge of CPURST for DP EISA systems.

Hardware Solution #3.

Add logic to ensure that PCIRST# remains asserted low until PWROK is valid and then once PWROK is valid, let PCIRST# become the logical inverse of CPURST. This ensures that the I/O APIC and both CPUs exit reset simultaneously preventing the I/O APIC from responding to phantom APIC messages.

Possible Implementations:

a. **Connect a buffered version of CPURST to an inverter and then connect the output of this inverter to the input** of a three-state buffer. PWROK# is used as the active low enable to the three-state buffer. The output of this three-state buffer is used as the PCIRST# signal for the entire system. Note that the PCMC PCIRST# output pin (pin 147) should be disconnected from the system. It is important to use a buffered version of CPURST and preferably one that already exists in the system due to the critical AC timing of this signal. A weak pulldown resistor (8.2K, for instance) should be added to the PCIRST# net to ensure it is low while PWROK is low.

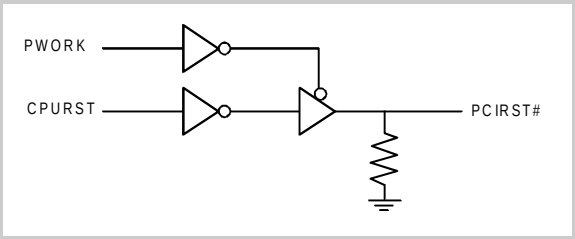


Figure 11. PCIRST# Soln 3a Logic

b. Connect a buffered version of CPURST to an inverter and then connect the output of this inverter to one input of a two-input AND gate. PWROK is the second input to the AND gate. The output of the AND gate becomes PCIRST# for the entire system. Again, the PCMC PCIRST# output pin (pin 147) should be disconnected from the system.

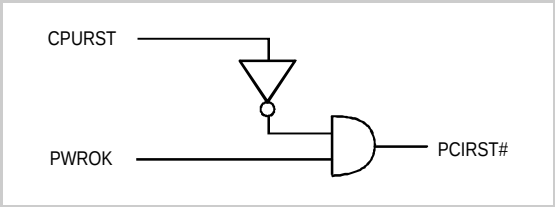


Figure 12. PCIRST# Soln 3b Logic

4.9.5. SIO.A ASSERTING SIGNALS LOW DURING PCIRST# (82379AB ONLY)

The SIO.A drives SMI#, ALT_A20, INT, NMI, IGNNE#, ALT_RST#, and STPCLK# low while PCIRST# is asserted low, and does not drive them high until after PCI reset is released. An anomaly can exist with these seven signals remaining low during and immediately after PCIRST# is negated. The three instances in which this can cause a problem are : during a targeted PCI Reset, and in a 82450GX/KX-P6 system, both during power-up and when BINIT# is asserted on the Pentium Pro processor bus.

Targeted PCI Resets

Systems that support targeted PCI Resets (Resetting the PCI Bus via software control without resetting the microprocessor) may have a problem with some of the seven signals being asserted low during the targeted PCI reset. Since the microprocessor can not know when a PCIRST is occurring, this fix must be incorporated in order to reset the PCI Bus via the register. This will affect all designs using the SIO.A.

The RESET MASK blocking circuit shown above will block signal A from being seen by the CPU during the PCI Reset. The second flip flop is necessary to avoid a glitch on the Z output to the CPU which can happen if signal A is asserted simultaneously with PCIRST#.

The blocking circuitry for all of the signals should be incorporated into a PLD. This will ease loading on PCICLK and PCIRST#. Signals ALT_RST#, IGNNE#, ALT20, STPCLK#, and SMI# should have blocking circuitry.

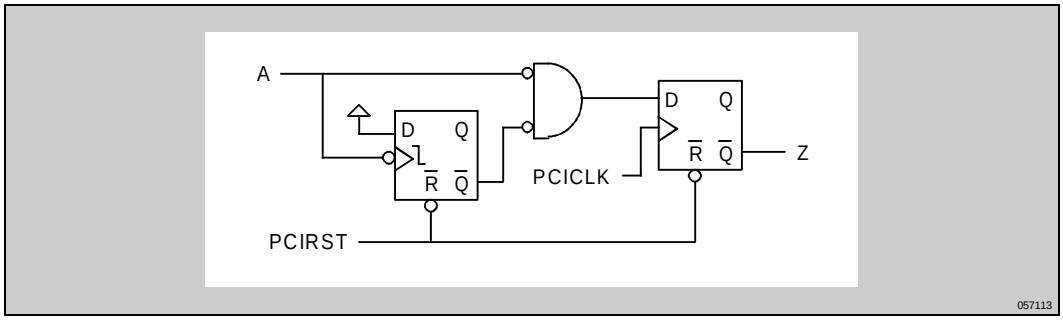


Figure 13. Reset Mask Blocking Logic Using SIO.A

4.10. Utility Bus Peripheral Support

The Utility Bus is a secondary bus buffered from the ISA Bus used to interface with peripheral devices that do not require a high speed interface. The buffer control for the lower 8 data signals is provided by the SIO/SIO.A via two control signals; UBUSOE# and UBUSTR. Figure 14 shows a block diagram of the external logic required as part of the decode and Utility Bus buffer control.

The SIO/SIO.A provides the address decode and three encoded chip selects to support Floppy Controller, Keyboard Controller, Real Time Clock, IDE Drive, 2 Serial Ports (COM1 and COM2), 1 Parallel Port (LPT1, 2, or 3), BIOS Memory, and Configuration Memory (8 Kbyte I/O Mapped). The SIO/SIO.A also supports Floppy DSKCHG Function, Port 92 Function (Alternate A20 and Alternate Reset), and Coprocessor Logic (FERR# and IGNNE# Function)

The binary code formed by the three Encoded Chip Selects determines which Utility Bus device is selected. The SIO/SIO.A also provides an Encoded Chip Select Enable signal (ECSEN#) that is used to select between the two external decoders. A zero selects decoder 1 and a one selects decoder 2. The table below shows the address decode for each of the Utility Bus devices.

Table 8. Encoded Chip Select Summary Table

ECS2	ECS1	ECS0	ECSEN#	Address Decoded	External Chip Select	Note	Cycle Type
Decoder 1							
0	0	0	0	70h, 72h, 74, 76h	RTCALE#		I/O W
0	0	1	0	71h, 73h, 75h, 77h	RTCCS#		I/O R/W
0	1	0	0	60h, 62h, 64h, and 66h (82378ZB) 60h and 64h (82379AB)	KEYBRDCS#		I/O R/W
0	1	1	0	000E0000–000FFFFFh FFFE0000–FFFFFFFFh FFF80000–FFFDFFFFh	BIOSCS#	1	MEM R/W
1	0	0	0	3F0h-3F7h (primary) 370h-377h (secondary)	FLOPPYCS#	2	I/O R/W
1	0	1	0	1F0–1F7h (primary) 170–177h (secondary)	IDECS0#	2	I/O R/W
1	1	0	0	3F6–3F7h (primary) 376–377h (secondary)	IDECS1#	2	I/O R/W
1	1	1	0	Reserved			
Decoder 2							
0	0	0	1	Reserved			
0	0	1	1	0C00h	CPAGECS#	3	I/O R/W
0	1	0	1	0800–08FFh	CFIGMEMCS#	3	I/O R/W
0	1	1	1	3F8–3FFh (COM1) -or- 2F8–2FFh (COM2)	COMACS#	4	I/O R/W
1	0	0	1	3F8–3FFh (COM1) -or- 2F8–2FFh (COM2)	COMBCS#	4	I/O R/W
1	0	1	1	3BC–3BFh (LPT1) 378–37Fh (LPT2) 278–27Fh (LPT3)	LPTCS#	5	I/O R/W
1	1	0	1	Reserved			
1	1	1	1	Idle State			

NOTES:

1. The encoded chip select signals for BIOSCS# will always be generated for accesses to the upper 64 Kbytes at the top of 1 Mbyte (F0000–FFFFFh) and its aliases at the top of the 4 Gbytes and 4 Gbytes - 1 Mbyte. Access to the lower 64 Kbytes (E0000–EFFFFh) and its aliases at the top of 4 Gbytes and 4 Gbytes –1 Mbyte can be enabled or disabled through the SIO/SIO.A. An additional 384 Kbytes of BIOS memory at the top of 4 Gbytes (FFFD0000–FFFDFFFFh) can be enabled for BIOS use.
2. The primary and secondary locations are programmable through the SIO/SIO.A. Only one location range can be enabled at any one time. The floppy and IDE share the same enable and disable bit (i.e., if the floppy is set for primary, the IDE is also set for primary).
3. These signals can be used to select additional configuration RAM.
4. COM1 and COM2 address ranges can be programmed for either Port A (COMACS#) or Port B (COMBCS#).
5. Only one address range (LPT1, LPT2, or LPT3) can be programmed at any one time.

Port 92h Function

The SIO/SIO.A integrates the Port 92h Register. This register provides the alternate reset (ALTRST) and alternate A20 (ALT_A20) functions. Figure 14 shows how these functions are tied into the system.

DSKCHG Function

DSKCHG is tied directly to the DSKCHG signal of the floppy controller. This signal is inverted and driven onto system data line 7 (SD7) during I/O read cycles to floppy address locations 3F7h (primary) or 377 (secondary) as indicated by Table 9.

Table 9. DSKCHG Summary Table

FLOPPYCS# Decode	IDECSx# Decode	State of SD7 (Output)	State of UBUSOE#
Enabled	Enabled	Tri-stated	Enabled
Enabled	Disabled	Driven via DSKCHG	Disabled
Disabled	Enabled	Tri-stated	Enabled[1]
Disabled	Disabled	Tri-stated	Disabled

NOTE:

1. This mode requires external logic to disable the U-Bus transceiver for access to 3F7h/377h. This is necessary due to potential contention between the Utility Bus buffer and a floppy on the ISA Bus driving the system bus at the same time during shared I/O accesses.

Coprocessor Error Support

If bit 5 in the ISA Clock Divisor Register is set to a one, the SIO/SIO.A will support coprocessor error reporting through the FERR#/IRQ13 signal. FERR# is tied directly to the coprocessor error signal of the CPU. If FERR# is driven active in this mode (coprocessor error detected by the CPU), an internal IRQ13 is generated and the INT output from the SIO/SIO.A is driven active. When a write to I/O location F0h is detected, the SIO/SIO.A negates IRQ13 and drives IGNNE# active. IGNNE# remains active until FERR# is driven inactive. Note that IGNNE# is not generated unless FERR# is active.

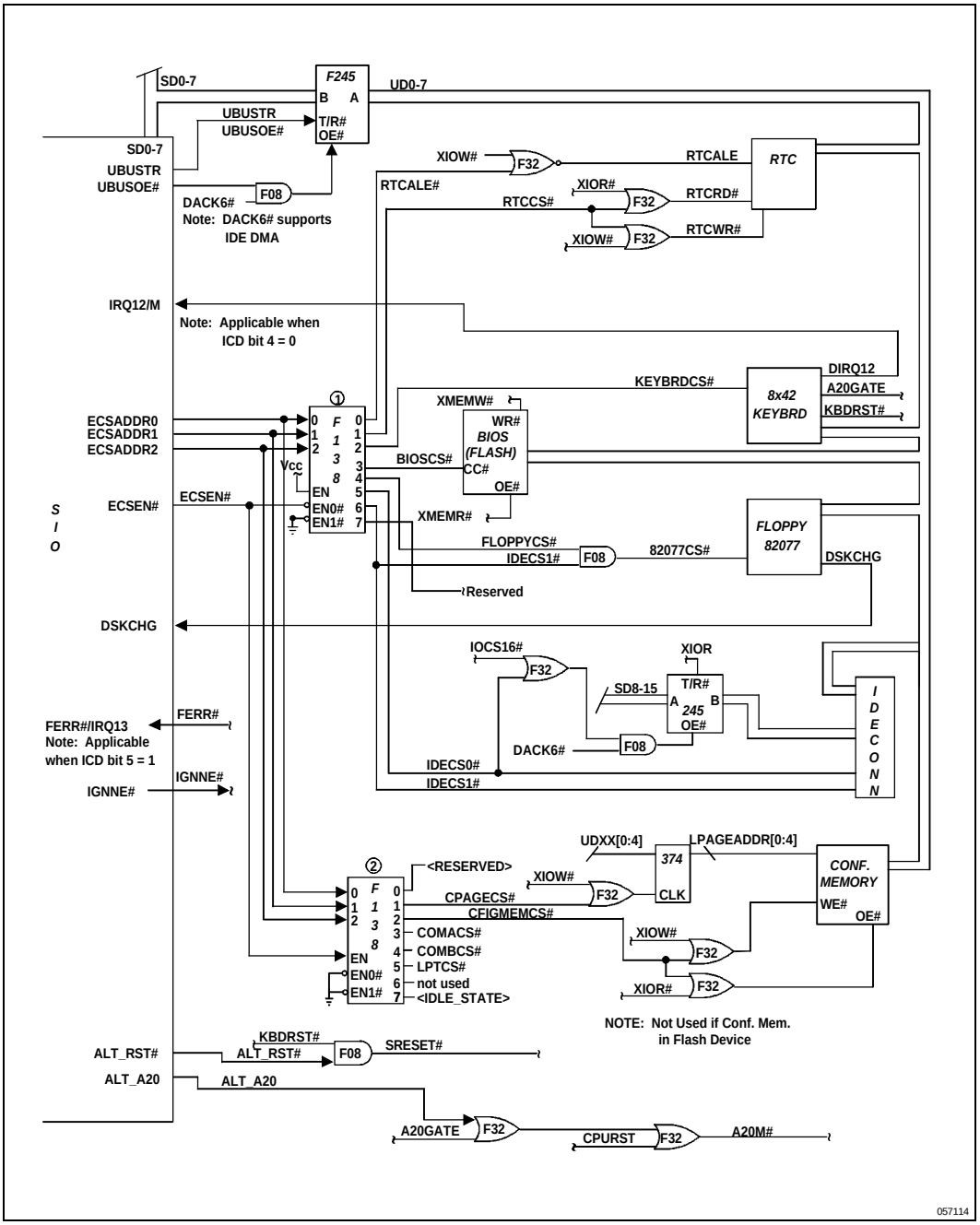


Figure 14. SIO/SIO.A Utility Bus External Support Logic

Utility Bus accesses by the SIO/SIO.A, by an ISA master, and by the DMA are shown in Figure 15 and Figure 16. UBUSOE# and UBUSTR are driven differently for DMA cycles as shown in Figure 16.

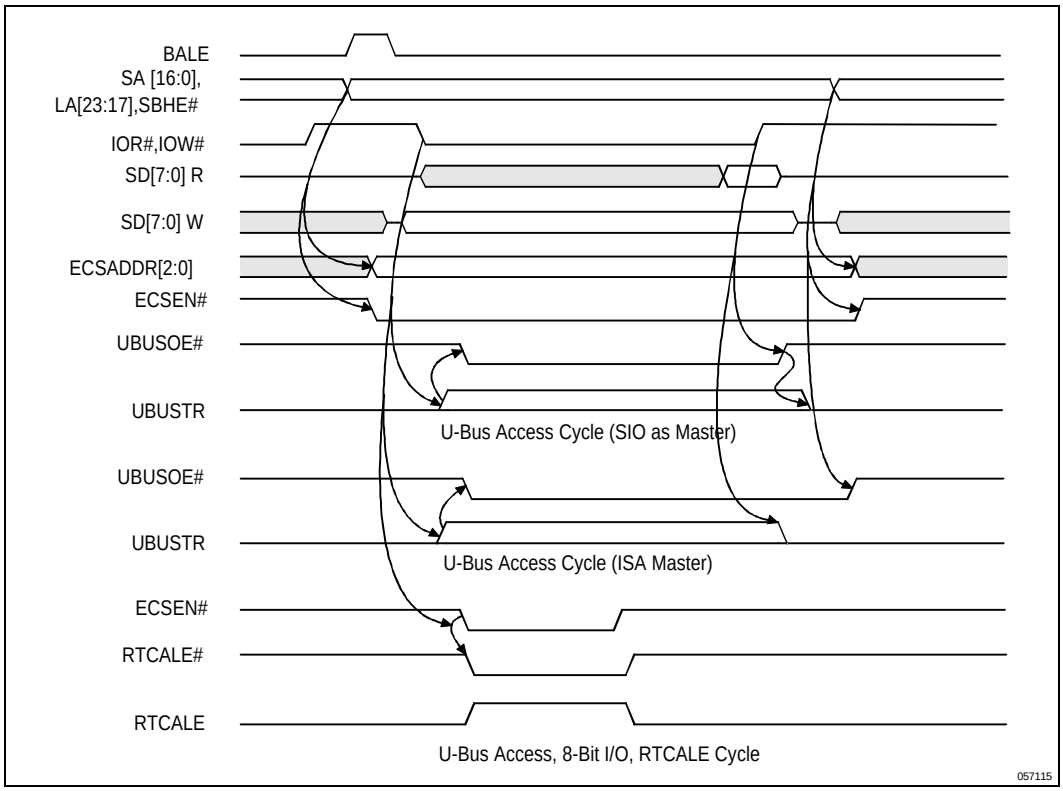


Figure 15. Utility Bus Access (SIO/SIO.A and ISA Master)

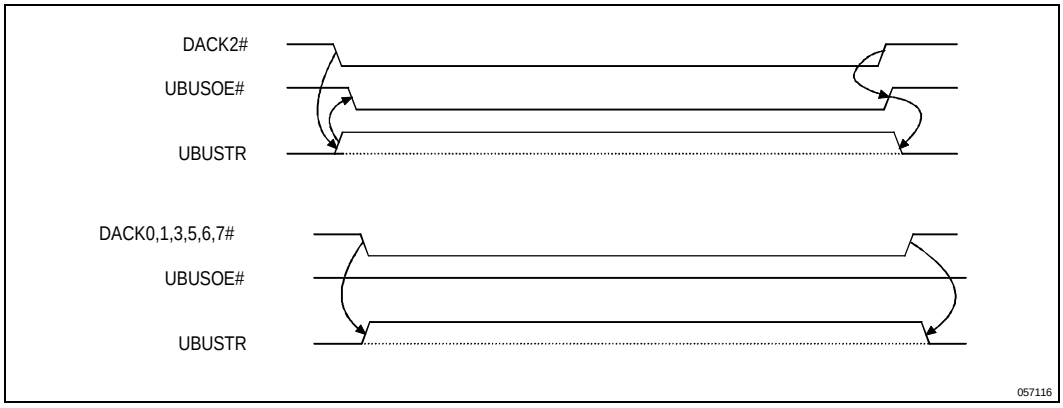


Figure 16. Utility Bus Access (DMA)

4.11. Power Management

The SIO/SIO.A 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 SIO/SIO.A provides a Fast-On/Off feature that creates a third state called Fast-Off (Figure 17). When in the Fast-Off state, the system consumes less power than the Power-On state.

The SIO/SIO.A power management architecture is based on three functions—System 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 SIO/SIO.A 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.

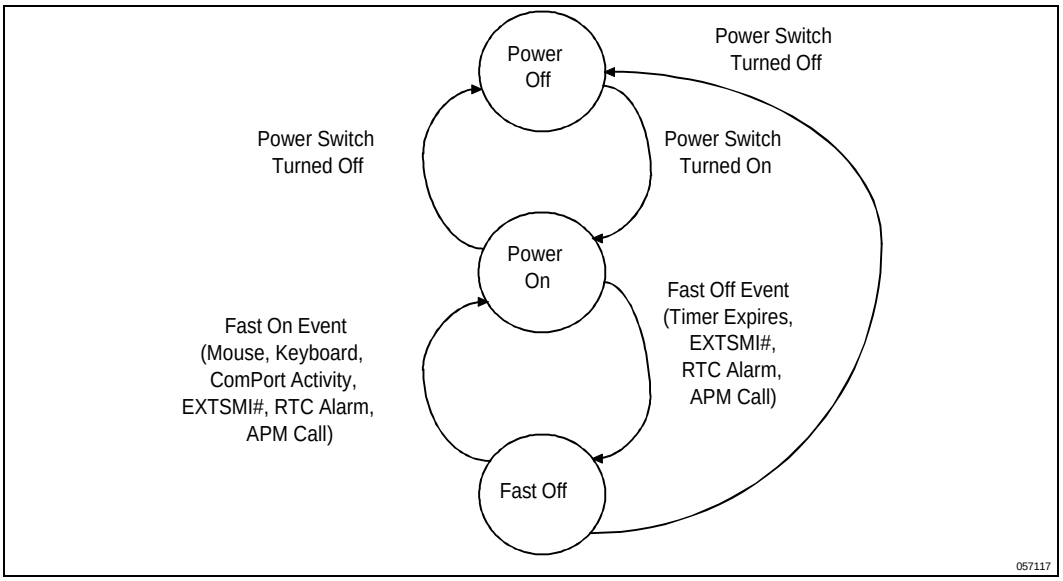


Figure 17. Fast-On/Off Flow

4.11.1. SMM MODE

SMM mode is invoked by asserting the SMI# signal to the CPU. The SIO/SIO.A 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.

4.11.2. 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 Register Description section.

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. These events are indicated by the assertion of IRQ[15:9,8#,7:3,1:0], NMI, or SMI signals. System events also include certain peripheral I/O address accesses (Table 10). 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 or an access within the defined range), the Fast-Off Timer is re-loaded with its initial count.

Table 10. Decode For System Events To Trigger Fast-Off Timer Re-load

Address Range	Device
170-17Fh, 1F0-1FFh	IDE/Floppy
278-27Fh	Printer
2E8-2EFh, 2F8-2FFh	COM
320-32Fh	IDE/Floppy
378-37Fh	Printer
3BC-3BFh	Printer
370-377h, 3F0-3F7h	IDE/FLOPPY
3E8-3EFh, 3F8-3FFh	COM
0-0Fh	DMA Registers[1]
80-8Fh	DMA Registers[1]
C0-DEh	DMA Registers[1]
400-43Fh, 481-4FFh	DMA Registers[1]

NOTE:
1. Access to positively decoded DMA registers (and aliases) creates a system event.

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.

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

4.11.3. 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 SIO/SIO.A only asserts SMI# when INIT is negated. If the SIO/SIO.A asserts SMI# and then the INIT signal is sampled asserted, the SIO/SIO.A negates SMI#.

4.11.4. 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 SIO/SIO.A 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 IN T) 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 SIO/SIO.A CSMIGATE (or sets the SIO/SIO.A 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 SIO/SIO.A 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 µsec internal clock. This allows a programmable timer interval for both the STPCLK# high and low times of 0–8 µsec. 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. If wait for stop clock special cycle is enabled (SMICNTL Register), the SIO/SIO.A does not start the timer until the Stop Grant Special Cycle is received. When the timer reaches 00h, STPCLK# is negated. Note that for the

82379AB, if the wait for stop grant special cycle function is disabled, the SIO.A does not wait for the stop grant special cycle before starting the timer. Note, also, that a break event also negates STPCLK#. This feature is not programmable in the 82378ZB.

NOTE

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

4.11.5. DUAL-PROCESSOR POWER MANAGEMENT SUPPORT (82379AB Only)

Figure 18 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.

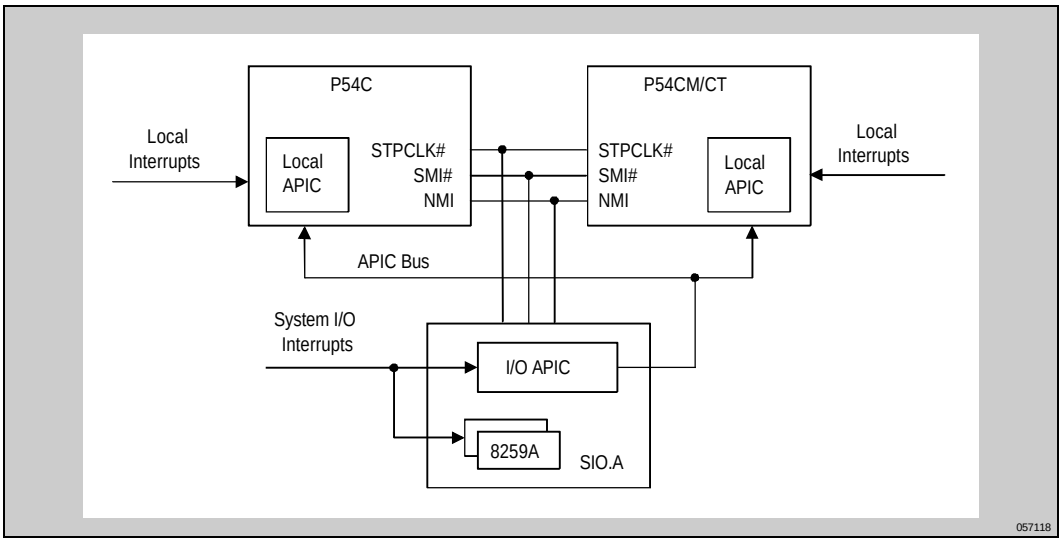


Figure 18. SIO.A Dual Processor System Configuration

4.11.5.1. SMI# Delivery Mechanism

For Uni or CT upgrade processor system configuration, SMI# can either be delivered through the SIO.A SMI# signal or I/O APIC. For the P54C/CM Dual-processor configuration, SMI# should be deliver 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.

4.11.5.2. 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. When bit 6=0 in the SMICNTL Register, the SIO.A recognizes the first STPGNT# assertion and negates STPCLK# upon the Stop Clock timer expiration or a stop break event.

4.11.5.3. 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 SIO.A 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.

4.11.6. INTERRUPT LEVELS AND SYSTEM EVENT GENERATION IN POWER MANAGED SYSTEMS (82378ZB Only)

The 82378ZB (SIO) can use the 14 IRQ input pins (IRQ[15,9,8,7:3,1]) to generate hardware system events in PCI systems that support power management mode by programming the System Event Enable Register bits [15:0], corresponding to the selected interrupt. Detection of these events causes the Fast-Off Timer to be re-loaded with its initial count value and negate the STPCLK# pin. The SIO samples the enabled interrupts as level sensitive, active high signals, for system event determination.

Most motherboard devices or ISA add-in cards using an interrupt drive the interrupt low when the interrupt is inactive, and only drive the interrupt high when it needs to generate an interrupt to the CPU. These devices work properly with the 82378ZB level sensitive logic.

PC-AT System Design Considerations

Devices or ISA add-in cards that float their interrupt line (i.e., a device that doe not drive its interrupt low when inactive) may not be able to use the power management capabilities of the 82378ZB. Thus, it is recommended that these devices not be used with the 82378ZB.

4.12. Design Considerations (82378ZB/82379AB)

4.12.1. Good Layout Practice

Incorrect board layout practices have shown that care must be taken to minimize ground bounce that could generate noise glitches above the threshold on the following signals—EOP, BALE, NMI, INT, DEVSEL#, RSTDRV, and DACK[7:0]. Care should be taken not to route these signals long distances and to ensure that they are terminated properly. An additional good layout practice is to pull any affected signals to ground with a capacitor (120 pF) at each receiver. For DEVSEL#, a 50 pF capacitor should be sufficient.

4.12.2. ASYNCHRONOUSLY SWITCHING SIGNALS

Good design practice should ensure that asynchronously switching signals should not be routed along synchronous signals to minimize crosstalk. On the SIO/SIO.A, the following signals switch asynchronously—ECSADDR[2:0], ECSEN#, MEM16#, MEMCS#, UBUSOE#, UBUSTR.

The IRQ lines are susceptible to cross coupled switching noise from these asynchronous signals. It is recommended that the IRQ signals not be routed along these asynchronously switching signals. If an IRQ signal is routed along these signals, capacitors should be used (100 pF) to decouple the switching from the IRQ input.

5.0. ELECTRICAL CHARACTERISTICS

5.1. Maximum Ratings

Case Temperature Under Bias	-65°C to 110°C.
Storage Temperature	-65°C to 150°C.
Supply Voltages with Respect to Ground	-0.5V to V_{CC} + 0.5V
Voltage On Any Pin	-0.5V to V_{CC} + 0.5V

WARNING

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

6.0. PIN ASSIGNMENT

The SIO and SIO.A packages are 208-pin Quad Flatpacks (QFP). The package signals are shown in Figure 19 and listed in Table 12. The following notations are used to describe pin types.

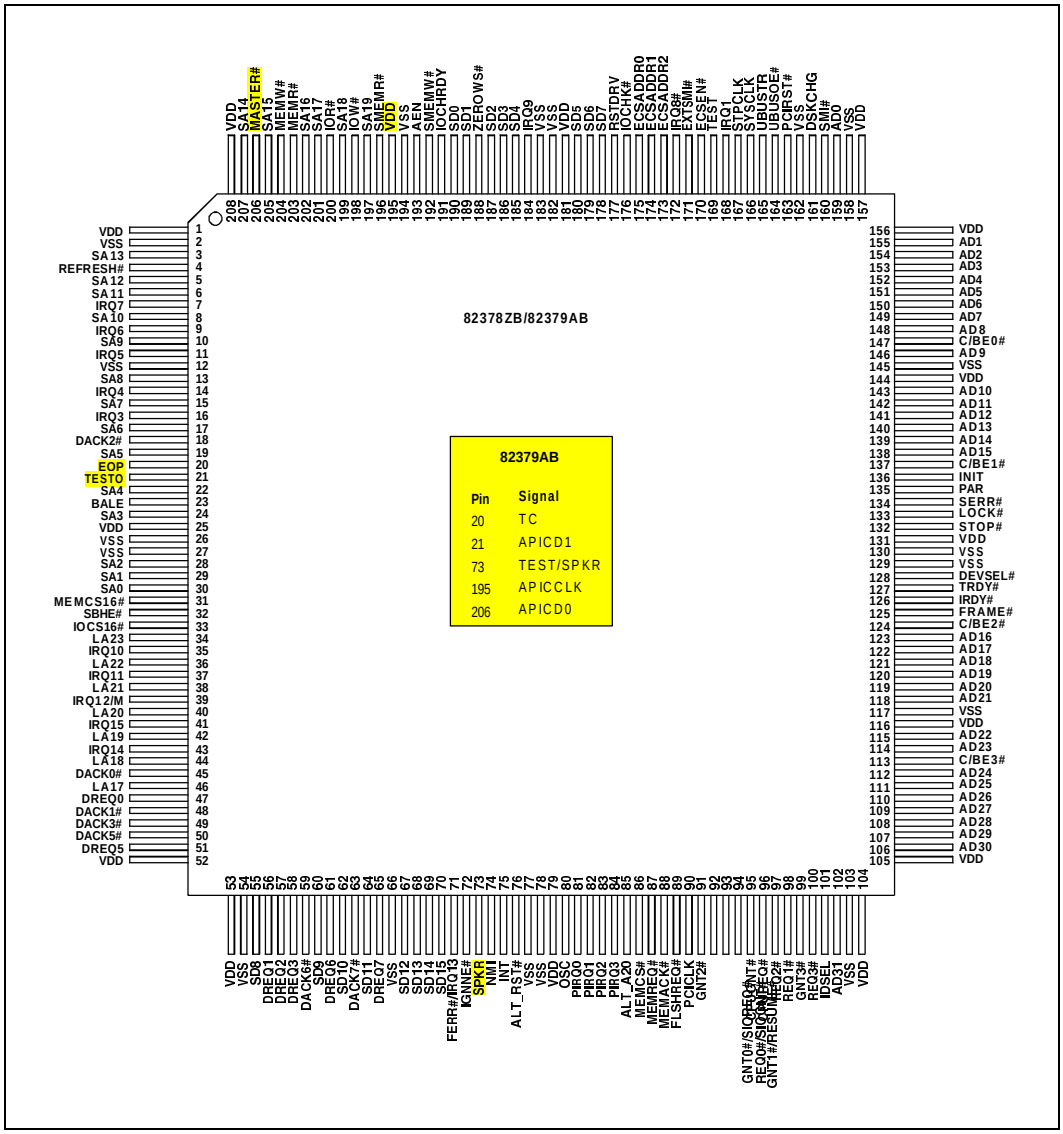


Figure 19. 82379AB Pinout

Table 11. Alphabetical Pin Assignment (82379AB)

Pin Name	Pin #	Type
AD00	159	I/O
AD01	155	I/O
AD02	154	I/O
AD03	153	I/O
AD04	152	I/O
AD05	151	I/O
AD06	150	I/O
AD07	149	I/O
AD08	148	I/O
AD09	146	I/O
AD10	143	I/O
AD11	142	I/O
AD12	141	I/O
AD13	140	I/O
AD14	139	I/O
AD15	138	I/O
AD16	123	I/O
AD17	122	I/O
AD18	121	I/O
AD19	120	I/O
AD20	119	I/O
AD21	118	I/O
AD22	115	I/O
AD23	114	I/O
AD24	112	I/O
AD25	111	I/O
AD26	110	I/O
AD27	109	I/O
AD28	108	I/O
AD29	107	I/O
AD30	106	I/O
AD31	102	I/O
AEN	193	O
ALT_A20	85	O
ALT_RST#	76	O
APICCLK (82379AB)	195	I
APICD0 (82379AB)	206	OD
APICD1 (82379AB)	21	OD
BALE	23	O
C/BE0#	147	I/O
C/BE1#	137	I/O
C/BE2#	124	I/O
C/BE3#	113	I/O
CPUGNT#	95	t/s/o
CPUREQ#	96	I
DACK0#	45	O
DACK1#	48	O
DACK2#	18	O
DACK3#	49	O
DACK5#	50	O
DACK6#	59	O
DACK7#	63	O
DEVSEL#	128	I/O (s/t/s)
DREQ0	47	I
DREQ1	56	I
DREQ2	57	I
DREQ3	58	I
DREQ5	51	I
DREQ6	61	I
DREQ7	65	I
DSKCHG	161	I
ECSADDR0	175	O
ECSADDR1	174	O
ECSADDR2	173	O
ECSEN#	170	O
EOP (82378ZB)	20	I/O
EXTSMI#	171	I
FERR#/ IRQ13	71	I
FLSHREQ#	89	t/s/o
FRAME#	125	I/O (s/t/s)
GNT0#/ SIOREQ#	92	t/s/o
GNT1#/ RESUME#	94	t/s/o
GNT2#	91	t/s/o
GNT3#	99	t/s/o
IDSEL	101	I
IGNNE#	72	O
INIT	136	I
INT	75	O
IOCHK#	176	I
IOCHRDY	191	I/O
IOCS16#	33	I
IOR#	200	I/O
IOW#	198	I/O
IRDY#	126	I/O (s/t/s)

Pin Name	Pin #	Type
IRQ1	168	I
IRQ10	35	I
IRQ11	37	I
IRQ12/M	39	I
IRQ14	43	I
IRQ15	41	I
IRQ3	16	I
IRQ4	14	I
IRQ5	11	I
IRQ6	9	I
IRQ7	7	I
IRQ8#	172	I
IRQ9	184	I
LA17	46	I/O
LA18	44	I/O
LA19	42	I/O
LA20	40	I/O
LA21	38	I/O
LA22	36	I/O
LA23	34	I/O
LOCK#	133	I (s/t/s)
MASTER# (82378ZB)	206	I
MEMACK#	88	I
MEMCS#	86	O
MEMCS16#	31	I/O (o/d)
MEMR#	203	I/O
MEMREQ#	87	t/s/o
MEMW#	204	I/O
NMI	74	O

Pin Name	Pin #	Type
OSC	80	I
PAR	135	O
PCICLK	90	I
PCIRST#	163	I
PIRQ0#	81	I
PIRQ1#	82	I
PIRQ2#	83	I
PIRQ3#	84	I
REFRESH#	4	I/O
REQ0#/ SIOGNT#	93	I
REQ1#	98	I
REQ2#	97	I
REQ3#	100	I
RSTDRV	177	O
SA00	30	I/O
SA01	29	I/O
SA02	28	I/O
SA03	24	I/O
SA04	22	I/O
SA05	19	I/O
SA06	17	I/O
SA07	15	I/O
SA08	13	I/O
SA09	10	I/O
SA10	8	I/O
SA11	6	I/O
SA12	5	I/O
SA13	3	I/O
SA14	207	I/O
SA15	205	I/O

Pin Name	Pin #	Type
SA16	202	I/O
SA17	201	I/O
SA18	199	I/O
SA19	197	I/O
SBHE#	32	I/O
SD00	190	I/O
SD01	189	I/O
SD02	187	I/O
SD03	186	I/O
SD04	185	I/O
SD05	180	I/O
SD06	179	I/O
SD07	178	I/O
SD08	55	I/O
SD09	60	I/O
SD10	62	I/O
SD11	64	I/O
SD12	67	I/O
SD13	68	I/O
SD14	69	I/O
SD15	70	I/O
SERR#	134	I
SMEMR#	196	O
SMEMW#	192	O
SMI#	160	O
SPKR (82378ZB)	73	O
SPKR/TESTO (82379AB)	73	I/O
STOP#	132	I/O (s/t/s)
STPCLK#	167	O

Pin Name	Pin #	Type
SYSCLK	166	O
TC (82379AB)	20	O
TEST	169	I
TESTO (82378ZB)	21	O
TRDY#	127	I/O (s/t/s)
UBUSOE#	164	O
UBUSTR	165	O
V_{DD}	1	V
V_{DD}	79	V
V_{DD}	104	V
V_{DD}	105	V
V_{DD}	116	V
V_{DD}	131	V

Pin Name	Pin #	Type
V_{DD}	144	V
V_{DD}	156	V
V_{DD}	157	V
V_{DD}	181	V
V_{DD} (82378ZB)		
V_{DD}	25	V
V_{DD}	52	V
V_{DD}	53	V
V_{DD}	208	V
V_{SS}	2	V
V_{SS}	12	V
V_{SS}	26	V
V_{SS}	27	V
V_{SS}	54	V
V_{SS}	66	V

Pin Name	Pin #	Type
V_{SS}	77	V
V_{SS}	78	V
V_{SS}	103	V
V_{SS}	117	V
V_{SS}	129	V
V_{SS}	130	V
V_{SS}	145	V
V_{SS}	158	V
V_{SS}	162	V
V_{SS}	182	V
V_{SS}	183	V
V_{SS}	194	V
ZEROWS#	188	I

7.0. MECHANICAL SPECIFICATIONS

7.1. Package Diagram

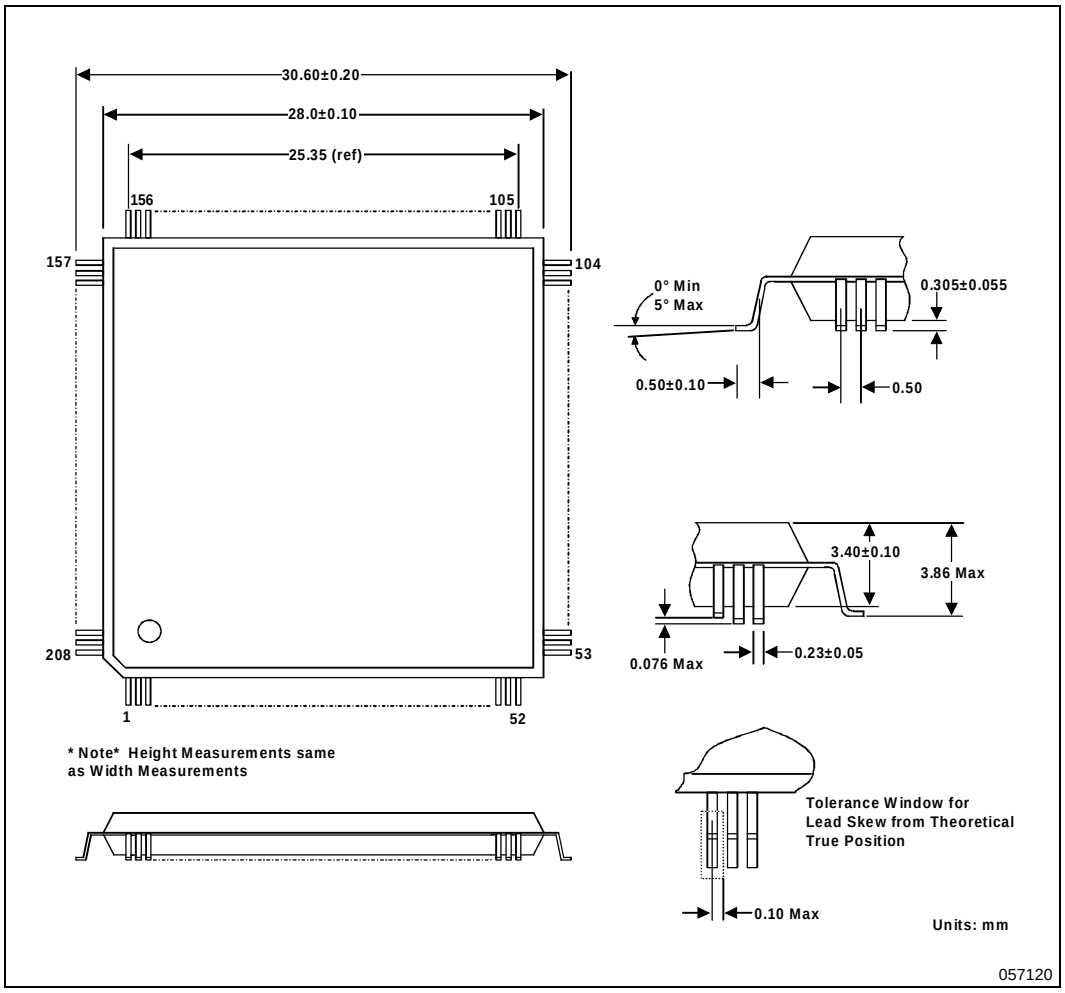


Figure 20. 208-Pin Quad Flat Pack (QFP) Package Dimensions

7.2. Thermal Specifications

Table 12. 82378 QFP Package Thermal Characteristics

Thermal Resistance - °C/Watt			
	Air Flow Rate (Ft./Min)		
Parameter	**0**	**200**	**400**
θ Junction to Case	6.6	6.6	6.6
θ Case to Ambient	36.6	27.4	24

8.0. TESTABILITY

The TEST and TESTO pins are used to test the SIO/SIO.A. During normal operations, the TEST pin must be grounded. The test output TESTO may be left as a no-connect (NC).

8.1. Global Tri-State

The TEST pin and IRQ3 are used to provide a high-impedance tri-state test mode. When the following input combination occurs, all outputs and bi-directional pins are tri-stated, with the exception of TESTO:

TEST = '1'
IRQ3 = '1'

The SIO/SIO.A must be reset after the bi-directional and output pins have been tri-stated in this manner.

8.2. Nand Tree

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

The TEST pin, along with IRQ5 or IRQ6, activates the NAND Tree. All bi-directional pins, and certain pure output pins using bi-directional buffers for performance reasons, are tri-stated when the following input combinations occur:

TEST = '1'
IRQ5 = '1'
- or -
TEST = '1'
IRQ6 = '0'

The output pulse train is observed at the TESTO test output. Pure output pins are not included directly in the NAND Tree. As noted in Section 8.3, each output can be expected to toggle after the corresponding node noted next to the pin name toggles from a "1" to a "0".

The sequence of the ATE test is as follows:

1. Drive TEST and IRQ5 high or TEST high and IRQ6 low.
2. Drive each input and bi-directional pin noted in Section 8.3 high.
3. Starting with the pin farthest from TESTO (SA8 on the 82378ZB and SMI# on the 82379AB), individually drive each pin low. Expect TESTO to toggle with each pin. Expect each pure output noted in Section 8.3 to toggle after each corresponding input pin has been driven low.
4. Turn off tester drivers before driving TEST low.
5. Reset the SIO/SIO.A prior to proceeding with further testing.

8.3. NAND Tree Cell Order

Table 13. NAND Tree Cell Order for 82378ZB

Tree Output#	Pin #	Pin Name	Notes
	14	IRQ4	Reserved
	21	TESTO	Test Mode Output
1	11	IRQ5	Cell Closest to TESTO
2	10	SA9	
3	9	IRQ6	
4	8	SA10	
5	7	IRQ7	
6	6	SA11	
7	5	SA12	
8	4	REFRESH#	
9	3	SA13	
10	207	SA14	
11	206	MASTER#	
12	205	SA15	
13	204	MEMW#	
14	203	MEMR#	
15	202	SA16	
16	201	SA17	
17	200	IOR#	
18	199	SA18	
19	198	IOW#	
20	197	SA19	

Tree Output#	Pin #	Pin Name	Notes
21	196	SMEMR#	
22	193	AEN	
23	192	SMEMW#	
24	191	IOCHRDY	
25	190	SD0	
26	189	SD1	
27	188	ZEROWS#	
28	187	SD2	
29	186	SD3	
30	185	SD4	
31	184	IRQ9	
32	180	SD5	
33	179	SD6	
34	178	SD7	
35	177	RSTDRV	
36	176	IOCHK#	
	175	ECSADDR0	NAND Tree Output of Tree Cell 28
	174	ECSADDR1	NAND Tree Output of Tree Cell 29
	173	ECSADDR2	NAND Tree Output of Tree Cell 30
37	172	IRQ8#	
38	171	EXTSMI#	
	170	ECSEN#	NAND Tree Output of Tree Cell 32
	169	TEST	PI => VCC, TEST must be '1'
39	168	IRQ1	
	167	STPCLK#	
40	166	SYSCLK	
	165	UBUSTR	NAND Tree Output of Tree Cell 33
	164	UBUSOE#	NAND Tree Output of Tree Cell 34
41	163	PCIRST#	
42	161	DSKCHG	
	160	SMI#	

Tree Output#	Pin #	Pin Name	Notes
43	159	AD0	
44	155	AD1	
45	154	AD2	
46	153	AD3	
47	152	AD4	
48	151	AD5	
49	150	AD6	
50	149	AD7	
51	148	AD8	
52	147	C/BE0#	
53	146	AD9	
54	143	AD10	
55	142	AD11	
56	141	AD12	
57	140	AD13	
58	139	AD14	
59	138	AD15	
60	137	C/BE1#	
61	136	INIT	
62	135	PAR	
63	134	SERR#	
64	133	LOCK#	
65	132	STOP#	
66	128	DEVSEL#	
67	127	TRDY#	
68	126	IRDY#	
69	125	FRAME#	
70	124	C/BE2#	
71	123	AD16	
72	122	AD17	
73	121	AD18	

Tree Output#	Pin #	Pin Name	Notes
74	120	AD19	
75	119	AD20	
76	118	AD21	
77	115	AD22	
78	114	AD23	
79	113	C/BE3#	
80	112	AD24	
81	111	AD25	
82	110	AD26	
83	109	AD27	
84	108	AD28	
85	107	AD29	
86	106	AD30	
87	102	AD31	
88	101	IDSEL	
89	100	REQ3#	
90	98	REQ1#	
91	97	REQ2#	
92	96	CPUREQ#	
	95	CPUGNT#	NAND Tree Output of Tree Cell 93
	94	GNT1#	NAND Tree Output of Tree Cell 95
93	93	REQ0#	
	92	GNT0#	NAND Tree Output of Tree Cell 100
94	90	PCICLK	
	89	FLSHREQ#	NAND Tree Output of Tree Cell 102
95	88	MEMACK#	
	87	MEMREQ#	NAND Tree Output of Tree Cell 103
	86	MEMCS#	NAND Tree Output of Tree Cell 104
	85	ALT_A20	NAND Tree Output of Tree Cell 105
96	84	PIRQ[3]#	
97	83	PIRQ[2]#	

Tree Output#	Pin #	Pin Name	Notes
98	82	PIRQ[1]#	
99	81	PIRQ[0]#	
100	80	OSC	
	76	ALT_RST#	NAND Tree Output of Tree Cell 23
	75	INT	NAND Tree Output of Tree Cell 24
	74	NMI	NAND Tree Output of Tree Cell 25
101	73	SPKR	
	72	IGNNE#	NAND Tree Output of Tree Cell 26
102	71	FERR#	
103	70	SD15	
104	69	SD14	
105	68	SD13	
106	67	SD12	
107	65	DREQ7	
108	64	SD11	
109	63	DACK7#	
110	62	SD10	
111	61	DREQ6	
112	60	SD9	
113	59	DACK6#	
114	58	DREQ3	
115	57	DREQ2	
116	56	DREQ1	
117	55	SD8	
118	51	DREQ5	
119	50	DACK5#	
120	49	DACK3#	
121	48	DACK1#	
122	47	DREQ0	
123	46	LA17	
124	45	DACK0#	

Tree Output#	Pin #	Pin Name	Notes
125	44	LA18	
126	43	IRQ14	
127	42	LA19	
128	41	IRQ15	
129	40	LA20	
130	39	IRQ12/M	
131	38	LA21	
132	37	IRQ11	
133	36	LA22	
134	35	IRQ10	
135	34	LA23	
136	33	IOCS16#	
137	32	SBHE#	
138	31	MEMCS16#	
139	30	SA0	
140	29	SA1	
141	28	SA2	
142	24	SA3	
143	23	BALE	
144	22	SA4	
145	20	EOP	
146	19	SA5	
147	18	DACK2#	
148	17	SA6	
149	16	IRQ3	Output signals will transition from high-impedance state to driving state after this pin is driven low.
150	15	SA7	
151	13	SA8	Cell furthest from TESTO Start of NAND Tree

Table 14. NAND Tree Cell Order for 82379AB

Order	Pin #	Pin Name
1	160	SMI#
2	91	GNT2#
3	99	GNT3#
4	195	APICCLK
5	21	APICD1
6	13	SA8
7	15	SA7
8	16	IRQ3
9	17	SA6
10	18	DACK2#
11	19	SA5
12	20	TC
13	22	SA4
14	23	BALE
15	24	SA3
16	28	SA2
17	29	SA1
18	30	SA0
19	31	MEMCS16#
20	32	SBHE#
21	33	IOCS16#
22	34	LA23
23	35	IRQ10
24	36	LA22
25	37	IRQ11
26	38	LA21
27	39	IRQ12/M
28	40	LA20
29	41	IRQ15
30	42	LA19
31	43	IRQ14
32	44	LA18
33	45	DACK0#
34	46	LA17

Table 14. NAND Tree Cell Order for 82379AB

Order	Pin #	Pin Name
35	47	DREQ0
36	48	DACK1#
37	49	DACK3#
38	50	DACK5#
39	51	DREQ5
40	55	SD8
41	56	DREQ1
42	57	DREQ2
43	58	DREQ3
44	59	DACK6#
45	60	SD9
46	61	DREQ6
47	62	SD10
48	63	DACK7#
49	64	SD11
50	65	DREQ7
51	67	SD12
52	68	SD13
53	69	SD14
54	70	SD15
55	71	FERR#
56	72	IGNNE#
57	74	NMI
58	75	INT
59	76	ALT_RST#
60	80	OSC
61	81	PIRQ0#
62	82	PIRQ1
63	83	PIRQ2
64	84	PIRQ3#
65	85	ALT_A20
66	86	MEMCS#
67	87	MEMREQ#
68	88	MEMACK#

Table 14. NAND Tree Cell Order for 82379AB

Order	Pin #	Pin Name
69	89	FLSHREQ#
70	90	PCICLK
71	92	GNT0
72	93	REQ0#
73	94	GNT1#
74	95	CPUGNT#
75	96	CPUREQ#
76	97	REQ2#
77	98	REQ1#
78	100	REQ3#
79	101	IDSEL
80	102	AD31
81	106	AD30
82	107	AD29
83	108	AD28
84	109	AD27
85	110	AD26
86	111	AD25
87	112	AD24
88	113	C/BE3#
89	114	AD23
90	115	AD22
91	118	AD21
92	119	AD20
93	120	AD19
94	121	AD18
95	122	AD17
96	123	AD16
97	124	C/BE2#
98	125	FRAME#
99	126	IRDY#
100	127	TRDY#
101	128	DEVSEL#
102	132	STOP#

Table 14. NAND Tree Cell Order for 82379AB

Order	Pin #	Pin Name
103	133	LOCK#
104	134	SERR#
105	135	PAR
106	136	INIT
107	137	C/BE1#
108	138	AD15
109	139	AD14
110	140	AD13
111	141	AD12
112	142	AD11
113	143	AD10
114	146	AD9
115	147	C/BE0#
116	148	AD8
117	149	AD7
118	150	AD6
119	131	AD5
120	152	AD4
121	153	AD3
122	154	AD2
123	155	AD1
124	159	AD0
125	161	DSKCHG
126	163	PCIRST#

Table 14. NAND Tree Cell Order for 82379AB

Order	Pin #	Pin Name
127	164	UBUSOE#
128	165	UBUSTR
129	166	SYSCLK
130	167	STPCLK#
131	168	IRQ1
132	170	ECSEN#
133	171	EXTSMI#
134	172	IRQ8#
135	173	ECSADDR2
136	174	ECSADDR1
137	175	ECSADDR0
138	176	IOCHK#
139	177	RSTDRV
140	178	SD7
141	179	SD6
142	180	SD5
143	184	IRQ9
144	185	SD4
145	186	SD3
146	187	SD2
147	188	ZEROWS#
148	189	SD1
149	190	SD0
150	191	IOCHRDY

Table 14. NAND Tree Cell Order for 82379AB

Order	Pin #	Pin Name
151	192	SMEMW#
152	193	AEN
153	196	SMEMR#
154	197	SA19
155	198	IOW#
156	199	SA18
157	200	IOR#
158	201	SA17
159	202	SA16
160	203	MEMR#
161	204	MEMW#
162	205	SA15
163	206	APICD0
164	207	SA14
165	3	SA13
166	4	REFRESH#
167	5	SA12
168	6	SA11
169	7	IRQ7
170	8	SA10
171	9	IRQ6
172	10	SA9
173	11	IRQ5

8.4. NAND Tree Diagram

Figures 21 and 22 shows the NAND Tree Diagrams. Note that the Pulse Generator shown is only used for testing the IRQ4 signal. It is enabled when IRQ4 is pulsed high.

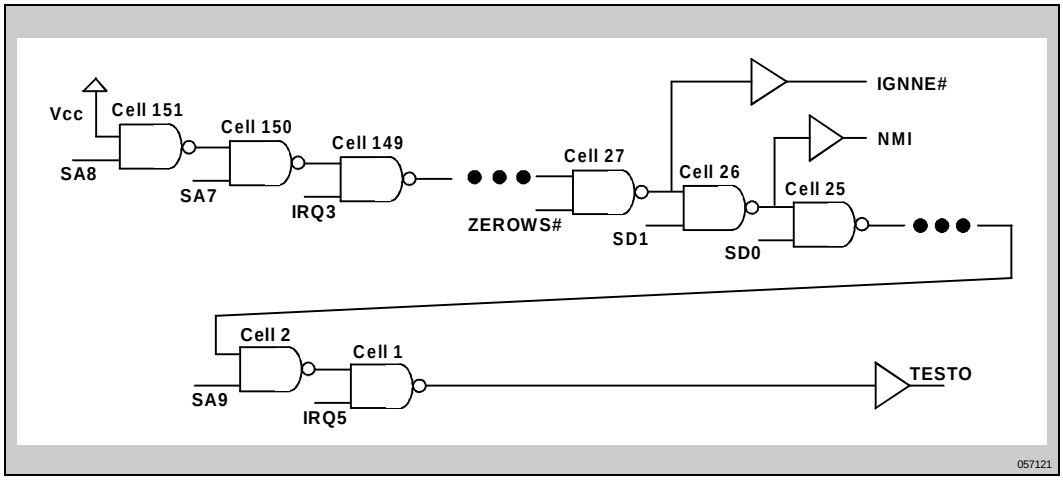


Figure 21. NAND Tree Diagram for 82378ZB

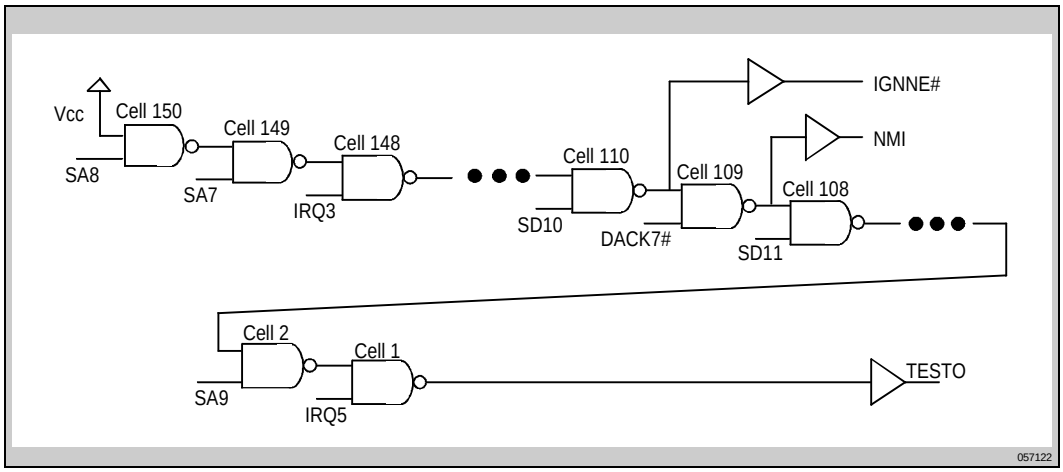


Figure 22. NAND Tree Diagram For the 82379AB