PRELIMINARY

82093AA I/O ADVANCED PROGRAMMABLE INTERRUPT CONTROLLER (IOAPIC)

- **Provides Multiprocessor Interrupt Management**
  - — **Dynamic Interrupt Distribution-Routing Interrupt to the Lowest Priority Processor**
  - — **Software Programmable Control of Interrupt Inputs**
  - — **Off Loads Interrupt Related Traffic From the Memory Bus**
- **24 Programmable Interrupts**
  - — **13 ISA Interrupts Supported**
  - — **4 PCI Interrupts**
  - — **1 Interrupt/SMI# Rerouting**
  - — **2 Motherboard Interrupts**
  - — **1 Interrupt Used for INTR Input**
  - — **3 General Purpose Interrupts**
  - — **Independently Programmable for Edge/Level Sensitivity Interrupts**
  - — **Each Interrupt Can Be Programmed to Respond to Active High or Low Inputs**
- **X-Bus Interface**
  - — **CS For Flexible Decode of the IOAPIC Device.**
  - — **Index Register Interface for Optimum Memory Usage**
  - — **Registers are 32-Bit Wide to Match the PCI to Host Bridge Architecture**
- **Package 64-Pin PQFP**

The 82093AA I/O Advanced Programmable Interrupt Controller (IOAPIC) 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. Each interrupt pin is individually programmable as either edge or level triggered. The interrupt vector and interrupt steering information can be specified per interrupt. An indirect register accessing scheme optimizes the memory space needed to access the IOAPIC's internal registers. To increase system flexibility when assigning memory space usage, the The IOAPIC's 2-register memory space is re-locatable.

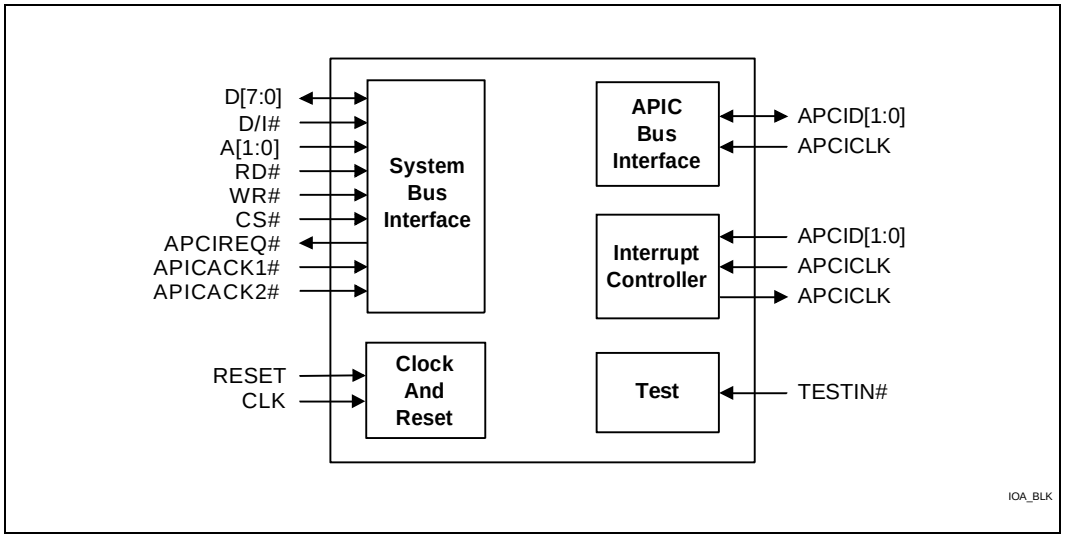

Figure 1. IOAPIC Simplified Block Diagram

May 1996

Order Number: 290566-001

CONTENTS

1.0. OVERVIEW

While the standard ISA Compatible interrupt controller (located in the PIIX3) is intended for use in a uni-processor system, the I/O Advanced Programmable Interrupt Controller (IOAPIC) can be used in either a uni-processor or multi-processor system. The IOAPIC 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, the IOAPIC'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 in the PIIX3, the IOAPIC unit, or mixed mode where both the standard ISA Compatible Interrupt Controller and IOAPIC 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 2.0)—one residing in the I/O subsystem (called the **IOAPIC**) and the other in the CPU (called the **Local APIC**). The local APIC and the IOAPIC communicate over a dedicated APIC bus. The IOAPIC bus interface consists of two bi-directional data signals (APICD[1:0]) and a clock input (APICCLK).

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

The IOAPIC Unit consists of a set of interrupt input signals, a 24-entry by 64-bit Interrupt Redirection Table, programmable registers, and a message unit for sending and receiving APIC messages over the APIC bus. I/O devices inject interrupts into the system by asserting one of the interrupt lines to the IOAPIC. The IOAPIC 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 IOAPIC contains a set of programmable registers. Two of the registers (I/O Register Select and I/O Window Registers) are located in the CPU's memory space and are used to indirectly access the other APIC registers as described in Section 3.0, Register Description. The Version Register provides the implementation version of the IOAPIC. The IOAPIC ID Register is programmed with an ID value that serves as a physical name of the IOAPIC. This ID is loaded into the ARB ID Register when the IOAPIC ID Register is written and is used during bus arbitration.

NOTE

The interrupt number or the vector does not imply a particular priority for being sent. The IOAPIC continually polls the 24 interrupts in a rotating fashion, one at a time. The pending interrupt polled first is the one sent.

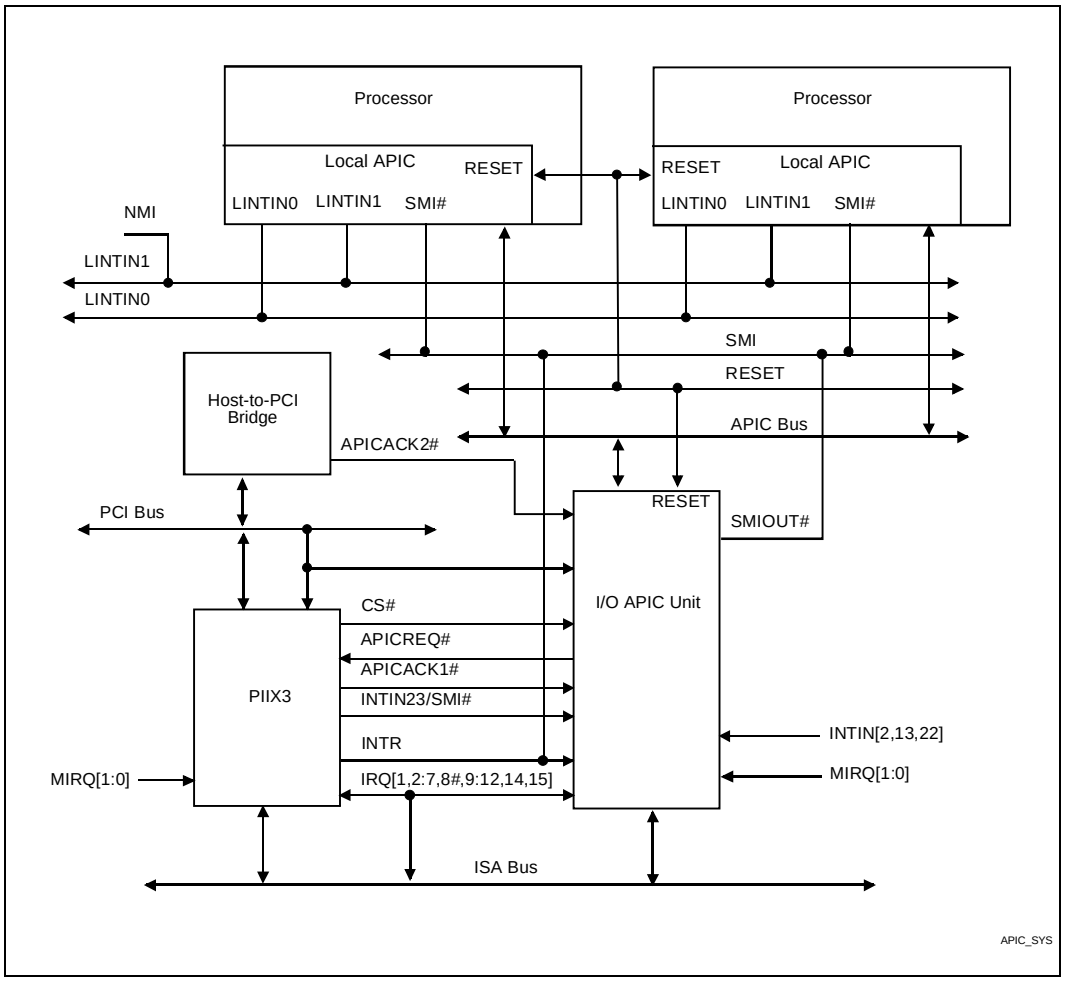

Figure 2. I/O And Local APIC Units

2.0. SIGNAL DESCRIPTION

This section contains a detailed description of each signal. The signals are arranged in function groups according to their 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 describe the signal and type:

I	Input pin
O	Output pin
ST	Schmitt Trigger Input pin
OD	Open Drain Output pin. This requires a pull-up to the VCC of the processor core
I/OD	Bi-directional Input withOpen Drain Output pin.
I/O	Bi-directional Input/Output pin

2.1. System Bus Signals

Signal Name	Type	Description
D[7:0]	I/O	**DATA:** D[7:0] contain the data when writing to or reading from internal IOAPIC registers. These signals are outputs when reading data from the IOAPIC and they are inputs when writing data to the IOAPIC. These signals are tri-stated during reset.
D/I#	I	**DATA/INDEX#:** This input selects whether the I/O Register Select (IOREGSEL) Register or I/O Window (IOWIN) Register is accessed. All internal IOAPIC registers are accessed with an indexing scheme. When the D/I# pin is low, the IOREGSEL Register is accessed. When the D/I# pin is high, the data becomes available from the register pointed to by the index register. Typically, this signal is connected to SA4 on the ISA bus (i.e., IOREGSEL Register is at 00h and IOWIN Register is at 10h).
A[1:0]	I	**ADDRESS:** The IOAPIC is a 32 bit device with an 8 bit ISA interface. A[1:0] steer the data byte to the correct 8 bit location within the 32 bit register. Typically, these input signals are connected to SA[1:0] of the ISA bus.
RD#	I	**READ STROBE:** RD# causes the IOAPIC to respond by driving internal register data onto the D[7:0] pins. Typically this pin is connected to the MEMRD# signal on the ISA bus.
WR#	I	**WRITE STROBE:** When this signal transitions from low to high, the data present on the IOAPIC's D[7:0] signals are written to an internal register. Typically, this signal is connected to the MEMWR# signal on the ISA bus.
CS#	I	**CHIP SELECT:** This active low input selects the IOAPIC as the target of the current read or write transaction.

Signal Name	Type	Description
APICREQ#	O	**APIC REQUEST:** APICREQ# is asserted prior to the APIC sending an interrupt message over the APIC data bus. This is the request part of a handshake that insures system level buffer coherency prior to sending an interrupt over the APIC bus. This signal is tri-stated during reset. This signal has an internal pull-up resistor.
APICACK1#	I	**APIC ACKNOWLEDGE 1:** This signal is the acknowledge part of the handshake indicating that the APIC can send the interrupt message over the APIC bus. This signal is typically connected to the PIIX3.
APICACK2#	I	**APIC ACKNOWLEDGE 2:** This signal is the second half of the acknowledge handshake indicating that the APIC can send the interrupt message over the APIC bus. This signal is typically connected to the host-to-PCI bridge and along with APICREQ# and APICACK1# makes up the complete buffer coherency protocol cycles. If the system does not have a host-to-PCI bridge, this signal can be tied low.

2.2. Clock and Reset Signals

Signal Name	Type	Description
PCICLK	I	**PCI CLOCK:** This signal is used to synchronize and strobe the data buffer status signals (APICREQ#, APICACK1#, and APICACK2#). This signal is typically connect to the PCI clock.
RESET	I	**RESET:** RESET initializes the IOAPIC's internal logic and sets the register bits to their default value.

2.3. APIC Bus Interface

Signal Name	Type	Description
APICD[1:0]	I/OD	**APIC DATA:** These signals are used to send and receive data over the APIC bus. These signals are tri-stated during reset and must be pulled up to the appropriate VCC levels of the CPU.
APICCLK	I	**APIC CLOCK:** The input signal is used to determine when valid data is being sent over the APIC bus.

2.4. Interrupt Signals

Signal Name	Type	Description
INTIN0	ST	**Interrupt Input 0:** This signal is connected to the redirection table entry 0. Typically, this signal may be connected to the INTR on the PIIX3 to communicate the status of IRQ0 and IRQ13 interrupts. Note that the IRQ0 and IRQ13 interrupts are embedded in the PIIX3 and are not available to the rest of the system.
INTIN1	ST	**Interrupt Input 1:** INTIN1 is connected to the redirection table entry 1. Typically, this signal will be connected to the keyboard interrupt (IRQ1).

Signal Name	Type	Description
INTIN2	ST	**Interrupt Input 2:** This signal is connected to the redirection table entry 2. If IRQ0 interrupt is available in hardware, it is connected to this pin.
INTIN[3:11, 14,15]	ST	**Interrupt Inputs 3 through 11, 14 and 15:** These signals are connected to the redirection table entries 3:11, 14 & 15. Typically, these signals are connected to the ISA interrupts IRQ[3:7,8#,9:11,14:15] respectively.
INTIN12	ST	**Interrupt Input 12:** This signal is connected to the redirection table entry 12. Typically, this signal will be connected to the mouse interrupt (IRQ12/M).
INTIN13	ST	**Interrupt input 13:** This signal is connected to the redirection table entry 13. If IRQ13 interrupt is available in hardware, it is connected to this signal. If IRQ13 is not available, it is routed through the INTR interrupt and this signal becomes INTIN13 (redirection table entry 13).
INTIN[16:19]	ST	**Interrupt inputs 16 through 19:** These signals are connected to the redirection table entries [16:19]. Typically, these signals are connected to the PCI interrupts (PIRQ[0:3]). The steering of the PCI IRQs to the ISA IRQs is accomplished in the IOAPIC by setting the PCI redirection table entry to the correct ISA interrupt vector.
INTIN[20:21]	ST	**Interrupt inputs 20 and 21:** These signals are connected to the redirection table entries 20 and 21. Typically, these signals are connected to the motherboard interrupts (MIRQ[0:1]). These pins could be used for the NMI and INIT signals or just general purpose interrupts.
INTIN22	ST	**Interrupt input 22:** This signal is connect to the redirection table entry 22. This signal is a general purpose interrupt.
INTIN23/ SMI#	ST	**Interrupt input 23:** This signal is connected to the redirection table entry 23. This input has a special feature for the SMI# interrupt routing. If the Mask bit is not set, the signal is a normal interrupt input that is sent over the APIC bus just like all the other interrupts. When the Mask bit is set, the INTIN23/SMI# input is routed through the IOAPIC to the SMIOUT# output signal.
SMIOUT#	OD	**SMI OUTPUT:** This signal is an output in response to the SMI# input when the MASK bit for the redirection table entry number 23 is set. If the MASK bit is not set, the redirection table can be setup to deliver an SMI# over the APIC bus.

2.5. Test and Power Signals

Pin Name	Type	Description
TESTIN#	I	**TEST INPUT:** This active-low input is used to invoke test modes. TESTIN# should be pulled high during normal operation.
VCC		**VCC POWER PIN:** 5V ± 10%.
GND		**GROUND POWER PIN:**

3.0. REGISTER DESCRIPTION

The IOAPIC is addressed with a CS# and the D/I# pin. The PIIX3 decodes the IOAPIC in memory space and sends a CS# to the IOAPIC device, when it is selected. The D/I# pin selects between the IOREGSEL Register (D/I#=0) and the IOWIN Register (D/I#=1). Typically, D/I# is connected to SA4 on the ISA bus (i.e., IOREGSEL Register is at 00h and IOWIN Register is at 10h).

The IOAPIC 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 located in the PIIX3). These two registers are re-locateable (via the APICBASE Register) as shown in Table 3.1. In the IOAPIC only the IOREGSEL and IOWIN Registers are directly accesable in the memory address space.

To reference an IOAPIC register, a byte memory write that the PIIX3 decodes for the IOAPIC loads the IOREGSEL Register with an 8-bit value that specifies the IOAPIC register (address offset in Table 3.2) to be accessed. The IOWIN Register is then used to read/write the desired data from/to the IOAPIC register specified by bits [7:0] of the IOREGSEL Register. The IOWIN Register must be accessed as a Dword quantity.

The IOREGSEL and IOWIN Registers (Table 3.1) can be relocated via the APIC Base Address Relocation Register in the PIIX3 and are aligned on 128 bit boundaries. 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.

Table 1. Memory Mapped Registers For Accessing IOAPIC Registers

Memory Address	Mnemonic	Register Name	Access	D/I# SIgnal
FEC0 xy00h	IOREGSEL	I/O Register Select (index)	R/W	0
FEC0 xy10h	IOWIN	I/O Window (data)	R/W	1

NOTES:

xy are determined by the x and y fields in the APIC Base Address Relocation Register located in the PIIX3. Range for x = 0-Fh and the range for y = 0,4,8,Ch.

Table 2. IOAPIC Registers

Address Offset	Mnemonic	Register Name	Access
00h	IOAPICID	IOAPIC ID	R/W
01h	IOAPICVER	IOAPIC Version	RO
02h	IOAPICARB	IOAPIC Arbitration ID	RO
10–3Fh	IOREDTBL[0:23]	Redirection Table (Entries 0-23) (64 bits each)	R/W

NOTES:

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

3.1. Memory Mapped Registers for Accessing IOAPIC Registers

3.1.1. IOREGSEL—I/O REGISTER SELECT REGISTER

Memory Address: FEC0 xy00h (xy=See APICBASE Register in the PIIX3)
Default Value: 00h
Attribute: Read/Write

This register selects the IOAPIC Register to be read/written. The data is then read from or written to the selected register through the IOWIN Register.

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

3.1.2. IOWIN—I/O WINDOW REGISTER

Memory Address: FEC0 xy10h (xy=See APICBASE Register in PIIX3)
Default Value: 00h
Attribute: Read/Write

This register is used to write to and read from the register selected by the IOREGSEL Register. Readability/writability is determined by the IOAPIC register that is currently selected.

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

3.2. IOAPIC Registers

3.2.1. IOAPICID—IOAPIC IDENTIFICATION REGISTER

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

This register contains the 4-bit APIC ID. The ID serves as a physical name of the IOAPIC. All APIC devices 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 IOAPIC for message transmission.

Bit	Description
31:28	**Reserved.**
27:24	**IOAPIC Identification—R/W.** This 4 bit field contains the IOAPIC identification.
23:0	**Reserved.**

3.2.2. IOAPICVER—IOAPIC VERSION REGISTER

Address Offset: 01h
Default Value: 00170011h
Attribute: Read Only

The IOAPIC 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	Descriptions
31:24	**Reserved.**
23:16	**Maximum Redirection Entry—RO.** 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 for the IOAPIC minus one. The range of values is 0 through 239. For this IOAPIC, the value is 17h.
15:8	**Reserved.**
7:0	**APIC VERSION—RO.** This 8 bit field identifies the implementation version. The version number assigned to the IOAPIC is 11h.

3.2.3. IOAPICARB—IOAPIC ARBITRATION REGISTER

Address Offset: 02h
Default Value: 0000_0000h
Attribute: Read Only

The APICARB Register contains the bus arbitration priority for the IOAPIC. This register is loaded when the IOAPIC ID Register is written.

The APIC uses a one wire arbitration to win 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 asssumed) only for messages that are transmitted successfully (except, in the case of low priority messages where Arbitration ID is changed even if message was not successfully transmitted). A message is transmitted successfully if no checksum error or acceptance error is reported for that message. The APICARB Register is always loaded with IOAPIC ID during a "level triggered INIT with de-assert" message.

Bit	Description
31:28	**Reserved.**
27:24	**IOAPIC Identification—R/W.** This 4 bit field contains the IOAPIC Arbitration ID.
23:0	**Reserved.**

3.2.4. IOREDTBL[23:0]—I/O REDIRECTION TABLE REGISTERS

Address Offset:	10–11h (IOREDTBL0)	28–29h (IOREDTBL12)
	12–13h (IOREDTBL1)	2A–2Bh (IOREDTBL13)
	14–15h (IOREDTBL2)	2C–2Dh (IOREDTBL14)
	16–17h (IOREDTBL3)	2E–2Fh (IOREDTBL15)
	18–19h (IOREDTBL4)	30–31h (IOREDTBL16)
	1A–1Bh (IOREDTBL5)	32–33Fh (IOREDTBL17)
	1C–1Dh (IOREDTBL6)	34–35h (IOREDTBL18)
	1E–1Fh (IOREDTBL7)	36–37h (IOREDTBL19)
	20–21h (IOREDTBL8)	38–39h (IOREDTBL20)
	22–23h (IOREDTBL9)	3A–3Bh (IOREDTBL21)
	24–25h (IOREDTBL10)	3C–3Dh (IOREDTBL22)
	26–27h (IOREDTBL11)	3E–3Fh (IOREDTBL23)
Default Value:	xxx1 xxxx xxxx xxxxh	
Attribute:	Read/Write	

There are 24 I/O Redirection Table entry registers. Each register is a dedicated entry for each interrupt input signal. Unlike IRQ pins of the 8259A, the notion of interrupt priority is completely unrelated to the position of the physical interrupt input signal on the APIC. Instead, software determines the vector (and therefore the priority) for each corresponding interrupt input signal. For each interrupt signal, 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.

The IOAPIC responds to an edge triggered interrupt as long as the interrupt is wider than one CLK cycle. The interrupt input is asynchronous; thus, setup and hold times need to be guaranteed for at lease one rising edge of the CLK input. Once the interrupt is detected, a delivery status bit internal to the IOAPIC is set. A new edge on that Interrupt input pin will not be recongnized until the IOAPIC Unit broadcasts the corresponding message over the APIC bus and the message has been accepted by the destination(s) specified in the destination field. 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. **Destination Mode IOREDTBLx[11]** — **Logical Destination Address** 0, Physical Mode — IOREDTBLx[59:56] = APIC ID 1, Logical Mode — IOREDTBLx[63:56] = Set of processors
55:17	**Reserved.**

Bit	Description
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. 1=Level sensitive, 0=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 IOAPIC. 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. 0=High active, 1=Low active.
12	**Delivery Status (DELIVS)—RO.** The Delivery Status bit contains the current status of the delivery of this interrupt. Delivery Status is read-only and writes to this bit (as part of a 32 bit word) do not effect this bit. 0=IDLE (there is currently no activity for this interrupt). 1=Send Pending (the interrupt has been injected but 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. **Destination Mode IOREDTBLx[11]** — **Logical Destination Address** 0, Physical Mode — IOREDTBLx[59:56] = APIC ID 1, Logical Mode — IOREDTBLx[63:56] = Set of processors

<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.
<table>
<tr><th>Bits [10:8]</th><th>Mode</th><th>Description</th></tr>
<tr><td>000</td><td>Fixed</td><td>Deliver the signal on the INTR signal of all processor cores listed in the destination. Trigger Mode for "fixed" Delivery Mode can be edge or level.</td></tr>
<tr><td>001</td><td>Lowest Priority</td><td>Deliver the signal on the INTR signal of the processor core that is executing at the lowest priority among all the processors listed in the specified destination. Trigger Mode for "lowest priority". Delivery Mode can be edge or level.</td></tr>
<tr><td>010</td><td>SMI</td><td>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.</td></tr>
<tr><td>011</td><td>Reserved</td><td></td></tr>
<tr><td>100</td><td>NMI</td><td>Deliver the signal on the NMI signal of all processor cores listed in the destination. Vector information is ignored. NMI is treated as an edge triggered interrupt, even if it is programmed as a level triggered interrupt. For proper operation, this redirection table entry must be programmed to “edge” triggered interrupt.</td></tr>
<tr><td>101</td><td>INIT</td><td>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 is always treated as an edge triggered interrupt, even if programmed otherwise. For proper operation, this redirection table entry must be programmed to “edge” triggered interrupt.</td></tr>
<tr><td>110</td><td>Reserved</td><td></td></tr>
<tr><td>111</td><td>ExtINT</td><td>Deliver the signal to the INTR signal of all processor cores listed in the destination as an interrupt that originated in an externally connected (8259A-compatible) interrupt controller. The INTA cycle that corresponds to this ExtINT delivery is routed to the external controller that is expected to supply the vector. A Delivery Mode of "ExtINT" requires an edge trigger mode.</td></tr>
</table>
</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 10h to FEh.</td></tr>
</table>

4.0. FUNCTIONAL DESCRIPTION

4.1. INTIN23/SMI# and SMIOUT# Functionality

The SMI# interrupt pin can be routed through the IOAPIC device. The redirection table entry can be programmed to send an SMI# interrupt over the APIC bus. The SMI# input is sent over the APIC bus when the delivery mode register is set to 010 for INTIN23/SMI# input and the Mask bit is set to 0. During this time, the SMIOUT# pin is inactive. If the Mask bit is set to 1, the INTIN23/SMI# input is routed to the SMIOUT# pin on the IOAPIC. Refer to Figure 3.

If the SMI# routing feature is not desired, the SMIOUT# pin is not connected to SMI# of the CPU's. The SMIOUT# pin is left open in the system. The INTIN23/SMI# redirection table entry has the same functionality as all other redirection table entries.

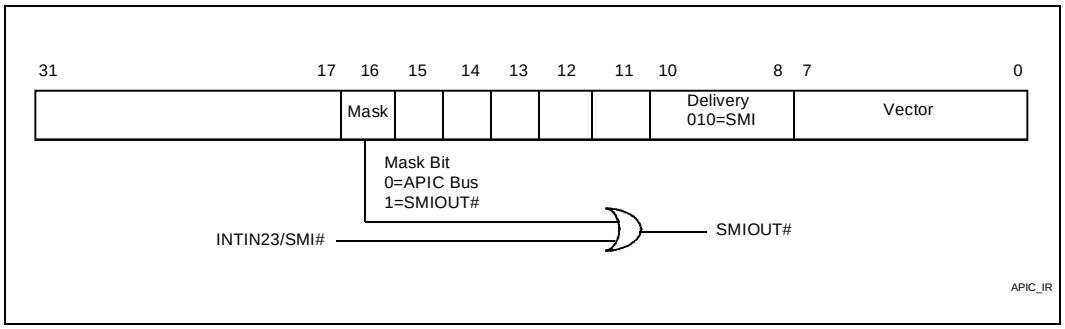

Figure 3. INTIN23/SMI# Routing And Control From Redirection Table Mask Bit

5.0. PINOUT AND PACKAGE SPECIFICATIONS

5.1. Pinout Specifications

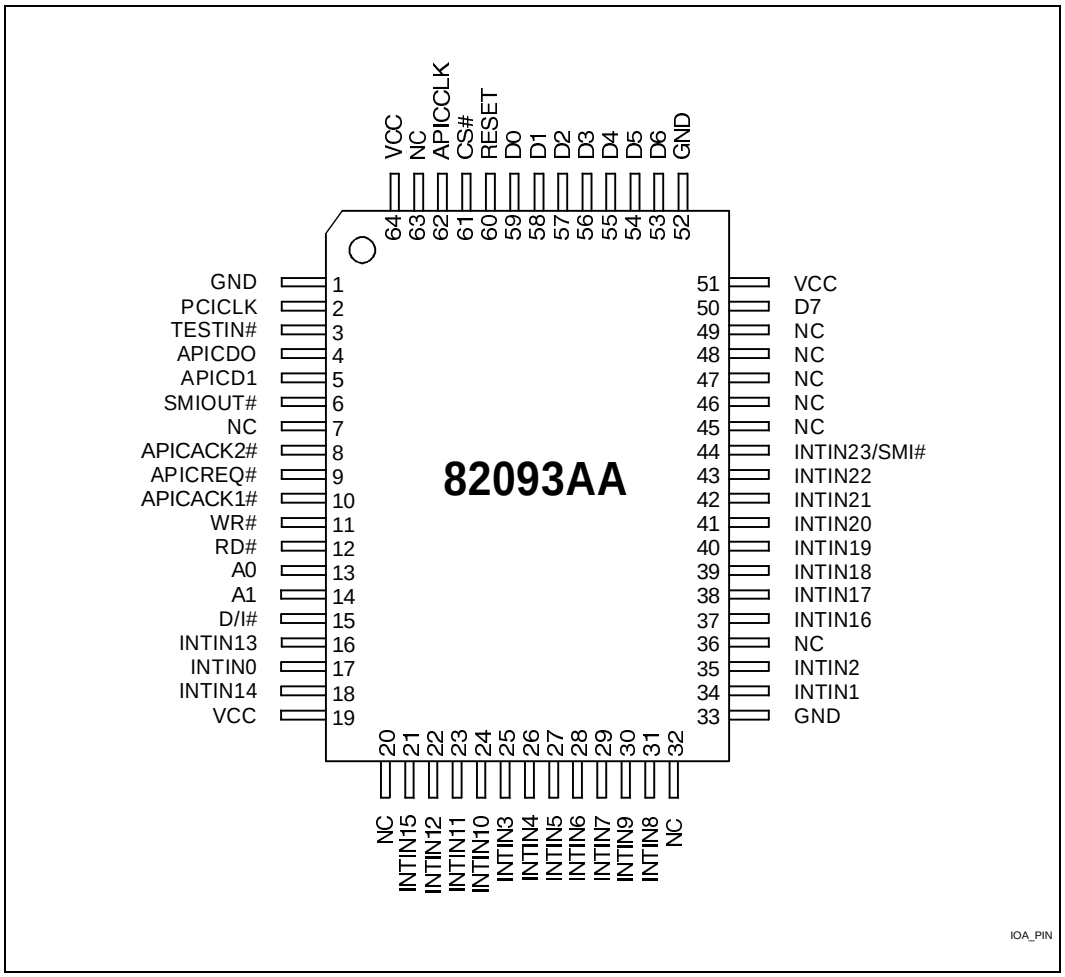

Figure 4. IOAPIC Pinout

Table 3. Alphabetical Pinout List

Name	Pin	Type
A0	13	I
A1	14	I
APICACK1#	10	I
APICACK2#	8	I
APICCLK	62	I
APICD0	4	I/OD
APICD1	5	I/OD
APICREQ#	9	O
CS#	61	I
D/I#	15	I
D0	59	I/O
D1	58	I/O
D2	57	I/O
D3	56	I/O
D4	55	I/O
D5	54	I/O
D6	53	I/O
D7	50	I/O
GND	1	V
GND	33	V
GND	52	V
INTIN0	17	ST

Table 3. Alphabetical Pinout List

Name	Pin	Type
INTIN1	34	ST
INTIN10	24	ST
INTIN11	23	ST
INTIN12	22	ST
INTIN13	16	ST
INTIN14	18	ST
INTIN15	21	ST
INTIN16	37	ST
INTIN17	38	ST
INTIN18	39	ST
INTIN19	40	ST
INTIN2	35	ST
INTIN20	41	ST
INTIN21	42	ST
INTIN22	43	ST
INTIN23/SMI#	44	ST
INTIN3	25	ST
INTIN4	26	ST
INTIN5	27	ST
INTIN6	28	ST
INTIN7	29	ST
INTIN8	31	ST

Table 3. Alphabetical Pinout List

Name	Pin	Type
INTIN9	30	ST
NC	7	—
NC	20	—
NC	32	—
NC	36	—
NC	45	—
NC	46	—
NC	47	—
NC	48	—
NC	49	—
NC	63	—
PCICLK	2	I
RD#	12	I
RESET	60	I
SMIOUT#	6	OD
TESTIN#	3	I
VCC	19	V
VCC	51	V
VCC	64	V
WR#	11	I

5.2. Package Specifications

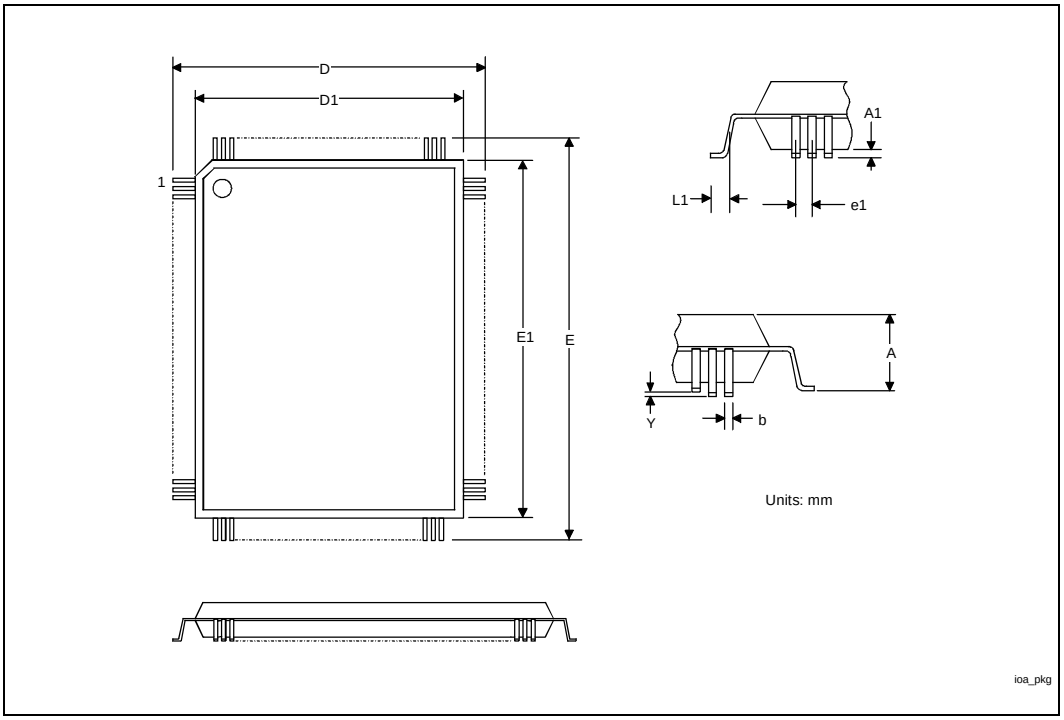

Figure 5. IOAPIC 64-Pin Quad Flat Pack (QFP)

Table 4. IOAPIC 64-Pin Quad Flat Pack (QFP)

Symbol	Description	Value mm (in.)	
		Min.	Max.
A	Seating Height		3.4 (0.134)
A1	Stand-off		0.25 (0.010)
b	Lead Width	0.35 (0.014)	0.50 (0.020)
D	Package Width, including pins	17.65 (0.695)	18.15 (0.715)
D1	Package Width, excluding pins	13.90 (0.547)	14.10 (0.555)
E	Package Length, including pins	23.65 (0.931)	24.15 (0.951)
E1	Package Length, excluding pins	19.90 (0.783)	20.10 (0.791)
e1	Linear Lead Pitch		1.0 (0.039)
Y	Lead Coplanarity		0.1 (0.004)
L1	Foot Length	0.65 (0.026)	1.15 (0.045)

6.0. TESTABILITY

The purpose of this section is to explain the test modes that the Stand Alone IOAPIC device will support. There are four test modes available in this device.

1. Tri-state all the outputs
2. Drive 1's to the outputs
3. Drive 0's to the outputs
4. Nand Tree

The Nand Tree and Tri-state mode are also valuable to the board testing environment. The Nand Tree can identify opens and shorts on input and output pins and the Tri-State mode can give the ATE tester control over other devices in the system.

The test modes are decoded from the INTIN inputs (INTIN21:17) qualified with the TESTIN# pin. Test mode selection is asynchronous. The following test modes are used:

Table 5. IOAPIC Test Modes

Test Modes	INTIN 21:17
NAND Tree	x1xxx xxx1x
Tristate All Outputs	00000
Drive Outputs High	00101
Drive Outputs Low	00100
Drive Outputs Low	00100

6.1. Tri-State Of All Output Pins

This mode is useful during ATE testing at a board level. If the IOAPIC tri-states its outputs, the tester can drive the signals without having to overdrive the IOAPIC output buffers. This mode is enabled in two ways:

1. Asserting the RESET pin. During RESET, all the IOAPIC device outputs are Tri-State
2. TESTIN# = 0 and INTIN(21:17) = 00000

6.2. Drive 1's to all the output pins

This is a special test mode that enables all the outputs are driving logic ones. How to enable this mode:

1. TESTIN# = 0
2. INTIN(21:17) = 00101
3. All the outputs will be driven to 1

6.3. Drive 0's to all the output pins

This is a special test mode that enables all the outputs into a low state. How to enable this mode:

1. TESTIN# = 0
2. INTIN(21:17) = 00100
3. All the outputs will be driven to 0

6.4. NAND Tree

The NAND tree is also useful for Automated Test Equipment (ATE) at board level testing. The NAND Tree allows the tester to test the solder connections for each individual signal pin. How to enable this mode:

1. All pins included in the NAND tree are driven to 1
2. TESTIN# = 0
3. INTIN(21:17) = x1x1x

Table 6. NAND Tree

Signal Name	Pin#	Comment	Nand tree Order
DATA_2	57	Start Nandtree	1
DATA_1	58		2
DATA_0	59		3
CSb	61		4
APICD_0	4		5
APICD_1	5		6
SMIOUTb	6		7
APICACK2b	8		8
APICREQb	9		9
APICACKb	10		10
WRb	11		11
RDb	12		12
SA_0	13		13
SA_1	14		14
DIb	15		15
INTIN_13	16		16
INTIN_0	17		17
INTIN_14	18		18
INTIN_15	21		19

Table 6. NAND Tree

Signal Name	Pin#	Comment	Nand tree Order
INTIN_12	22		20
INTIN_11	23		21
INTIN_10	24		22
INTIN_3	25		23
INTIN_4	26		24
INTIN_5	27		25
INTIN_6	28		26
INTIN_7	29		27
INTIN_9	30		28
INTIN_8	31		29
INTIN_1	34		30
INTIN_2	35		31
INTIN_16	37		32
INTIN_17	38		33
INTIN_18	39		34
INTIN_19	40		35
INTIN_20	41		36
INTIN_21	42		37
INTIN_22	43		38
INTIN_23	44		39
DATA_7	50		40
DATA_6	53		41
DATA_5	54		42
DATA_4	55		43
DATA_3	56	Nand tree Output	Output