Intel® 82806AA PCI 64 Hub (P64H)

Datasheet

March 2001

Document Number: 298025-002

Contents

Figures

Tables

Revision History

Rev.	Draft/Changes	Date
-001	• Initial Release	November 1999
-002	• Added Section 2.4.8, *Boot Configuration Register* • Modified the first sentence of the first paragraph of Section 3.2.6, *64-Bit Operation*	March 2001

This page is intentionally left blank.

Intel® 82806AA PCI 64 Hub (P64H)

Product Features

- PCI Interface
  - — Supports both 64-bit/32-bit 33 MHz or 66 MHz devices
  - — Provides Synchronous operation to hub interface bus using 1:1(66 MHz) or 2:1(33MHz) hub interface/ PCI Bus gearing ratio
  - — Allows I/O transactions to occur concurrently with processor transactions to isolate traffic
  - — Supports Parity and System Error (PERR#/ SERR#)
  - — Allows peer-to-peer communication within a single PCI bus segment
  - — Provides PCI transaction forwarding for all I/O and memory
  - — Provides address decoding for:
    - 16-bit I/O addressing
    - 32-bit non-prefetchable memory addressing
    - 44-bit prefetchable memory addressing (upstream only)
    - VGA addressing
  - — Includes downstream LOCK# capabilities
  - — Supports Fast Back-to-Back cycles (upstream only)
  - — Supports bus parking
  - — Implements Delayed Transaction for:
    - PCI configuration read/write,
    - I/O read, and Memory Read commands (downstream);
    - Memory Read, I/O Read and I/O Write commands (upstream)
- Scalability/ Flexibility
  - — Provides arbitration support for six PCI devices
  - — Supports either 2 x 66 MHz or 4 x 33 MHz PCI slots
  - — Processes dual address cycle (DAC) for upstream access >4GB
  - — Handles 3.3V operation with 5.0V tolerant on all input pins
- Upstream Hub Interface
  - — Connects to the MCH via a 16-bit hub interface
  - — Provides 64-bit and 32-bit addressing
  - — Utilizes 66 MHz base clock
- Integrated Functions
  - — I/O APIC to provide 24 interrupts
  - — Six copies of PCLKOUT signals to its PCI devices

The Intel® 82806AA PCI-64 Hub (P64H) is a multi-function PCI device that provides a PCI bridging function and an I/O Advanced Peripheral Interrupt Controller (APIC) function. The P64H is an integral part of the Intel® 840 chipset and future chipsets.

The P64H performs PCI bridging functions between the hub interface and the PCI Bus. The P64H has a 16-bit primary hub interface to the Memory Controller Hub (MCH) and a secondary 64-bit PCI Bus interface. This controller operates transparently with either 64-bit or 32-bit devices and supports either two 66 MHz or four 33 MHz PCI slots. The P64H also integrates I/O APIC controller functions. The interrupt controller provides up to 24 interrupts. In addition, the P64H provides 6 copies of the PCI clock.

Simplified Block Diagram

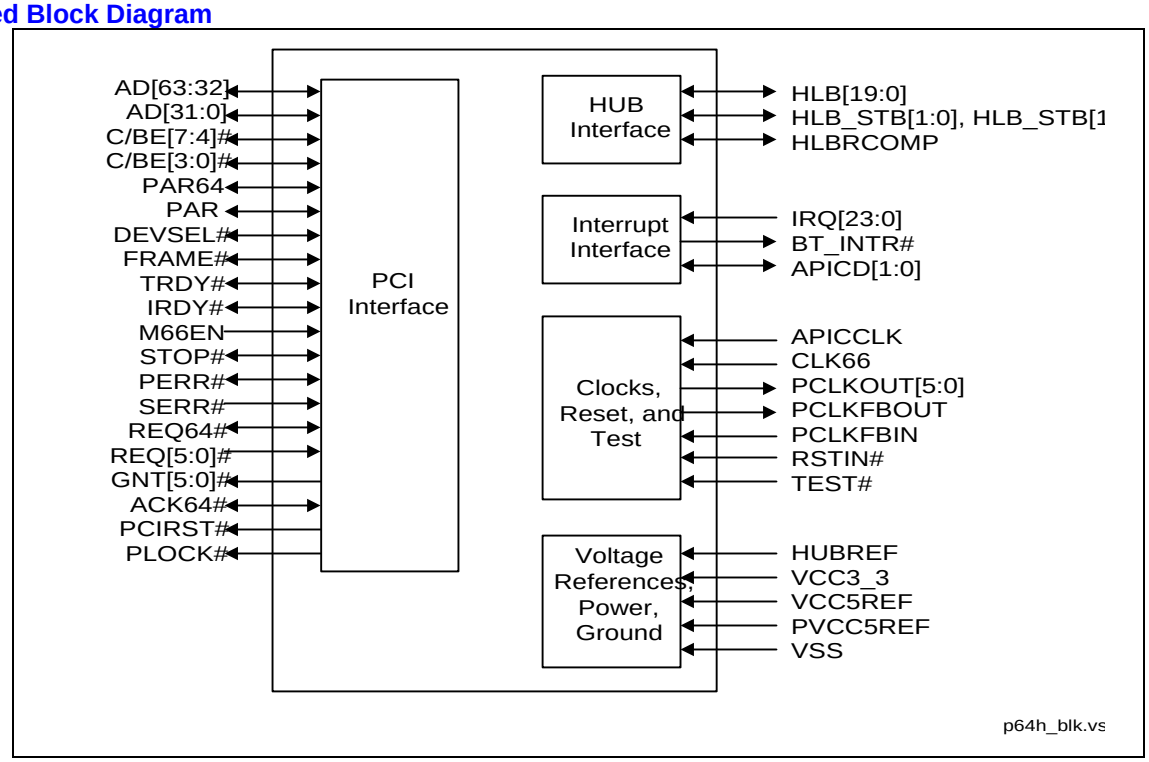
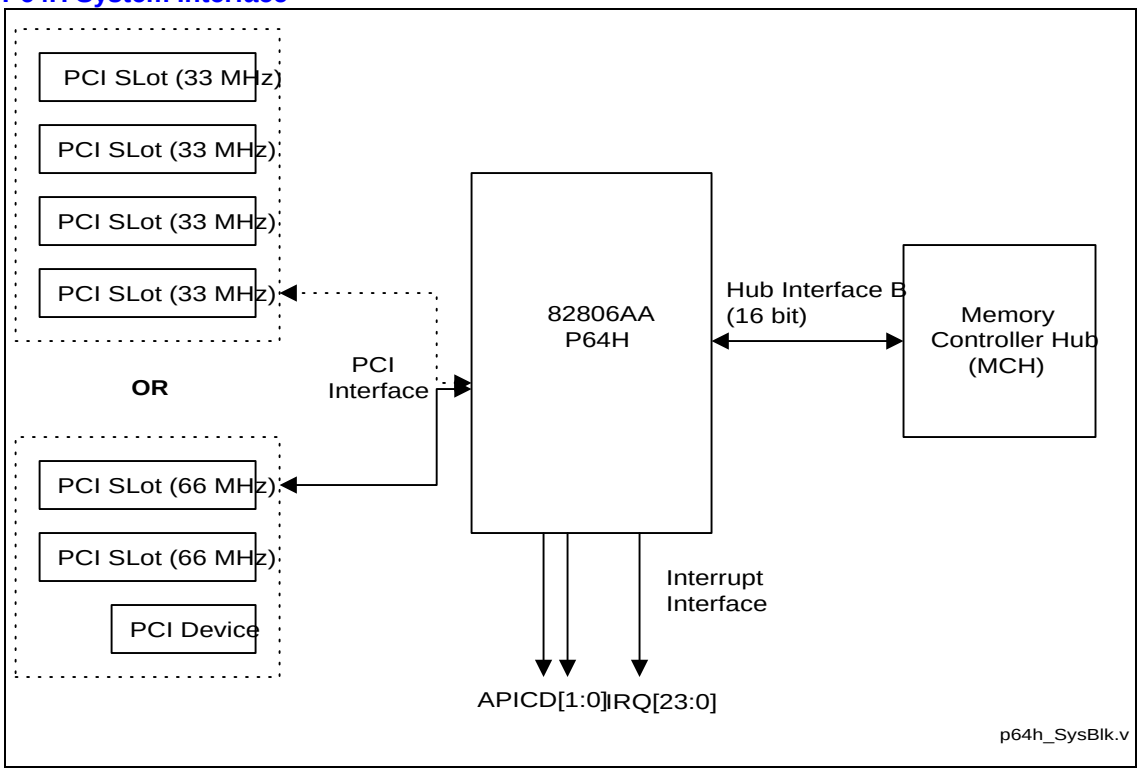


Figure 1. P64H System Interface



1. Signal Description

This section provides a detailed description of P64H signals. The signals are arranged in functional groups according to their associated interface. The states of all of the signals during reset are provided in Clocking and Reset, Section 3.6

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 following notations are used to describe the signal type:

I Input pin

O Output pin

OD Open Drain Output pin.

I/OD Input / Open Drain Output pin.

I/O Bi-directional Input/Output pin

1.1. Hub Interface to the Host Controller

Table 1. Hub Interface Signals

Name	Type	Description
HLB_STB[1:0]	I/O CMOS	**Hub Interface Packet Strobe**: This signal is a part of the differential strobe pair. This signal is used to transmit or receive the data on the hub interface.
HLB_STB[1:0]#	I/O CMOS	**Hub Interface Packet Strobe Complement**: This signal is a part of the differential strobe pair. This signal is used to transmit or receive the data on the hub interface.
HLBRCOMP	I/O CMOS	**Hub Interface B Resistive Compensation:** This signal is used to calibrate the hub interface B I/O buffers. The P64H only supports RCOMP. This signal must be tied to ground via a 30 Ω pull-down resistor.
HLB[19:0]	I/O CMOS	**Hub Interface Signals.**

1.2. PCI Interface

Table 2. PCI Interface Signals

Name	Type	Description
AD[63:32]	I/O	**PCI Address/Data:** AD[63:32] signals are multiplexed address and data bus. This provides an additional 32 bits to the PCI bus. During the address phase (when Dual Address command is used and REQ64# is asserted), the upper 32 bits are transferred. During the data phase, an additional 32 bit of data are transferred when REQ64# and ACK64# are both asserted. Unused AD[63:32] signals should be pulled up, to a valid logic level, through external resistors.
AD[31:0]	I/O	**PCI Address/Data:** AD[31:0] signals are multiplexed address and data bus. During the first clock of a transaction, AD[31:0] contains a physical byte address (32 bits). During the subsequent clocks, AD[31:0] contain data.
C/BE[7:4]#	I/O	**Bus Command and Byte Enables Upper 4 Bits:** C/BE[7:4]# signals are multiplexed command field and byte enable field. During the address phase (when the Dual Address command is used and REQ64# is asserted), the initiator will drive the transaction type on C/BE[7:4]#. Otherwise, these bits are undefined and the initiator drives a valid logic level onto the pins. During the data phase, the initiator will drive byte enables for the AD[63:32] data bits when REQ64# and ACK64# are both asserted. Unused C/BE[7:4]# signals should be pulled up, to a valid logic level, through external resistors.
C/BE[3:0]#	I/O	**Bus Command and Byte Enables:** These signals are a multiplexed command field and byte enable field. During the address phase of a transaction, C/BE[3:0]# define the Bus command. During the data phase, C/BE[3:0]# define the byte enables. **C/BE[3:0]# — Command Type** 0001 — Special Cycle 0010 — I/O Read 0011 — I/O Write 0110 — Memory Read 0111 — Memory Write 1010 — Configuration Read 1011 — Configuration Write 1100 — Memory Read Multiple 1101 — Dual Address Cycle 1110 — Memory Read Line 1111 — Memory Write and Invalidate All command encoding not shown are reserved. The P64H will not decode reserved values, and will not respond if a PCI master generates a cycle using one of the reserved values. NOTE: The Memory Read Multiple and Memory Read Line will be sent by the P64H on the hub interface as a Memory Reads. The memory write and invalidate will be sent by P64H on the hub interface as Memory Writes.
PAR64	I/O	**Upper 32-Bits Parity:** Signal PAR64 carries the even parity of AD[63:32] and C/BE[7:4]# for both address and data phases. Signal PAR64 is driven by the initiator and is valid one clock cycle after the first address phase when a Dual Address command is indicated by C/BE[3:0]# and REQ64# is asserted. Signal PAR64 is also valid one clock cycle after the second address phase of a dual address transaction when REQ64# is asserted. Once PAR64 is valid, it remains valid for one clock after the completion of the data phase. PAR64 share the same timing as AD[63:32] but delayed by one clock. If unused, PAR64 should be pulled up to a valid logic level through external resistors.

Name	Type	Description
PAR	I/O	**Parity:** PAR is “even” parity and is calculated on 36 bits - AD[31:0] plus C/BE[3:0]#. “Even” parity means that the sum of 1’s, within the 36 bits plus PAR, is always even. PAR is always calculated on 36 bits regardless of the valid byte enables. PAR is generated for address and data phases, and is only guaranteed to be valid one PCI clock after the corresponding address or data phase. PAR is driven and tri-stated identically to the AD[31:0] lines but PAR is delayed by one PCI clock. PAR is an output during the P64H address phase (delayed one clock) and data phase. The P64H checks parity when it is the initiator of read transaction and when it is the target of write transactions.
DEVSEL#	I/O	**Device Select:** The P64H asserts DEVSEL# to claim a PCI transaction. As an output (P64H Target), the P64H asserts DEVSEL# when a PCI master peripheral attempts an access to an internal P64H address or an address destined for hub interface (main memory or AGP). As an input (P64H Initiator), DEVSEL# indicates the response to a P64H-initiated transaction on the PCI bus. For P64H initiated transaction, it looks for the assertion of DEVSEL# within five cycles of FRAME# assertion; otherwise the P64H terminates the transaction with a master abort. DEVSEL# is tri-stated from the leading edge of PCIRST#. DEVSEL# remains tri-stated by the P64H until driven as a target.
FRAME#	I/O	**Frame:** The initiator drives FRAME# to indicate the beginning and duration of an access. Data transfers continues as long as FRAME# is asserted. When FRAME# is deasserted, this indicates the final data phase. FRAME# is an input when the P64H is the Target. FRAME# is an output when the P64H is the initiator. FRAME# remains tri-stated by the P64H until driven as an initiator.
IRDY#	I/O	**Initiator Ready:** IRDY# indicates the P64H’s ability, as an Initiator, to complete the current data phase of the transaction. It is used in conjunction with TRDY#. A data phase is completed on any clock when both IRDY# and TRDY# are sampled asserted. During a write, IRDY# indicates the P64H has valid data present on AD[31:0]. During a read, it indicates the P64H is prepared to latch data. IRDY# is an input to the P64H when the P64H is the Target and an output when the P64H is an Initiator. IRDY# remains tri-stated by the P64H until driven as an initiator.
TRDY#	I/O	**Target Ready:** TRDY# indicates the P64H’s ability to complete the current data phase of the transaction. TRDY# is used in conjunction with IRDY#. A data phase is completed when both TRDY# and IRDY# are sampled asserted. During a read, TRDY# indicates that the P64H (target) has placed valid data on its AD lines. During a write, it indicates that the P64H (target) is prepared to latch data. TRDY# is an input to the P64H when it is the initiator and an output when it is a target. TRDY# is tri-stated from the leading edge of PCIRST#. TRDY# remains tri-stated by the P64H until driven as a target.
M66EN	I	**66 MHz Enable:** This signal indicates the speed of the PCI Bus. If it is high then the Bus speed is 66 MHz and if it is low then the bus speed is 33 MHz.
STOP#	I/O	**Stop:** STOP# indicates that the P64H (target) is requesting the initiator to stop the current transaction. As an initiator, STOP# causes the P64H to stop the current transaction. STOP# is an output when the P64H is a Target and an input when the P64H is an Initiator. STOP# is tri-stated from the leading edge of PCIRST# and remains tri-stated until driven by P64H.
PERR#	I/O	**Parity Error:** The P64H drives PERR# when it detects a parity error during read/write transactions.
SERR#	I	**System Error:** SERR# can be pulsed active by any PCI device that detects a system error condition. The P64H samples SERR# as an input and conditionally forwards it to the hub interface.
REQ64#	I/O	**PCI Request, 64-Bit Transfer:** This signal indicates that the initiator is requesting a 64-bit data transfer. REQ64# has the same timing as FRAME#. When the P64H is the initiator, this signal is an output. When the P64H is the target, this signal is an input.
REQ[5:0]#	I	**PCI Requests:** Supports up to 6 masters on the PCI bus. The P64H accepts six request inputs into its internal bus arbiter.

Name	Type	Description
GNT[5:0]#	O	**PCI Grants:** Supports up to six masters on the PCI bus. The arbiter can assert one of the six bus grant outputs, to indicate that an initiator can start a transaction on the PCI Bus.
ACK64#	I/O	**Acknowledge, 64-Bit Transfer:** Signal ACK64# is asserted by target only when REQ64# is asserted by the initiator, to indicate the target's ability to transfer data using 64 bits. ACK64# has the same timing as DEVSEL#.
PCIRST#	O	**PCI Reset:** P64H asserts PCIRST# to reset devices that reside on its PCI bus. PCIRST# occurs when either RSTIN# is asserted or PCI reset bit (BRIDGE_CNT register) is set.
PLOCK#	O	**PCI Lock:** Indicates exclusive bus operation and may require multiple transactions to complete. P64H asserts PLOCK# when it performs exclusive transactions on PCI. PLOCK# is ignored when PCI masters are granted the bus. The P64H does not propagate locked transaction upstream (hub interface).

1.3. Interrupt Interface

Table 3. Interrupt Interface Signals

Name	Type	Description
IRQ[23:0]	I	**Interrupt Request [23:0]:** These are APIC interrupts and do not support (8259) PIC mode. External 8.2K pull-up resistors are required if these signals are not used.
BT_INTR#	O/OD	**Boot Interrupt:** This open drain signal should be tied to a ICH PIRQx to support boot devices on the P64H PCI segment. Upon power reset, P64H will forward the interrupt request to the ICH via this signal. This pin will automatically be disabled after the I/O APIC ID register is written by the OS.
APICD[1:0]	I/OD	**APIC Data:** These bi-directional open drain signals are used to send and receive data over the APIC bus. As inputs the data is valid on the rising edge of APICCLK. As outputs, new data is driven from the rising edge of the APICCLK. External 8.2K pull-up resistors are required if these signals are not used.

1.4. Clocks

Table 4. Clock Signals

Name	Type	Description
APICCLK	I	**APIC Clock:** This clock is 16.66667 MHz and is the APIC bus clock. The clock frequency can be raised, up to 33 MHz, to support FRC mode on dual processor systems. External 8.2K pull-up resistors are required if these signals are not used.
CLK66	I	**66 MHz Clock:** Used to run the hub interface.
PCLKOUT[5:0]	O	**PCI Clock:** 33/66 MHz clock. PCICLK provides timing for all transactions on the PCI bus. These pins can be disabled via the P64H Configuration register. The default state is enabled.
PCLKFBOUT	O	**PCI Feedback Clock Out:** This signal synchronizes the PCI clocks to the internal PLL. This signal should be fed into the P64H PCLKFBIN pin
PCLKFBIN	I	**PCI Feedback Clock In:** This signal is used by the PLL to synchronize the PCI clock driven from PCLKFBOUT to the clock used for the internal PCI logic.

1.5. Miscellaneous

Table 5. Miscellaneous Signals

Name	Type	Description
RSTIN#	I	**Reset In:** When asserted this signal will asynchronously reset the P64H logic.
TEST#	I	**Test:** This pin is used for manufacturing testing only. This signal contains an internal pull-up resistor. Externally, this pin must be pulled-up to Vcc3.3.
RSV[15:1]	I/O	**Reserved:** RSV1 and RSV3 must be pull-down to GND. Other RSV signals should not be used and left as N/C.

1.6. Power and Ground Signals

Table 6. Power and Ground Signals

Name	Type	Description
HUBREF	I	**Hub interface Reference:** Reference voltage input for the hub interface. This will be turned-off in power management states.
VCC3_3	I	**3.3V Core Voltage:** This will be turned-off in power management states.
VCC5REF	I	**Vcc 5V Reference Voltage:** The VCC5REF is the reference voltage for the IRQ signals. This will be turned-off in power management states. VCC5REF at ball P7 is used for RSTIN#. VCC5REF at ball H17 is used for the IRQ's, TEST#, and BT_INT#.
PVCC5REF	I	**PCI Vcc 5V Reference Voltage:** The PVCC5REF is the reference voltage for PCI signals (excluding PCICLK signals).
VSS	I	**Ground:** Grounds for periphery and hub interface.

This page is intentionally left blank.

2. Register Description

The P64H contains two PCI Devices—Hub Interface-to-PCI Bridge (Device 31 : Function 0) and an I/O APIC device (Device 0 : Function 0). This chapter describes these P64H configuration registers. A detailed bit description is provided.

- **Hub Interface-to-PCI Bridge (D31:F0).** This portion of the P64H implements the buffering and control logic between PCI and the hub interface. The PCI bus arbitration is handled by this PCI device. The PCI decoder in this device must decode the ranges for hub interface to the MCH. All register contents are lost when core well power is removed.
- **APIC device (D0:F0).** The P64H implements a variation of the APIC known as the I/O APIC. The I/O APIC reside on the secondary PCI bus (bus #1).

The P64H supports PCI configuration space accesses using the mechanism denoted as Configuration Mechanism #1 in the *PCI specification*.

2.1. Register Nomenclature and Access Attributes

Symbol	Description
RO	Read Only. If a register is read only, writes to this register have no effect.
WO	Write Only. If a register is write only, reads of this register return undefined results..
R/W	Read/Write. A register with this attribute can be read and written
R/WC	Read/Write Clear. A register bit with this attribute can be read and written. However, a write of a 1 clears (sets to 0) the corresponding bit and a write of a 0 has no effect.
R/WO	Read/Write Once. A register bit with this attribute can be written to only once after power up. After the first write, the bit becomes read only.
Reserved Bits	Software must deal correctly with fields that are reserved. On reads, software must use appropriate masks to extract the defined bits and not rely on reserved bits being any particular value. On writes, software must ensure that the values of reserved bit positions are preserved. That is, the values of reserved bit positions must first be read, merged with the new values for other bit positions and then written back. Note the software does not need to perform read, merge, write operation for the configuration address register.
Reserved Registers	In addition to reserved bits within a register, the P64H contains address locations in its configuration space that are marked "Reserved. When a “Reserved” register location is read, a random value can be returned. (“Reserved” registers can be 8-, 16-, or 32-bit in size). Registers that are marked as “Reserved” must not be modified by system software. Writes to “Reserved” registers may cause system failure.
Default Value Upon Reset	Upon a Full Reset, the P64H sets all of its internal configuration registers to predetermined **default** states.

2.2. Hub Interface to PCI Bridge Registers (D31:F0)

The hub interface to PCI Bridge resides in PCI Device 31, Function 0 on the PCI bus.

Note: Reserved registers are read only and are not shown. Reads to these registers return all 0s.

Table 7. PCI Configuration Map (Hub Interface-PCI—D30:F0)

Address Offset	Mnemonic	Register Name	Default	Type
00–01	VID	Vendor ID	8086h	RO
02–03	DID	Device ID	1360h	RO
04–05	CMD	Device Command Register	0000h	R/W
06–07	PD_STS	Device Status Register	00A0h	R/WC
08	REVID	Revision ID	See latest Specification Update	RO
09	PIF	Programming Interface	00h	RO
0A	SCC	Sub Class Code	04h	RO
0B	BCC	Base Class Code	06h	RO
0C	CLS	Cache Line Size Register	00h	RO
0D	PMLT	Master Latency Timer	00h	RO
0E	HEADTYP	Header Type	80h	RO
18	PBUS_NUM	Primary Bus Number	00h	R/W
19	SBUS_NUM	Secondary Bus Number	00h	R/W
1A	SUB_BUS_NUM	Subordinate Bus Number	00h	R/W
1B	SMLT	Secondary Master Latency Timer	00h	R/W
1C	IOBASE	IO Base Address Register	00h	R/W
1D	IOLIM	IO Limit Address Register	00h	R/W
1E–1F	SECSTS	Secondary Status Register	02A0h	R/W
20–21	MEMBASE	Non-Prefetchable Memory Base Address	0000h	R/W
22–23	MEMLIM	Non-Prefetchable Memory Limit Address	0000h	R/W
24–25	PREF_MEM_BASE	Prefetchable Memory Base Address	0000h	R/W
26–27	PREF_MEM_LIM	Prefetchable Memory Limit Address	0000h	R/W
28–2B	PREF_MEM_BASE UPPER	Prefetchable Memory Base Upper 32-Bit Address	00000000h	R/W
2C–2F	PREF_MEM_LIM UPPER	Prefetchable Memory Limit Upper 32-Bit Address	00000000h	R/W
30–31	IOBASE_HI	I/O Base Upper 16-Bit Address	0000h	RO
32–33	IOBASE_LO	I/O Limit Upper 16-Bit Address	0000h	RO
3C–3D	INT_LINE	Interrupt Line	00h	RO
3E–3F	BRIDGE_CNT	Bridge Control	0000h	R/W

Address Offset	Mnemonic	Register Name	Default	Type
40–43	—	Reserved	00006801	R/W
50–51	CNF	P64H Configuration Register	0000h	R/W
70	MTT	Multi-Transaction Timer	00h	R/W
72	—	Reserved	00	RO
80	DTT	Delay Transaction Timer	00h	R/W
90	ERR_CMD	Error Command Register	00h	R/W
92	ERR_STS	Error Status Register	00h	R/W

2.2.1. VID—Vendor ID Register (D31:F0)

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

Bit	Description
15:0	**Vendor Identification Number:** This 16-bit value is assigned to Intel. VID=8086h.

2.2.2. DID—Device ID (D31:F0)

Address Offset: 02–03h
Default Value: 1360h
Attribute: RO
Size: 16 bits

Bit	Description
15:0	**Device Identification Number:** This is a 16-bit value assigned to the P64H hub interface to PCI bridge. ID =1360.

2.2.3. CMD—Device Command (D31:F0)

Address Offset: 04–05h
Default Value: 0000h
Attribute: R/W
Size: 16 bits

Bit	Description
15:10	Reserved
9	**Fast Back to Back Enable (FBE)—RO:** This bit indicates whether the P64H is allowed to run cycles originating from hub interface B in fast back-to-back mode. The P64H does not support this capability.
8	**SERR# Enable (SERR_EN)—R/W:** Controls SERR# enable on the hub interface.
7	**Wait Cycle Control—RO:** Hardwired to 0.
6	**Parity Error Response—R/W:** Controls the P64H's response when a parity error is detected on the hub interface. 1 = Enable. P64H reports parity errors on the hub interface B. 0 = Disable. P64H ignores parity errors on the hub interface B.
5	**VGA Palette Snoop Enable—RO:** Hardwired to 0.
4	**Memory Write and Invalidate Enable (MWIE)—RO:** Hardwired to 0.
3	**Special Cycle Enable (SCE)—RO:** Hardwired to 0 by P2P Bridge specification.
2	**Bus Master Enable (BME)—R/W:** Controls the P64H's ability to initiate memory transactions on the hub interface on behalf of an initiator on the PCI bus. 1 = Enable. P64H accepts cycles from PCI to be passed to the hub interface. 0 = Disable. P64H does not respond to Memory or I/O transactions on the PCI bus and does not initiate I/O or memory transactions on the hub interface.
1	**Memory Space Enable (MSE)—R/W:** Controls the P64H's response to memory transactions on the hub interface. 1 = Enable. P64H responds to memory transactions initiated on the hub interface. 0 = Disable. P64H does not respond to memory transactions initiated on the hub interface.
0	**I/O Enable (IOE)—R/W:** Controls the P64H's response to I/O transactions on the hub interface. 1 = Enable. P64H responds to I/O transaction initiated on the hub interface. 0 = Disable. P64H does not respond to I/O transaction initiated on the hub interface.

2.2.4. PD_STS—Primary Device Status Register (D31:F0)

Address Offset: 06–07h
Default Value: 0020h
Attribute: R/WC
Size: 16 bits

For the writeable bits in this register, writing a 1 clears the bit. Writing a 0 has no effect.

Bit	Description
15	**Detected Parity Error (DPE)—R/WC:** 1 = Indicates that the P64H detected a address or data parity error on the hub interface. This bit will be set even if the parity error response bit is not set. 0 = The bit is cleared by writing a 1 to the location.
14	**Signaled System Error (SSE)—R/WC:** 1 = SERR# cycle is initiated by the P64H to the hub interface. 0 = The bit is cleared by writing a 1 to the location.
13	**Received Master Abort (RMA)—R/WC:** 1 = P64H as a master on the hub interface receives a master abort from the MCH. 0 = The bit is cleared by writing a 1 to the location.
12	**Received Target Abort (RTA)—R/WC:** 1 = P64H as a master on the hub interface receives a target abort from the MCH. This bit can be used to generate an internal SERR# (see the ERR_CMD register). 0 = The bit is cleared by writing a 1 to the location.
11	**Signaled Target Abort (STA)—R/WC:** 1 = P64H, as the target on the hub interface, signals a target abort. 0 = The bit is cleared by writing a 1 to the location.
10:9	Reserved
8	**Data Parity Error Detected (DPD)—R/WC:** With the PERR signal removed from the hub interface, the P64H must interpret this bit differently than it is in the PCI specification. 1 = This bit is set to 1 when the parity error response bit (CMD register) is set to 1 and the P64H detects a parity error as a hub interface initiator. 0 = The bit is cleared by writing a 1 to the location.
7	**Fast Back to Back—RO:** Hardwired to 0.
6:0	Reserved

2.2.5. REVID—Revision ID (D31:F0)

Address Offset: 08h
Default Value: See latest Specification Update
Attribute: RO
Size: 8 bits

Bit	Description
7:0	**Revision Identification Number:** Indicates the revision number for the P64H.

2.2.6. PIF – Programming Interface

Address Offset: 09h
Default Value: 00h
Attribute: RO
Size: 8 bits

Bit	Description
7:0	**Programming Interface:** Indicates that this is a standard (non-subtractive) PCI-PCI bridge.

2.2.7. SCC—Sub-Class Code Register (D31:F0)

Address Offset: 0Ah
Default Value: 04h
Attribute: RO
Size: 8 bits

Bit	Description
7:0	**Sub-Class Code:** Indicates the category of bridge for the P64H. 04h = PCI-to-PCI bridge.

2.2.8. BCC—Base-Class Code Register (D31:F0)

Address Offset: 0Bh
Default Value: 06h
Attribute: RO
Size: 8 bits

Bit	Description
7:0	**Base Class Code:** Indicates the P64H device type. 06h = Bridge device.

2.2.9. CLS—Cache Line Size Register (D31:F0)

Address Offset: 0Ch
Default Value: 00h
Attribute: RO
Size: 8 bits

Bit	Description
7:0	**Cache Line Size Register:** The P64H does not perform memory write and invalid.

2.2.10. PMLT—Primary Master Latency Timer (D31:F0)

Address Offset: 0Dh
Default Value: 00h
Attribute: R/W
Size: 8 bits

Bit	Description
7:3	**Timer Value:** This is not implemented.
2:0	Reserved

2.2.11. HEADTYP—Header Type Register (D31:F0)

Address Offset: 0Eh
Default Value: 01h
Attribute: RO
Size: 8 bits

Bit	Description
7	**Multi-function Device:** Hardwired to 0.
6:0	**Header Type:** Identifies the header layout of the configuration space. 01h = Register layout conforms to the typical PCI-to-PCI bridge layout.

2.2.12. PBUS_NUM—Primary Bus Number Register (D31:F0)

Address Offset: 18h
Default Value: 00h
Attribute: R/W
Size: 8 bits

Bit	Description
7:0	**Primary Bus Number:** This field indicates the hub interface bus number and the value is programmed by configuration software.

2.2.13. SBUS_NUM—Secondary Bus Number Register (D31:F0)

Address Offset: 19h
Default Value: 00h
Attribute: R/W
Size: 8 bits

Bit	Description
7:0	**Secondary Bus Number:** This field indicates the PCI bus number of the PCI bus connected to the secondary interface. The P64H uses this register to determine when to respond or forward Type 1 configuration transactions on the hub interface. This field is also used to determine when to convert Type 1 configuration transaction to type 0 or special cycle transactions on the PCI bus.

2.2.14. SUB_BUS_NUM—Subordinate Bus Number Register (D31:F0)

Address Offset: 1Ah
Default Value: 00h
Attribute: R/W
Size: 8 bits

Bit	Description
7:0	**Subordinate Bus Number:** This field specifies the highest PCI bus number below the hub interface-to-PCI bridge. If a Type 1 configuration cycle from hub interface does not fall in either the Secondary or Subordinate Bus range of Device 31, the P64H will perform a master abort (all 1's) back to the hub interface.

2.2.15. SMLT—Secondary Master Latency Timer (D31:F0)

Address Offset: 1Bh
Default Value: 00h
Attribute: R/W
Size: 8 bits

This Master Latency Timer (MLT) controls the amount of time that the P64H will continue to burst data as a master on the PCI bus. When the P64H starts the cycle after being granted the bus, the counter is loaded and starts counting down from the assertion of FRAME#. If the internal grant to this device is removed, the expiration of the MLT counter results in the deassertion of FRAME#. Otherwise, the P64H will continue to own the bus.

Bit	Description
7:3	**Secondary Latency Count:** This 5-bit field indicates the number of PCI clocks, in 8-clock increments, that the P64H will remain as master of the bus.
2:0	Reserved

2.2.16. IOBASE—I/O Base Register (D31:F0)

Address Offset: 1Ch
Default Value: 00h
Attribute: RO, R/W
Size: 8 bits

Bit	Description
7:4	**I/O Address Base Bits [15:12]—R/W:** This field defines the base address of when the P64H should forward the I/O transaction. This field in conjuction with the IOLIM Register define a window where P64H forwards I/O transactions.The bits in this field correspond to address lines [15:12] for a 4 KB alignment. Address base bits [11:0] are assumed to be 000h.
3:0	**I/O Addressing Capability—RO:** Hardwired to 0. This bit indicates the hub interface-to-PCI bridge I/O addressing is 16 bit only.

2.2.17. IOLIM—I/O Limit Register (D31:F0)

Address Offset: 1Dh
Default Value: 00h
Attribute: RO, R/W
Size: 8 bits

Bit	Description
7:4	**I/O Address Limit Bits [15:12]—R/W:** This field defines the upper boundary address limit when the P64H should forward the I/O transaction. This field is used in conjuction with the IOBASE register to define a window where P64H should forward the I/O transaction. I/O Base bits corresponding to address lines [15:12] for a 4-KB alignment. Address limit bits [11:0] are assumed to be FFFh.
3:0	**I/O Addressing Capability—RO:** Hardwired to 0. This bit indicates the hub interface-to-PCI bridge I/O addressing is 16 bit only.

2.2.18. SECSTS—Secondary Status Register (D31:F0)

Address Offset: 1E–1Fh
Default Value: 02A0h
Attribute: RO, R/WC
Size: 16 bits

For the writeable bits in this register, writing a 1 clears the bit. Writing a 0 has no effect.

Bit	Description
15	**Detected Parity Error (DPE)—R/WC:** 1 = P64H detects an address or data parity error on the PCI bus. This bit will be set, even if the parity error response enable bit (BRIDGE_CNT register) is not set. 0 = The bit is cleared by writing a 1 to the location.
14	**Signal System Error (SSE)—R/WC:** 1 = P64H detected a SERR# on the PCI bus. 0 = The bit is cleared by writing a 1 to the location.
13	**Received Master Abort (RMA)—R/WC:** 1 = Hub interface-to-PCI cycle is master aborted on PCI. 0 = The bit is cleared by writing a 1 to the location.
12	**Received Target Abort (RTA)—R/WC:** 1 = Hub interface-to-PCI cycle is target aborted on PCI. For “completion required” cycles from the hub interface, this event should also set the Signaled Target Abort in the Primary Status Register of this device, and the P64H must send the “target abort” status back to the hub interface. 0 = The bit is cleared by writing a 1 to the location.
11	**Signaled Target Abort (STA)—R/WC:** 1 = P64H does not generate target aborts and does not forward target abort response from the hub interface to the PCI bus. 0 = The bit is cleared by writing a 1 to the location.
10:9	**DEVSEL# Timing Status—RO:** Indicates slowest response to a command on the secondary interface. 01h = Medium timing.
8	**Data Parity Error Detected—R/W:** 1 = The P64H sets this bit to 1 when all of the following conditions are true: • Hub interface-to-PCI bridge is the initiator on the PCI bus. • PERR# is detected asserted or a parity error is detected internally. • Parity error response enable bit (BRIDGE_CNT register) is set.
7	**Fast Back to Back—RO:** Hardwired to 0.
6	Reserved
5	**66 MHz Capable—RO**. This bit is hardwired to 1 to indicate that it is 66 MHz capable.
4:0	Reserved

2.2.19. MEMBASE—Non-Prefetchable Memory Base Address Register (D31:F0)

Address Offset: 20–21h
Default Value: 0000h
Attribute: R/W
Size: 16 bits

This register defines the base of the hub interface-to-PCI non-prefetchable memory range. This register must be initialized by configuration software. For the purpose of address decode, address bits A[19:0] are assumed to be 0. Thus, the bottom of the defined memory address range will be aligned to a 1 MB boundary.

Bit	Description
15:4	**Memory Address Base:** Defines the base of the memory range for PCI. This field in conjuction with the MEMLIM register define a memory window for PCI. These 12 bits correspond to address bits [31:20].
3:0	Reserved

2.2.20. MEMLIM—Non-Prefetchable Memory Limit Address Register (D31:F0)

Address Offset: 22–23h
Default Value: 0000h
Attribute: R/W
Size: 16 bits

This register defines the upper limit of the HL-to-PCI non-prefetchable memory range. This register must be initialized by configuration software. For the purpose of address decode, address bits A[19:0] are assumed to be FFFFFh. Thus, the top of the defined memory address range will be aligned to a 1-B boundary.

Bit	Description
15:4	**Memory Address Limit:** Defines the top of the memory range for PCI. This field in conjuction with the MEMBASE register define a memory window for PCI. These 12 bits correspond to address bits [31:20].
3:0	Reserved

2.2.21. PREF_MEM_BASE—Prefetchable Memory Base Address Register (D31:F0)

Address Offset: 24–25h
Default Value: 0000h
Attribute: R/W
Size: 16 bits

This register must be initialized by configuration software. These bits are compared to bits 31:20 of the incoming address to determine the lower 1-B range.

Bit	Description
15:4	**Prefetchable Memory Base Address [31:20]—R/W:** These bits define the bottom address, of the address range, used by P64H to determine when to forward memory read and write transactions. The upper 12 bits are writeable and correspond to address bits [31:20]. The lower 20 bits of the address are assumed to be 0. The memory base register upper 32 bits contain the upper half of the base address. The memory address range adheres to the 1-B alignment and granularity. This field in conjuction with the PREF_MEM_MLT register define a memory window for prefetchable memory addresses.
3:0	**64-Bit Indicator—RO:** These lower four bits indicates that 32-bit addressing is supported.

2.2.22. PREF_MEM_LIM—Prefetchable Memory Limit Address Register (D31:F0)

Address Offset: 26–27h
Default Value: 0000h
Attribute: R/W
Size: 16 bits

These bits are compared to bits 31:20 of the incoming address to determine the upper 1-MB range.

Bit	Description
15:4	**Prefetchable Memory Limit Address [31:20]—R/W:** Defines the top address, of an address range, used by P64H to determine when to forward memory read and write transactions. The upper 12 bits are writeable and correspond to address bits [31:20]. The lower 20 bits of the address are assumed to be FFFFFh. The memory limit upper 32-bit register contain the upper half of the limit address. The memory address range adheres to the 1-MB alignment and granularity.
3:0	**64-Bit Indicator—RO:** These lower four bits indicates that 32-bit addressing is supported.

2.2.23. PREF_MEM_BASE_UPPER—Prefetchable Memory Base Upper 32-Bit Address Register (D31:F0)

Address Offset: 28h
Default Value: 00000000h
Attribute: R/W
Size: 32 bits

This register must be initialized by configuration software.

Bit	Description
31:12	Reserved
11:0	**Prefetchable Memory Base Upper Portion**. The P64H only supports 44-bit addressing. Only bits [11:0] are writeable.

2.2.24. PREF_MEM_LIM_UPPER—Prefetchable Memory Base Upper 32-Bit Address Register (D31:F0)

Address Offset: 2Ch
Default Value: 00000000h
Attribute: R/W
Size: 32 bits

This register must be initialized by configuration software.

Bit	Description
31:12	**Reserved**
11:0	**Prefetchable Memory Limit Upper Portion:** The P64H only supports 44-bit addressing. Only bits [11:0] are writeable.

2.2.25. IOBASE_HI—I/O Base Upper 16-Bit Address Register (D31:F0)

Address Offset: 30–31h
Default Value: 0000h
Attribute: RO
Size: 16 bits

Bit	Description
15:0	**I/O Address Base Upper 16 Bits [31:16]:** Hardwired to 0.

2.2.26. IOBASE_LO—I/O Limit Upper 16-Bit Address Register (D31:F0)

Address Offset: 32–33h
Default Value: 0000h
Attribute: RO
Size: 16 bits

Bit	Type	Description
15:0	RO	**I/O Address Limit Upper 16 Bits [31:16]:** Hardwired to 0.

2.2.27. INT_LINE—Interrupt Line Register (D31:F0)

Address Offset: 3Ch
Default Value: 00h
Attribute: RO
Size: 8 bits

Bit	Type	Description
7:0	RO	**Interrupt Line Routing:** Hardwired to 0 to indicate that the bridge does not generate interrupts.

2.2.28. BRIDGE_CNT—Bridge Control Register (D31:F0)

Address Offset: 3E–3Fh
Default Value: 0000h
Attribute: R/W
Size: 16 bits

Bit	Description
15	**PCI Bus Frequency—RO**. The P64H determines the bus operating frequency and sets this bit. 1= 66 MHz PCI bus 0= 33 MHz PCI bus
14:12	Reserved.
11	**PCI Master Time Out SERR# Enable—R/W:** This bit controls the SERR# special cycle over the hub interface. 1= Enable. SERR# special cycle is generated when PCI master time out bit is set. 0= Disable. SERR# special cycle is not generated when the PCI master time out bit is set.
10	**PCI Master Timeout Status—R/WC:** 1 = PCI master time out counter expired and a delayed transaction is discarded. Hub interface SERR# special cycle is generated when this bit, PCI Master Timeout SERR# enable bit (bit 11), and SERR_EN bit are set. 0 = Writing a 1 will clear this bit to 0.

Bit	Description
9	**PCI Master Timeout— R/W:** This bit sets the maximum number of PCI clock cycles that the P64H will wait for the hub interface initiator to repeat a delayed transaction request. The counter starts when the delayed transaction completion is at the head of the queue. If the master does not repeat the transaction before the counter expires, the P64H discards the transaction from its queues. 0= PCI master time out value is between 2^{15} and 2^{16} PCI clocks. 1= PCI master time out value is between 2^{10} and $2^{11.}$
8	**Hub Interface Master Timeout— R/W:** This bit is read/write for software compatibility only. Changes to this bit have no effects.
7	**Fast Back to Back Enable— RO:** The P64H does not generate fast back-to-back transactions on the PCI bus. This bit is read as 0.
6	**Secondary Bus Reset— R/W:** Controls the reset on the PCI Bus. 1 = The P64H asserts PCIRST# 0 = The P64H deasserts PCIRST#.
5	**Master Abort Mode— R/W:** This bit controls the P64H behavior upon a master abort by the P64H on the PCI bus or to hub interface. When master abort occurs on the hub interface: 1 = The P64H returns a target abort on the PCI bus. 0 = The P64H asserts TRDY# on the PCI bus for delayed transaction, drives all 1 for read transaction, and discard all data on write transaction. When master abort occurs on the PCI bus due to a hub interface initiated transaction: 1 = A target abort packet will be sent to the hub interface. 0 = A normal completion packet will be sent to the hub interface master.
4	Reserved.
3	**VGA Enable— R/W:** 1 = Enable. Indicates that the VGA device is on PCI. The P64H positively decodes and forwards memory (A0000h–BFFFFh) and I/O (3B0h–3BBh, 3C0h–3DFh) transactions downstream, regardless of the values in the I/O Base and Limit registers. 0 = Disable. the P64H will not forward VGA compatible memory and I/O addressing from the hub interface to the PCI bus unless specified by the memory and I/O address range. Memory (A0000h–BFFFFh) and I/O (3B0h–3BBh, 3C0h–3DFh) transactions are instead sent from the PCI bus to the hub interface.
2	**ISA Enable— R/W:** This bit is read/write for software compatibility. Changes to this bit have no effects.
1	**SERR# Enable— R/W:** Controls the P64H SERR# special cycle forwarding to the hub interface. 1 = Enable. P64H generates SERR# special cycle when SERR# is detected on the PCI bus. 0 = Disable. P64H does not generate SERR# special cycle when SERR# is detected.
0	**Parity Error Response Enable— R/W:** Controls the P64H's response when a parity error is detected on the PCI bus. 1 = Enable. The P64H drives PERR# and will set the data parity detected bit in the secondary status register, if data parity is detected on the PCI bus. 0 = Disable. The P64H does not assert PERR# and will not set the data parity detected bit in the secondary status register.

2.2.29. CNF—P64H Configuration Register (D31:F0)

Address Offset: 50–51h
Default Value: 0000h
Attribute: R/W
Size: 16 bits

Bit	Description
15	**PCLKOUT5 Disable:** 1 = Disable. Clock output is disabled and the output tri-stated. 0 = Enable.
14	**PCLKOUT4Disable:** 1 = Disable. Clock output is disabled and the output tri-stated. 0 = Enable.
13	**PCLKOUT3 Disable:** 1 = Disable. Clock output is disabled and the output tri-stated. 0 = Enable.
12	**PCLKOUT2 Disable:** 1 = Disable. Clock output is disabled and the output tri-stated. 0 = Enable.
11	**PCLKOUT1 Disable:** 1 = Disable. Clock output is disabled and the output tri-stated. 0 = Enable.
10	**PCLKOUT0 Disable:** 1 = Disable. Clock output is disabled and the output tri-stated. 0 = Enable.
9:5	Reserved
4:3	**Prefetch Policy:** 00 = Allows prefetching on memory read multiple, memory read and memory read line. 01 = Allow prefetching on memory read multiple, memory read line and not memory read. 10 = Disables all prefetching. 11 = Disables all prefetching.
2	**Delay Transaction (DT) Depth:** If the system is designed to support only 2 PCI master only, this bit should be set to 1. This will increase the DT buffer depth to 512 bytes for each of these PCI masters. If the system is designed to support more than 2 PCI master, the DTD bit should be set to 0. 0 = 4 DT, 256 bytes each 1 = 2 DT, 512 bytes each
1:0	**Maximum Delay Transactions:** These bits set the maximum number of delay transaction allowed during test mode. 00 = 4 active, 4 pending 01 = 1 active, 1 pending 10 = 2 active, 2 pending 11 = Reserved

2.2.30. Multi-Transaction Timer (MTT) Register (D31:F0)

Address Offset: 70h
Default Value: 00h
Attribute: R/W
Size: 8 bits

MTT is an 8-bit register that controls the amount of time that the P64H's arbiter allows a PCI initiator to perform multiple back-to-back transactions on the PCI bus. The P64H's MTT mechanism is used to guarantee a fair share of the Primary PCI bandwidth to an initiator that performs multiple back-to-back transactions to fragmented memory ranges (and as a consequence it can not use long burst transfers).

The number of clocks programmed in the MTT represents the guaranteed time slice (measured in PCI clocks) allotted to the current agent, after which the arbiter will grant another agent that is requesting the bus. The default value of MTT is 00h and disables this function. The MTT value can be programmed with 8-clock granularity in the same manner as MLT. For example, if the MTT is programmed to 18h, then the selected value corresponds to the time period of 24 PCI clocks.

Bit	Description
7:3	**Multi-Transaction Timer Count Value:** This field specifies the amount of time that GNT# will remain asserted to a master requesting multiple transfers. This field specifies the count in an 8-clock (PCI clock) granularity.
2:0	Reserved

2.2.31. Soft_DT_Timer—Soft Delayed Transaction Timer Register (D31:F0)

Address Offset: 80h
Default Value: 00h
Attribute: R/W
Size: 8 bits

Bit	Description
7:2	Reserved
1:0	**2 Bit Soft Delayed Transaction Timer:** These two bits will select the granularity of a counter. For example, value '00' represents 64x33 MHz PCI clocks: 00 = 64 01 = 128 10 = 192 11 = 256

2.2.32. ERR_CMD—Error Command Register (D31:F0)

Address Offset: 90h
Default Value: 00h
Attribute: R/W
Size: 8 bit

This register configures the P64H's Device N responses to various system errors. The actual assertion of the internal SERR# is enabled via the PCI Command register.

Bit	Description
7:3	Reserved
2	**Enable for SERR# on Receiving Target Abort:** 1 = Enable. When this bit is 1 and SERR_EN is 1, the P64H will report SERR# when SERR_RTA is set. 0 = Disable.
1:0	Reserved

2.2.33. ERR_STS—Error Status Register (D31:F0)

Address Offset: 92h
Default Value: 00h
Attribute: R/WC
Size: 8 bit

This register records the cause of system errors in Device 30. The actual assertion of SERR# is enabled via the PCI Command register.

Bit	Description
7:3	Reserved
2	**SERR# Due to Received Target Abort:** 1 = P64H received a target abort. If SERR_EN, the P64H generates an SERR# when SERR_RTA is set. 0 = This bit is cleared by writing a 1 to it.
1:0	Reserved

2.3. Advanced Interrupt Controller (APIC) Registers (D0:F0)

Table 8. APIC Configuration Registers (D0:F0)

Address Offset	Symbol	Register Name/Function	Default	Type
00–01	VID	Vendor ID	8086h	RO
02–03	DID	Device ID	1161	RO
04–05	CMD	Device Command Register	0000h	R/W
06–07	STS	Device Status Register	0000h	RO
08	REVID	Revision ID	See latest Specification Update	RO
09	PIF	Programming Interface Register	20h	RO
0A	SCC	Sub Class Code	00h	RO
0B	BCC	Base Class Code	08h	RO
0C–0D	—	Reserved	00h	RO
0E	HEADTYP	Header Type	80h	RO
10–13	BAR	I/OAPIC Memory Base Register	0_XXXX_X000h	R/W
2C	SUB_vid	Subsystem Vendor ID	8086h	RO
2D	SUB_ID	Subsystem ID	1161h	RO
3F–14	—	Reserved	00h	RO
40–41	APIC_BASE	I/OAPIC Base Address Register	0000h	R/W

NOTES:
1. Reserved registers are read only.

2.3.1. VID—Vendor ID Register (D0:F0)

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

Bit	Description
15:0	**Vendor ID Number**. This is a 16-bit value assigned to Intel. Intel VID=8086.

2.3.2. DID—Device ID Register (D0:F0)

Address Offset: 02–03h
Default Value: 1361h
Attribute: RO
Size: 16 bits

Bit	Description
15:0	**Device ID Number**. This is a 16-bit value assigned to the P64H I/O APIC. DID= 1161h.

2.3.3. CMD—Command Register (D0:F0)

Address Offset: 04–05h
Default Value: 0000h
Attribute: R/W
Size: 16 bits

Bit	Description
15:10	Reserved
9	**Fast Back to Back Enable (FBE)—RO:** Reserved as 0.
8	**SERR# Enable (SERR_EN)—RO:** Reserved as 0.
7	**Wait Cycle Control (WCC)—RO:** Reserved as 0
6	**Parity Error Response (PER)—R/W:** 1 = The P64H reports parity errors and sets the data parity detected bit in the status register when a data parity error is detected on the hub interface. 0 = No action is taken when parity is detected.
5	**VGA Palette Snoop Enable (VPS)—RO:** Reserved as 0
4	**Memory Write and Invalidate Enable (MWIE)—RO:** Reserved as 0.
3	**Special Cycle Enable (SCE)—RO:** Reserved as 0.
2	**Bus Master Enable (BME)—R/W:** Controls the P64H's ability to initiate memory transactions on the hub interface on behalf of the I/O APIC. 1 = The P64H is enabled to operate as an initiator on the hub interface and allows P64H to accept cycles from I/O APICI to run on the hub interface. 0 = The P64H does not respond to Memory or I/O transactions from I/O APIC and does not initiate I/O or memory transactions on the hub interface.
1	**Memory Space Enable (MSE)—R/W:** Controls the I/O APIC's response to memory transactions on the hub interface. 1 = The I/O APIC response to memory transactions initiated on the hub interface is enabled. 0 = The I/O APIC does not respond to memory transactions initiated on the hub interface.
0	**I/O Space Enable (IOE)—RO:** Reserved as 0.

2.3.4. STS—Device Status Register (D0:F0)

Address Offset: 06–07h
Default Value: 0000h
Attribute: RO
Size: 16 bits

Bit	Description
15:0	Reserved. Hardwired to 0.

2.3.5. REVID—Revision ID (D0:F0)

Address Offset: 08h
Default Value: See latest Specification Update
Attribute: RO
Size: 8 bits

Bit	Description
7:0	**Revision Identification Number:** 8-bit value that indicates the revision number for the P64H I/O APIC.

2.3.6. PIF—Programming Interface Register (D0:F0)

Address Offset: 09h
Default Value: 20h
Attribute: RO
Size: 8 bits

Bit	Description
7:0	**Programming Interface:** This 8-bit value identifies a specific register-level programming interface so that device independent software can interact with the device. (This is related to the Version ID in the APIC version register).

2.3.7. SCC—Sub-Class Code Register (D0:F0)

Address Offset: 0Ah
Default Value: 00h
Attribute: RO
Size: 8 bits

Bit	Description
7:0	**Sub-Class Code:** This 8-bit value indicates a generic 8259 PIC. 00h = Generic 8259 PIC

2.3.8. BCC—Base-Class Code Register (D0:F0)

Address Offset: 0Bh
Default Value: 08h
Attribute: RO
Size: 8 bits

Bit	Description
7:0	**Base Class Code:** This 8-bit value indicates the device type. 08h = Base system peripheral.

2.3.9. HEADTYP—Header Type Register (D0:F0)

Address Offset: 0Eh
Default Value: 80h
Attribute: RO
Size: 8 bits

Bit	Description
7	**Multi-function Device:** This bit is 1 to indicate a multi-function device.
6:0	**Header Type:** This indicates that it is a normal PCI device 00 = Register layout conforms to typical PCI device

2.3.10. BAR—I/O APIC Memory Base Address Register (D0:F0)

Address Offset: 10h
Default Value: 00h
Attribute: R/W
Size: 32 bits

Bit	Description
31:12	**I/O APIC Memory Base Address (IOAPIC_MBA)—R/W:** These bits enable the I/O APIC to be relocated anywhere within the lower 4-GB address space.
11:4	Reserved.
3	**Prefetchable—RO**. Indicates that the BAR is not prefetchable.
2:1	**Location—RO**. Indicates that the address can be located anywhere in the 32-bit address space.
0	**Space Indicator—RO**. Indicates that the BAR is in memory space.

2.3.11. SUB_VID—Subsystem Vendor ID Register (D0:F0)

Address Offset: 2Ch
Default Value: 8086h
Attribute: RO
Size: 16 bits

Bit	Description
15:0	**Subsystem Vendor ID**. The subsystem vendor ID is 8086h.

2.3.12. SUB_ID—Subsystem ID Register (D0:F0)

Address Offset: 2Eh
Default Value: 1161h
Attribute: RO
Size: 16 bits

Bit	Description
15:0	**Subsystem ID**. The subsystem ID is 1161h.

2.3.13. APIC_BASE—I/O APIC Base Register (D0:F0)

Address Offset: 40–41h
Default Value: 00h
Attribute: R/W
Size: 16 bit

Bit	Description
15	**APIC Base Enable:** 1 = Enable. Range FECX_YZ00 to FECX_YZFF is enabled as an alternative access to the I/O APIC registers. Bits XYZ are defined by bits 11:8, 7:4, and 3:0 of this register. 0 = Disable.
14:12	Reserved.
11:8	**X Base Address (XBAD)—Bits [19:16]:** These bits determine the high order bits of the I/O APIC address map. When a memory address is recognized by the P64H, which matches either FECX_YZ00 or FECX_YZ10, the P64H will access the internal I/O APIC.
7:4	**Y Base Address (YBAD)—Bits[15:12]:** These bits determine the low order bits of the I/O APIC address map. When a memory address is recognized by the P64H, which matches either FECX_YZ00 or FECX_YZ10, the P64H will access the internal I/O APIC.
3:0	**Z Base Address (ZBAD)—Bits[11:8]:** These bits determine the low order bits of the I/O APIC address map. When the memory address is recognized by the P64H, which matches either FECX_YZ00 or FECX_YZ10, the P64H will access the internal I/O APIC.

2.4. I/O APIC Registers

The APIC is accessed via an indirect addressing scheme. Two registers are visible by software for manipulation of most of the APIC registers. Two additional registers are included for functions beyond the original APIC. These registers are mapped into memory space. The address locations of these registers can be relocated via the APIC Base Address Relocation register in the PCI configuration space for Device 0:Function 0 at Offset 40h. The registers are shown in Table 9.

Table 9: APIC Direct Registers

Address	Register	Size	Type
FECX_YZ00h	Index Register	8 bits	R/W
FECX_YZ10h	Window Register	32 bits	R/W
FECX_YZ20h	IRQ Pin Assertion Register	8 bits	WO
FECX_YZ40h	EOI Register	8 bits	WO

Table 10 lists the registers, which can be accessed within the APIC via the Index Register. When accessing these registers, accesses must be a DWord at a time. For example, software should never access byte 2 from the window register before accessing bytes 0 and 1. The hardware will not attempt to recover from an incorrect programming model in this case.

Table 10: APIC Direct Access Memory Registers

Index	Register	Size	Type
00h	Index Register	8 bits	R/W
10–13h	Window Register	8 bits	R/W
20h	IRQ Pin Assertion Register	8 bits	R/W
40h	EOI Register	8 bits	R/W

2.4.1. Index Register (D0:F0)

Memory Address: 00h
Default Value: 00h
Attribute: R/W
Size: 8 bits

The Index Register will select which indirect register appears in the window register to be manipulated by software. Software will program this register to select the desired APIC internal register.

Bit	Description
7:0	**Index:** This is an 8-bit pointer into the I/O APIC register table.

2.4.2. Window Register (D0:F0)

Memory Address: 10h
Default Value: 00000000h
Attribute: R/W
Size: 32 bits

This is a 32-bit register specifying the data to be read from or written to by the Register Select register. This register can be accessed in byte quantities.

Bit	Description
31:0	**Window:** Data to be written to the indirect register on writes, and location of register data from the indirect register on reads.

2.4.3. IRQ Pin Assertion Register (D0:F0)

Memory Address: 20h
Default Value: N/A
Attribute: WO
Size: 8 bits

The IRQ Pin Assertion Register is present to provide a mechanism to scale the number of interrupt inputs into the I/O (x) APIC without increasing the number of dedicated input pins. When a device that supports this interrupt assertion protocol requires interrupt service, that device issues a write to this register. Bits 4:0 written to this register contain the IRQ number for this interrupt. The only valid values are 0-23. Bits 31:5 are ignored.

Bit	Description
7:0	**Assertion:** Virtual pin to be asserted.

2.4.4. EOI Register (D0:F0)

Memory Address: 40h
Default Value: N/A
Attribute: WO
Size: 8 bits

The EOI register is present to provide a mechanism to maintain the level triggered semantics for level-triggered interrupts issued on the parallel bus.

When a write is issued to this register, the I/O (x) APIC will check the lower 8 bits written to this register, and compare it with the vector field for each entry in the I/O Redirection Table. When a match is found, the Remote_IRR bit for that I/O Redirection Entry would be cleared.

Note that if multiple I/O Redirection entries, for any reason, assign the same vector for more than one interrupt input, each of those entries will have the Remote_IRR bit reset to 0. If the interrupt was prematurely reset, it will not be lost because if its input remained active when the Remote_IRR bit is cleared, the interrupt will be reissued and serviced at a later time.

Bit	Description
7:0	**Redirection Entry Clear—WO:** When a write is issued to this register, the I/O APIC will check this field and compare it with the vector field for each entry in the I/O Redirection Table. When a match is found, the Remote_IRR bit for that I/O Redirection Entry would be cleared.

Table 11. Indirect Access Memory Register

Index	Register	Size	Type
00	APIC ID	32 bits	R/W
01	Version	32 bits	RO
02	Arbitration ID	32 bits	RO
10	Redirection Table Low DWord	32 bits	R/W
11	Redirection Table High DWord	32 bits	R/W

2.4.5. ID Register (D0:F0)

Index Offset: 00h
Default Value: 00000000h
Attribute: R/W
Size: 32 bits

The APIC ID serves as a physical name of the APIC. The APIC bus arbitration ID for the APIC is derived from its I/O APIC ID. This register is reset to zero on power up reset.

Bit	Description
31:28	Reserved
27:24	**APIC ID:** Software must program this value before using the APIC.
23:16	Reserved.
15	**DT:** This bit defines the Delivery Type. 0 = Intel 840 chipset. This bit must be programmed to 0 for processors used with the Intel 840 chipset. A 0 indicates that the interrupt delivery mechanism is APIC serial bus. 1 = Mode for APIC
14	**LTS:** 1 = Enable. Software sets this bit to 1 to enable “Level Deassert” messages. These occur only if all of the following conditions occur: • the DT bit is set 1, and • the Remote IRR bit is active for a particular interrupt, and • the particular interrupt is set for level-triggered mode, and • the interrupt input associated with that particular interrupt goes away.
13:0	Reserved

2.4.6. Version Register (D0:F0)

Index Offset: 01h
Default Value: 00170020h
Attribute: RO
Size: 32 bits

Each I/O APIC contains a hardwired Version Register that identifies different implementation of APIC and their versions. The maximum redirection entry information also is in this register, to let software know how many interrupts are supported by this APIC.

Bit	Description
31:24	Reserved
23:16	**Maximum Redirection Entries:** This is the entry number (0 being the lowest entry) of the highest entry in the redirection table. It is equal to the number of interrupt input pins minus one and is in the range 0 through 239. This field is hardwired and is read-only. In the P64H this field is hardwired to 17h to indicate 24 interrupts.
15	**PRQ:** This bit is set to 1 to indicate that this version of the I/O APIC implements the IRQ Assertion register and allows PCI devices to write to it to cause interrupts.
14 :8	Reserved
7:0	**Version:** This is a version number that identifies the implementation version. This field is hardwired and is read-only. Version = 20h.

2.4.7. ARBID—Arbitration ID Register (D0:F0)

Index Offset: 02h
Default Value: 00000000h
Attribute: RO
Size: 32 bits

This register contains the bus arbitration priority for the APIC. This register is loaded when the APIC ID register is loaded. A rotating priority scheme is used for APIC bus arbitration. The winner of the arbitration becomes the lowest priority agent and assumes 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 will take the winner's arbitration ID and will increment it by one. Arbitration IDs are changed only for messages that are transmitted successfully, except in the case of Low Priority messages, where the arbitration ID is changed even if message was not successfully transmitted. A message is transmitted successfully if no checksum error or acceptance error was reported for that message. The APIC Arbitration ID register is always loaded with APIC ID during a "level triggered INIT with deassert" message.

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

2.4.8. Boot Configuration Register

Index Offset: 03h
Default Value: 00000000h
Attribute: RW
Size: 32 bits

In systems using the Intel® Pentium® 4 processor and using Microsoft* Windows NT* or Windows* 2000 operating systems, Windows NT and Windows 2000 will overwrite a register bit inside of the P64H APIC previously set by BIOS. This means that Windows NT and Windows 2000 operating systems essentially disable the APIC during OS boot.

Bit	Description
31:1	Reserved
0	**DT:** This bit defines the delivery type. 0 = This bit must be programmed to 0 for systems using the Intel® Pentium® III processor-based systems and chipsets that use external APIC signaling. A 0 indicates that the interrupt delivery mechanism is the APIC serial bus. 1 = MSI Capable (Front Side Bus); This is used for Pentium 4 processor-based systems and chipsets that do not use external APIC signaling.

2.4.9. Redirection Table Low DWord (D0:F0)

Index Offset: 10h
Default Value: Bit 16 = 1. Bits [15:12] = 0. All other bits undefined
Attribute: R/W
Size: 32 bits

The Redirection Table has a dedicated entry for each interrupt pin. The information in the Redirection Table is used to translate the interrupt manifestation on the corresponding interrupt pin into an APIC message.

The APIC will respond to an edge triggered interrupt as long as the interrupt is held until after the acknowledge cycle has begun. Once the interrupt is detected, a delivery status bit internally to the I/O APIC is set. The state machine will step ahead and wait for an acknowledgment from the APIC bus unit that the interrupt message was sent over the APIC bus. Only then will the I/O APIC be able to recognize a new edge on that interrupt pin. That new edge will only result 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.

Bit	Description
31:17	Reserved
16	**Mask— R/W:** 0 = Not masked: An edge or level on this interrupt pin results in the delivery of the interrupt to the destination. 1 = Masked: Interrupts are not delivered nor held pending. Setting this bit after the interrupt is accepted by a local APIC has no effect on that interrupt. This behavior is identical to the device withdrawing the interrupt before it is posted to the processor. It is software's responsibility to deal with 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.
15	**Trigger Mode— R/W:** This field indicates the type of signal on the interrupt pin that triggers an interrupt. 0 indicates edge sensitive, 1 indicates level sensitive.

Bit	Description
14	**Remote IRR— R/W:** This bit is used for level triggered interrupts; its meaning is undefined for edge triggered interrupts. For level triggered interrupts, this bit is set when Local APIC/s accept the level interrupt sent by the I/O APIC. Remote IRR bit is reset when an EOI message is received from a local APIC.
13	**Interrupt Input Pin Polarity— R/W:** This bit specifies the polarity of each interrupt signal connected to the interrupt pins. A value of 0 means the signal is active high. A value of 1 means the signal is active low.
12	**Delivery Status— RO:** This field contains the current status of the delivery of this interrupt. It is read only. Writes to this bit have no effect. 0 = Idle. No activity for this interrupt 1 = Pending. Interrupt has been injected, but delivery is held up due to the APIC bus being busy or the inability of the receiving APIC unit to accept the interrupt at this time.
11	**Destination Mode— R/W:** This field determines the interpretation of the Destination field. 0 = Physical. Destination APIC ID is identified by bits [59:56]. 1 = Logical. Destinations are identified by matching bit [63:56] with the Logical Destination in the Destination Format Register and Logical Destination Register in each Local APIC.
10:8	**Delivery Mode—R/W:** This field specifies how the APICs listed in the destination field should act upon reception of this signal. Certain delivery modes will only operate as intended when used in conjunction with a specific trigger mode. These encoding are: 000 **Fixed.** Deliver the signal on the INTR signal of all processor cores listed in the destination. Trigger mode can be edge or level. 001 **LowestPriority.** 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 can be edge or level. 010 **SMI.** System Management Interrupt. Requires the interrupt to be programmed as edge triggered. 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 is treated as an edge triggered interrupt even if it is programmed as level triggered. For proper operation this redirection table entry must be programmed to edge triggered. The NMI delivery mode does not set the RIRR bit. Once the interrupt is detected, it will be sent over the APIC bus. If the redirection table is incorrectly set to level, the loop count will continue counting through the redirection table addresses. Once the count for the NMI pin is reached again, the interrupt will be sent over the APIC bus again. 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 is always treated as an edge triggered interrupt even if programmed as level triggered. For proper operation this redirection table entry must be programmed to edge triggered. The INIT delivery mode does not set the RIRR bit. Once the interrupt is detected, it will be sent over the APIC bus. If the redirection table is incorrectly set to level, the loop count will continue counting through the redirection table addresses. Once the count for the INIT pin is reached again, the interrupt will be sent over the APIC bus again 110 Reserved 111 **ExtINT.** Deliver the signal to the INTR signal of all processor cores listed in the destination as an interrupt that originated in an externally connected 8259A compatible interrupt controller. The INTA cycle that corresponds to this ExtINT delivery will be routed to the external controller that is expected to supply the vector. Requires the interrupt to be programmed as edge triggered.
7:0	**Vector—RO:** This field contains the interrupt vector for this interrupt. Values range between 10h and FEh.

2.4.10. Redirection Table High DWord

Index Offset: 11h
Default Value: 0000h
Attribute: R/W
Size: 32 bits

Bit	Description
31:24	**Destination ID**. This information is transferred depending on the type of message.
23:00	**Extended Destination ID**.
15:0	Reserved

This page is intentionally left blank.

3. Functional Description

3.1. Address Decoding

The P64H uses three address ranges to control I/O and memory transaction forwarding from the hub interface to PCI bus, and from PCI bus to hub interface.

- One 16-bit I/O address range (upstream/downstream)
- One 32-bit non-prefetchable memory address range (non-prefetchable memory-downstream)
- One 44-bit prefetchable memory address range (upstream)

These address ranges are defined by the base and limit address registers in the P64H P-2-P configuration space. Transactions, within the 16-bit I/O address range are forwarded downstream from the hub interface to the PCI bus. Otherwise, the transactions are forwarded upstream, from the PCI bus to hub interface. Transactions, within the prefetchable memory address, are forwarded downstream. Otherwise, the transactions are forwarded upstream.

3.1.1. I/O Address Decoding

The P64H uses the following mechanisms (defined in the Register Description Chapter) to specify the I/O address space for downstream and upstream forwarding:

- I/O Base Address (IOBASE) register and I/O Limit Address (IOLIM) register (D31:F0)
- VGA mode bit (Bridge_CNT register, bit 3) (D31:F0)

The P64H response to I/O transaction may be modified via:

- I/O Enable bit (CMD register) (D31:F0)
- Master Enable bit (CMD register) (D31:F0)
- VGA Enable bit (BRIDGE_CNT register) (D31:F0)

All P64H I/O transactions are forwarded to the PCI bus according to the address range specified by the IOBASE register and IOLIM register. To enable downstream forwarding of I/O transactions, the I/O Enable (IOE) bit must be set in the Command register (CMD), in P64H configuration space. If the I/O enable bit is not set, all I/O transactions initiated on the hub interface are master aborted and PCI bus I/O transaction within 64 KB will be forwarded upstream irrespective of the value in the IOBASE register and IOLIM register.

To enable upstream forwarding of I/ O transactions, the master enable bit must be set in the Command register. Otherwise, the P64H ignores all I/O memory transaction initiated on the PCI bus.

3.1.1.1. I/O Base/Limit Address Registers

The P64H implements one set of I/O Base Address and I/O Limit Address registers (IOBASE and IOLIM) that define the I/O address range for downstream and upstream forwarding. The P64H supports 16-bit I/O addressing that allows I/O addresses downstream of the P64H to be mapped anywhere in a 64-KB I/O address space. I/O transactions with addresses, within the range specified by the IOBASE and IOLIM registers, are forwarded downstream from the hub interface to PCI bus. I/O transaction addresses outside of this range are forwarded upstream. The I/O range can be turned off by setting the I/O base address to a value greater than that of the I/O limit address. I/O transactions will not be forwarded upstream from the PCI bus to the hub interface.

The P64H I/O range has a minimum granularity of 4 KB and is aligned on a 4-KB boundary. The maximum I/O range is 64 KB in size. The IOBASE register consists of an 8-bit field at configuration address 1Ch, and a 16-bit field at address 30h. The top four bits of the 8-bit field define bits [15:12] of the I/O base address. The bottom four bits are 0s to indicate that the P64H supports 16-bit I/O addressing. Bits [11:0] of the base address are assumed to be 0s, which naturally aligns the base address to a 4-KB boundary. Since the P64H does not support 32-bit I/O addressing, the I/O Base Upper 16-Bits Register (IOBASE_HI) is not used. After reset, the value of the I/O base address is initialized to 0000h.

The IOLIM register consists of an 8-bit field at configuration offset 1Dh and a 16-bit field at offset 32h. The top four bits of the 8-bit field define bits [15:12] of the I/O limit address. The bottom four bits are hardwired to 0s to indicate that only 16-bit I/O addressing is supported by the P64H. Bits [11:0] of the limit address are assumed to be FFFh, which naturally aligns the limit address to the top of a 4-KB I/O address block. Since the P64H does not support 32-bit I/O addressing, the I/O Limit Upper 16-bits Register (IOBASE_LO) is not used. After reset, the value of the I/O limit address is reset to 0FFFh.

The initial states of the IOBASE and IOLIM registers define an I/O range of 0000 0000h to 0000 0FFFh, which is the bottom 4 KB of the I/O space. These registers must be written with their appropriate values before setting either the I/O Enable bit (IOE) or Bus Master Enable bit (BME) in the CMD register.

3.1.2. Memory Address Decoding

The P64H has mechanisms for defining memory address ranges for forwarding of memory transactions:

- Memory Base Address and Memory Limit Address registers (MEMBASE and MEMLIM)
- Prefetchable Memory Base Address and Prefetchable Memory Limit Address registers (PREF_MEM_BASE and PREF_MEM_LIM)
- Prefetch Memory Base Address Upper 32 bits (Read Only) (PREF_MEM_BASE_UPPER)
- Prefetch Memory Limit Address Upper 32 bits (Read Only) (PREF_MEM_LIM_UPPER)
- VGA mode (Bridge_CNT, bit 3)

The P64H will not prefetch data from PCI devices and reads only the exact number of bytes requested by the host. The P64H decodes either the MEMBASE and MEMLIM registers or the PREF_MEM_BASE and PREF_MEM_LIM registers to determine if the access is to the P64H. If the memory access was not designated for the P64H, the transaction would then be forwarded to the PCI Bus. PREF_MEM_BASE_UPPER and Prefetchable PREF_MEM_LIM_UPPER registers are set to 0s for PCI memory access below 4 GB. The P64H does not implement 64-bit addressing (DAC Cycles) downstream.

Note: The P64H supports 44 bits addressing for the upstream DAC cycles only. The PREF_MEM_BASE_UPPER and PREF_MEM_LIM_UPPER registers will only contain 12 bits, address bits 32 to 43. If the master abort bit is set (Bridge_CNT register), the P64H will generate a target abort if it detects non-zero in any address bits 44 through 63. The PREF_MEM_BASE_UPPER and PREF_MEM_LIM_UPPER Upper registers are not used to decode upstream DAC cycles.

The P64H response to memory transaction may be modified via:

- Memory enable bit (CMD register) (D31:F0)
- Master enable bit (CMD register) (D31:F0)
- VGA enable bit (BRIDGE_CNT register) (D31:F0)

To enable downstream forwarding of memory transactions, the Memory Space Enable bit (MSE) must be set in the P64H's Command register (CMD). To enable upstream forwarding of memory transactions, the Bus Master Enable (BME) bit must be set in the Command register. If the BME bit is not set, the P64H will ignore all memory transactions initiated on the PCI bus.

3.1.2.1. Non-Prefetchable Memory Base/Limit Address Registers

The Memory Base Address register (MEMBASE) and Memory Limit Address register (MEMLIM) define the address range that the P64H uses to determine when to forward memory commands. The P64H forwards a memory transaction from the hub interface to the PCI interface if the transaction address falls within the memory address range. The P64H ignores memory transactions initiated on the PCI interface that fall into this address range. Any transactions that fall outside of this address range are also ignored on the hub interface and are forwarded upstream from the PCI bus (provided that they do not fall into the prefetchable memory range) to the hub interface.

The non-prefetchable memory range supports 32-bit addressing only. The *PCI-to-PCI Bridge Architecture Specification* does not provide for 64-bit addressing in the non-prefetchable memory space. The non-prefetchable memory address range has a granularity and alignment of 1 MB. The maximum non-prefetchable memory address range is 4 GB.

The non-prefetchable memory address range is defined by a 16-bit memory base register at configuration offset 20h and by a 16-bit memory limit register (MEMLIM) at offset 22h. The upper 12 bits of each of these registers correspond to bits [31:20] of the memory address. The lower four bits are hardwired to 0s. The lower 20 bits of the non-prefetchable memory base address are assumed to be 0s, which results in a natural alignment to a 1-MB boundary. The lower 20 bits of the non-prefetchable memory limit address are assumed to be FFFFFh, which results in an alignment to the top of a 1-MB block.

The initial state of the MEMBASE register is 00000000h. The initial state of the MEMLIM register is 000FFFFFh. The initial states of these registers define a non-prefetchable memory range at the bottom of 1-MB block of memory. These registers must be written with the appropriate values before setting either the IOE bit or the BME bit in the Command register in configuration space. Setting the base register to a value greater than that of the limit turns off the memory range.

3.1.2.2. Prefetchable Memory Base/Limit Address Registers

The P64H uses this register to determine if the access is intended for itself or for the PCI bus. The P64H does not prefetch data from the PCI devices. The prefetchable memory base address and prefetchable memory limit address registers (PREF_MEM_BASE and PREF_MEM_LIM) define an address range that the P64H uses to determine when to forward memory commands. The P64H forwards a memory transaction from the hub interface to the PCI interface if the transaction address falls within the prefetchable memory address range. The P64H mater aborts memory transactions initiated on the PCI interface that fall into this address range. The P64H does not respond to any transactions that fall outside this address range on the hub interface and forwards those transactions upstream from the secondary interface (provided that they do not fall into the non-prefetchable memory range.).

The P64H does not support 64-bit addressing. The Prefetchable Memory Base Upper 32 bits (PREF_MEM_BASE UPPER, offset 28h–2Bh) and Prefetchable Memory Limit Upper 32 bits (PREF_MEM_LIM UPPER, offset 2Ch–2Fh) registers are not used and are read only. The prefetchable memory address range has a granularity and alignment of 1 MB. The maximum memory address range is 4 GB when 32-bit addressing is used.

The prefetchable memory address range is defined by the 16-bit PREF_MEM_BASE register (offset 24h) and 16-bit PREF_MEM_LIM register (offset 28h). The upper 12 bits of each of these registers correspond to bits [31:20] of the memory address. The lower four bits are hardwired to 1h, indicating 64-bit address support. The lower 20 bits of the prefetchable memory base address are assumed to be 00000h, which results in a natural alignment to a 1-MB boundary. The lower 20 bits of the prefetchable memory limit address are assumed to be F FFFFh, which results in an alignment to the top of a 1-MB block.

The initial state of the PREF_MEM_BASE register is 00000000h. The initial state of PREF_MEM_LIM register is 000F FFFFh. Note that the initial states of these registers define a prefetchable memory range at the bottom 1-MB block of memory. Write these registers with their appropriate values before setting either the memory enable bit or the master enable bit in the Command register in configuration space.

To turn off the prefetchable memory address range, write the PREF_MEM_BASE register with a value greater than that of the PREF_MEM_LIM register. The entire base value must be greater than the entire limit value.

3.1.2.3. P64H 44-Bit Addressing

The P64H supports 44-bit memory address decoding for forwarding of dual address memory transactions. The dual address cycle is used to support 44-bit addressing. The first address phase of a dual address transaction contains the low 32 address bits, and the second address phase contains the high 12 address bits. The prefetchable address space on the PCI bus will reside in the first 4 GB of memory. The P64H ignores all dual address cycle transactions initiated on the hub interface and forwards all dual address transaction initiated on the PCI interface upstream.

3.1.3. VGA Mode

The P64H supports only the VGA mode, allowing downstream and upstream VGA-compatible addressing. When a VGA-compatible device exists downstream from the P64H, the VGA enable bit must be set (BRIDGE CNT register) in configuration space to enable VGA mode. The I/O enable and Memory Enable bits, in the CMD register, must be set in order for VGA transactions to be forwarded downstream. When the P64H is operating in VGA mode, it forwards downstream those transactions addressing the VGA frame buffer memory and VGA I/O registers, regardless of the values of the P64H base and limit address registers. The P64H ignores transactions initiated on the PCI interface addressing these locations.

The VGA frame buffer consists of the following memory address range: 000A 0000h–00B FFFFh. Read transactions to frame buffer memory are treated as non-prefetchable read transactions. The P64H requests only a single data transfer from the target, and read byte enable bits are forwarded to the target bus. The VGA I/O addresses consist of the following I/O addresses:

- 3B0h–3BBh
- 3C0h–3DFh

These I/O addresses are aliased every 1 KB throughout the first 64 KB of I/O space. This means that:

- Address bits [9:0] (3B0h–3BBh and 3C0h–3DFh) are decoded,
- Address bits [15:10] are not decoded and can be any value
- Address bits [31:16] must be all 0’s.

VGA BIOS addresses starting at C0000h are not decoded in VGA mode. When VGA enable bit is cleared, the same memory and I/O cycles mentioned are forwarded upstream.

3.1.3.1. Memory Map

The I/O APIC is the only function that resides in the P64H memory map. Cycles that arrive from MCH (via the hub interface) will either be claimed by the I/O APIC or be forwarded to the PCI Bus.

3.1.4. PCI Devices and Functions

The P64H integrates three PCI functions (Table 12). The P64H PCI-to-PCI bridge configuration registers (D31:F0) must be accessed by the MCH using Type 0 mechanism. The I/O APIC function Configuration registers must be accessed by MCH using Type 1 mechanism.

Table 12. PCI Devices and Functions

Device #:Function #	Function Description	P64H Bus
Device 31: Function 0	Hub interface to PCI Bridge	Primary bus
Device 0: Function 0	I/O APIC	Secondary bus (PCI)

3.2. PCI Bus Operation

3.2.1. Types of Transactions

Table 13 lists the command code and name of each PCI transaction. The Master and Target columns indicate P64H support for each transaction on the hub interface and on the PCI bus.

Table 13. Intel® P64H PCI Transaction

Transaction Types		P64H as Master	P64H as Target
0000	Interrupt acknowledge	No	No
0001	Special cycle	Yes	No
0010	I/O read	Yes	Yes
0011	I/O write	Yes	Yes
0100	Reserved	No	No
0101	Reserved	No	No
0110	Memory read	Yes	Yes
0111	Memory write	Yes	Yes
1000	Reserved	No	No
1001	Reserved	No	No
1010	Configuration Read	Yes	Yes
1011	Configuration Write	Yes	Yes
1100	Memory Read Multiple	No	Yes
1101	Dual Address Cycle	No	Yes
1110	Memory Read Line	No	Yes
1111	Memory Write and Invalidate	No	Yes

The following PCI commands are not supported by the P64H:

- The P64H never initiates a PCI transaction with a reserved command code and as a target, the P64H ignores reserved command codes.
- The P64H never initiates an interrupt acknowledge transaction and, as a target, the P64H ignores interrupt acknowledge transactions. Interrupt acknowledge transactions are expected to reside entirely on the primary PCI bus closest to the host bridge.
- The P64H does not respond to special cycle transactions. The P64H cannot guarantee delivery of a special cycle transaction to downstream buses because of the broadcast nature of the special cycle command and the inability to control the transaction as a target. To generate special cycle transactions on other PCI buses, either upstream or downstream, a Type 1 configuration command must be used.
- The P64H does not generate Type 0 configuration transactions on the primary interface, nor does it respond to Type 0 configuration transactions on the secondary PCI interface.
- Memory Write and Invalidate command is treated as a Memory Write command.

3.2.2. Address Phase

The typical PCI transaction generally consists of one or two address phases, followed by one or more data phases. An address phase lasts one PCI clock cycle. The first address phase is designated by the assertion of the FRAME# signal. The number of address phases depends on whether the address is 32 bits or 44 bits.

Single Address Phase

A 32-bit address uses a single address phase. This address is driven on AD[31:0], and the Bus command is driven on C/BE[3:0]#. The P64H supports the linear increment address mode only, which is indicated when the lower 2 address bits are equal to 0. If either of the lower 2 address bits is nonzero, the P64H automatically disconnects the transaction after the first data transfer.

Dual Address Phase

Dual address transactions are PCI transactions that contain two address phases specifying a 64-bit address. The first address phase is denoted by the assertion of the FRAME# signal. The second address phase always follows on the next clock cycle.

For a 32-bit interface, the first address phase contains the Dual Address command code on the C/BE[3:0]# lines, and the low 32 address bits on the AD[31:0] lines. The second address phase consists of the specific Memory Transaction command code on the C/BE[3:0]# lines, and the high 32 address bits on the AD[31:0] lines. In this way, 64-bit addressing can be supported on 32-bit PCI buses.

The *PCI-to-PCI Bridge Architecture Specification* supports the use of dual address transactions in the prefetchable memory range only. The P64H supports dual address transactions in both the upstream and downstream direction. The P64H supports a programmable 64-bit address range in prefetchable memory for downstream forwarding on dual address transaction. Dual address transaction falling outside of the prefetchable range will be forwarded upstream only. Prefetching and posting are performed in a manner consistent with the guidelines given in this specification for each type of memory transaction in prefetchable memory space.

The P64H responds only to dual address transactions that use the following transaction command codes:

- Memory Write
- Memory Write and Invalidate
- Memory Read
- Memory Read Line
- Memory Read Multiple

Use of other transaction codes may result in a master abort.

Device Select (DEVSEL#) Generation

The P64H always performs positive address decoding when accepting transactions from the PCI bus. The P64H never subtractively decodes and operates with medium DEVSEL# timing on the PCI bus.

Data Phase

The address phase(s) of a PCI transaction are followed by one or more data phases. A data phase is completed when IRDY# and either TRDY# or STOP# are asserted. A transfer of data occurs only when both IRDY# and TRDY# are asserted within the same PCI clock cycle. The last data phase of a transaction is indicated when either FRAME# is deasserted and both TRDY# and IRDY# are asserted, or IRDY# and STOP# are asserted.

3.2.3. Write Transactions

All upstream memory write transactions, from the PCI bus, are treated as posted write transactions. Posted write forwarding is used for memory write, and memory write and invalidate transactions.

Table 14. Posted Write Forwarding

Type of Transaction	Type of Forwarding
Memory Write	Posted
Memory Write and Invalidate	Posted

Posted Write Transactions

When the P64H determines that a memory write transaction is to be forwarded upstream across the bridge, it will check that buffer space is available in the posted data queue. The P64H then asserts DEVSEL# and TRDY# within the same cycle. This enables the P64H to accept write data without obtaining access to the hub interface. The P64H can accept 1 DWord or 1QWord of write data with every PCI clock cycle (i.e., no target wait-states are inserted). This write data is stored in the posted write buffers and is subsequently delivered to the target. The P64H continues to accept write data until one of the following events occurs:

- The initiator terminates the transaction by deasserting FRAME# and IRDY#.
- An internal write address boundary is reached or an aligned 4-KB boundary.
- The posted write data buffer is full.

If either of the last two events (above) occurs, the P64H will return a target disconnect to the initiator to terminate the transaction.

Memory Write and Invalidate Transactions

Posted write forwarding is used for Memory Write and Invalidate transactions. The P64H treats Memory Write and Invalidate transactions as Memory Write commands.

Delayed Write Transaction

Delayed write forwarding is used for I/O write transactions. A delayed write transaction guarantees that the actual target response is returned back to the initiator without holding the initiating bus in the wait-states. A delayed write transaction is limited to a single DWord data transfer.

When a write transaction is first detected on the PCI bus, the P64H will forward it as a delayed transaction, claim the access by asserting DEVSEL# and return a target retry to the initiator. During the address phase, the P64H will sample the command, address and address parity one PCI cycle later. After IRDY# is asserted, the P64H also samples the first DWord, byte enable bits and data parity. This information is placed into the delayed transaction queue. The transaction is queued only if the existing delayed transactions do not have the same address and command, and the delayed transaction queue is not full. When all ordering constraints are satisfied, the P64H will transfer the data, on the hub interface, to the target.

If the initiator repeats the same write transaction (same command, address, byte enable bits) while the delayed transaction is at the head of the queue, the P64H will assert DEVSEL# and return TRDY# to the initiator. This indicates that the write data was transferred.

If the initiator repeats the write transaction while the data is being transferred to the target, the P64H will return a target retry to the initiator. The P64H continues to return a target retry to the initiator until the write data delivery is complete.

Fast Back-to-Back Write Transaction

The P64H supports (recognizes) upstream Fast Back-to-Back write transaction. During Fast Back-to-Back write transaction, the P64H returns a target retry if it cannot accept the second write transaction because of buffer space limitation. The P64H does not perform write combining or merging.

3.2.4. Read Transactions

Delayed read forwarding is used for all memory and I/O read transactions that cross the P64H. Delayed memory read transactions are prefetched on the hub interface. Delayed I/O transaction is non-prefetched.

Prefetchable Read Transactions

Prefetchable read transactions are consisted of multiple data transfers. For prefetchable read transaction, the P64H performs speculative DWord reads and transfers the data from the target before the initiator requests it. The amount of data prefetched is dependent on the PCI clock, request size, and command type. It may also be affected by the amount of available buffer space and read address boundaries.

Delayed Read Requests

The P64H treats all read transactions as delayed read transactions, which means that the read request from the initiator is posted into a delayed transaction queue. Read data from the target is placed in the read data queue directed toward the initiator bus interface and is transferred to the initiator when the initiator repeats the read transaction.

When the P64H accepts a delayed read request, it samples the read address, Read Bus command and address parity. When IRDY# is asserted, the P64H also samples the byte enable bits for the I/O transactions. This information is entered into the delayed transaction queue. The P64H then terminates the transaction by signaling a target retry to the initiator. Upon reception of the target retry, the initiator is required to continue to repeat the same read transaction until at least one data transfer is completed, or until a target response (target abort or master abort) is received.

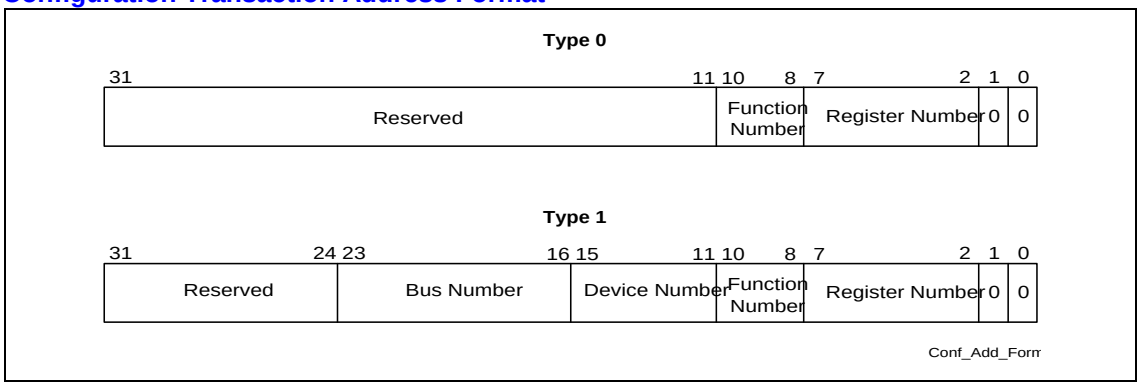
3.2.5. Configuration Transaction

Configuration transactions are used to initialize a PCI system. Every PCI device has a configuration space that is accessed by configuration commands. All P64H registers are accessible in configuration space only. In addition to accepting configuration transactions for initialization of its own configuration space, the P64H also forwards configuration transactions for device initialization in hierarchical PCI systems, as well as for special cycle generation. To support hierarchical PCI bus systems, two types of configuration transactions are specified: Type 0 and Type 1.

Type 0 configuration transactions are issued when the intended target resides on the same PCI bus as the initiator. A Type 0 configuration transaction is identified by the configuration command and the lowest 2 address bits are 00b.

Type 1 configuration transactions are issued when the intended target resides on another PCI bus or when a special cycle is to be generated on another PCI bus. A Type 1 configuration command is identified by the configuration command and the lowest 2 address bits are 01b.

Figure 2. Configuration Transaction Address Format



Type 1 configuration transaction addresses also include a 5-bit field designating the device number that identifies the device on the target PCI bus that is to be accessed. In addition, the bus number in Type 1 transactions specifies the PCI bus to which the transaction is targeted.

The register number is found in both Type 0 and Type 1 formats and it provides the configuration register address to be accessed. The function number is also included in both Type 0 and Type 1 formats and indicates which function of a multifunction device is to be accessed. For single-function devices, this value is not decoded.

The PCI-to-PCI bridge, in the P64H, resides on the hub interface as device #31, function 0. The I/O APIC resides on the secondary bus (PCI bus) of the P64H as device #0, function 0.

Type 0 Access to the P64H

The P64H PCI-PCI Bridge (D: 31, F: 0) configuration space is accessed by a Type 0 configuration transaction on the hub interface. The P64H configuration space cannot be accessed from the PCI bus. The P64H responds to a Type 0 configuration transaction when the following conditions are met during the address phase:

- The Bus command is a configuration read or configuration write transaction.
- Lower 2 address bits must be 00b.

The function code is ignored because device #31 is a single-function device. Read transactions to P64H configuration space do not have side effects, all bytes in the requested DWord are returned regardless of the value of the byte enable bits. The P64H ignores all Type 0 transactions initiated on the secondary PCI interface by other PCI masters.

Type 1 to Type 0 Translation

Type 1 configuration transactions are used specifically for device configuration in a hierarchical PCI bus system. A PCI-to-PCI bridge is the only type of device that should respond to a Type 1 configuration command. Type 1 configuration commands are used when the configuration access is intended for a PCI device that resides on a PCI bus.

The P64H performs a Type 1 to Type 0 translation when the Type 1 transaction is generated on the hub interface and is intended for a device attached directly to the P64H PCI bus. For example, the I/O APIC resides on this PCI bus. The P64H must convert the configuration command to a Type 0 format so that the secondary bus device can respond to it. Type 1 to Type 0 translations are performed only in the downstream direction and never on the hub interface bus.

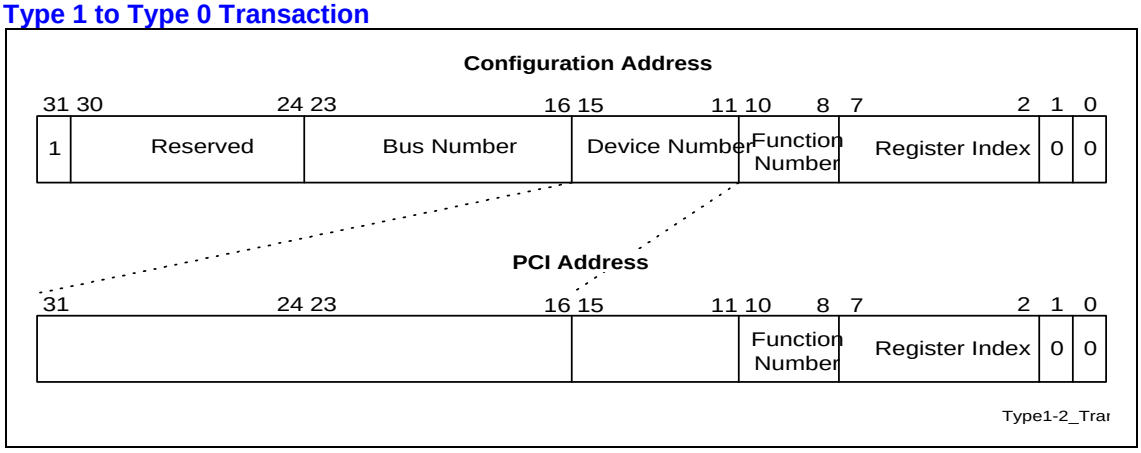
The P64H responds to a Type 1 configuration transaction and translates it into a Type 0 transaction on the PCI bus when the following conditions are met during the address phase:

- The lower 2 address bits on AD[1:0] are 01b.
- The bus number in address field AD[15:11] is equal to the value in the secondary bus number register in P64H configuration space.
- The Bus command is a configuration read or configuration write transaction.

When the P64H translates the Type 1 transaction to a Type 0 transaction on the PCI interface, it performs the following translations to the address:

- Sets the lower 2 address bits on ad[1:0] to 00b
- Decodes the device number and drives the bit pattern on PCI Bus AD[31:16] for the purpose of asserting the device's IDSEL signal
- Leaves unchanged the function number and register number fields

Figure 3. Type 1 to Type 0 Transaction



The P64H can assert up to 16 unique address lines to be used as IDSEL signals for up to 16 devices on the secondary bus, for device numbers ranging from 0 through 15. If IDSEL pin is not asserted to a secondary device, the transaction ends in a master abort. Types 1 to Type 0 configuration read or write transactions are limited to a single 32-bit data transfer.

Type 1 to Type 1 Forwarding

Type 1 to Type 1 transaction forwarding provides a hierarchical configuration mechanism when two or more levels of PCI-to-PCI bridges are used.

When the P64H detects a Type 1 configuration transaction intended for a PCI bus downstream from the secondary bus, the P64H forwards the transaction unchanged to the secondary bus. Ultimately, this transaction is translated to a Type 0 configuration command or to a special cycle transaction by a downstream PCI-to-PCI bridge. Downstream Type 1 to Type 1 forwarding occurs when the following conditions are met during the address phase:

- The lower 2 address bits are equal to 01b.
- The bus number falls in the range defined by the lower limit (exclusive) in the secondary bus number register and the upper limit (inclusive) in the subordinate bus number register.
- The Bus command is a configuration read or write transaction.
- Type 1 to Type 1 configuration write transactions are limited to a single data transfer.

Special Cycles

The Type 1 configuration mechanism is used to generate special cycle transactions in hierarchical PCI systems. Special cycle transactions are ignored by a PCI-to-PCI bridge acting as a target and are not forwarded across the bridge. Special cycle transactions can be generated from Type 1 configuration write transactions in the downstream direction.

The P64H initiates a special cycle on the target bus when a Type 1 configuration write transaction is detected on the initiating bus and the following conditions are met during the address phase:

- The low 2 address bits on AD[1:0]are equal to 01b.
- The device number in address bits AD[15:11] is equal to 11111b.
- The function number in address bits AD[10:8] is equal to 111b.
- The register number in address bits AD[7:2] is equal to 000000b.
- The bus number is equal to the value in the secondary bus number register in configuration space for downstream forwarding or equal to the value in the primary bus number register in configuration space for upstream forwarding.
- The Bus command is a Configuration Write command.

When the P64H initiates the transaction on the target interface, the Bus command is changed from configuration write to special cycle. The address and data are forwarded unchanged. Devices that use special cycles ignore the address and decode only the Bus command. The data phase contains the special cycle message. The transaction is forwarded as a non-posted transaction, but in this case the target response is not forwarded back (because special cycles result in a master abort). Once the transaction is completed on the target bus (through detection of the master abort condition), the P64H responds with TRDY# to the next attempt of the configuration transaction from the initiator. If more than one data transfer is requested, the P64H responds with a target disconnect operation during the first data phase.

3.2.6. 64-Bit Operation

The P64H provides 64-bit extension support on the secondary interfaces. Both 64-bit and 32-bit operations are supported on the PCI interfaces. This section describes how to use the 64-bit extensions. It describes the conditions under which a transaction can be treated as a 64-bit transaction and includes information about how the transaction is forwarded.

64-Bit and 32-Bit Transactions Initiated by the P64H

The P64H requests a 64-bit transaction by asserting REQ64# on the PCI during the address phase. The P64H asserts and deasserts REQ64# during the same cycles in which it asserts and deasserts FRAME#, respectively. Under certain circumstances, the P64H does not use the 64-bit extension when initiating transactions and therefore does not assert REQ64#.

The P64H does not assert REQ64# when any of the following is true:

- Signal REQ64# was not asserted by the PCI Bus central function during reset.
- The P64H is initiating an I/O transaction.
- The P64H is initiating a configuration transaction.
- The P64H is initiating a special cycle transaction.
- A 1- or 2-Dword memory write transaction is being performed.
- A single Dword read transaction is being performed.
- The transaction address initiated on the hub interface is not quad word aligned.

Address Phase of 64-Bit Transactions

When a transaction is a single address cycle (SAC)—that is, the address falls below the 4 GB boundary, and the upper 32 bits of the address are assumed to be zero—AD[63:32] and C/BE[7:4]# are not defined but are driven to valid logic level during the address phase. When the transaction is a dual address cycle (DAC)—that is, the address falls above the 4GB boundary, and the upper 32 bits of the address are nonzero—signals AD[63:32] contain the upper 32 bits of the address for both address phases. Signals C/BE[7:4]# contain the Memory Bus command during both address phases. A 64-bit target then has the opportunity to decode the entire 64-bit address and Bus command after the first address phase. A 32-bit target needs both address phases to decode the full address and Bus command.

Data Phase of 64-Bit Transactions

During memory write transactions, if the P64H has driven REQ64# to indicate it is initiating a 64-bit transfer, during the data phase the P64H drives the following:

- The lower 32 bits of data on AD[31:0]
- The lower four byte enable bits on C/BE[3:0]#
- The higher 32 bits of data on AD[63:32]
- The high four byte enable bits on C/BE[7:4]#

If the P64H detects ACK64# asserted by the target at the same time that it detects DEVSEL# asserted, every data phase then consists of 64 bits and eight byte enable bits.

For write transactions, when the P64H does not detect ACK64# asserted at the same time that it detects DEVSEL# asserted, the P64H redirects the write data that it has on the AD[63:32] bus to AD[31:0] during the second data phase. Similarly, the upper four byte enable bits are redirected to C/BE[3:0]# during the second data phase. All data phases then consist of 32 bits.

For 64-bit memory write transactions that end at an odd DWord boundary, the P64H drives the byte enable bits to 1 during the last data phase. AD[63:32] are then unpredictable but are driven to a valid logic level.

For read transactions, when the P64H has asserted REQ64#, it drives 8 bits of byte enables on C/BE[7:0]#. Because the only read transactions that use the 64-bit extension are prefetchable memory read transactions, the byte enable bits are always zero. Therefore, no special redirection is needed based on the target's assertion or lack of assertion of ACK64#. When the target asserts ACK64# at the same time that it asserts DEVSEL#, all read data transfers then consist of 64 bits and the target drives PAR64, which covers AD[63:32] and C/BE[7:4]#. When the target does not assert ACK64# when it asserts DEVSEL#, all data phases then consist of 32 bits.

64-Bit Transactions Received by the P64H

When the P64H is the target of a transaction and the P64H detects REQ64# asserted during a memory transaction to be forwarded across the bridge, the P64H asserts ACK64# at the same time that it asserts DEVSEL# to indicate its ability to perform 64-bit data transfers. The P64H does not assert ACK64# if any of the following is true:

- REQ64# is not asserted by the initiator.
- The P64H is responding to an I/O transaction.
- The I/O APIC is being accessed.

64-Bit Extension Support during Reset

When the P64H supports a 64-bit interface on its PCI Bus, it drives REQ64# while RSTIN# is asserted to determine whether the PCI 64-bit extension signals are connected on the board. If REQ64# is high, the 64-bit extension signals are not connected, the P64H will always drive the 64-bit extension outputs to have valid logic levels on the inputs, and all PCI transactions are treated as 32-bit transactions.

If REQ64# is low, the 64-bit signals are connected to pull-up resistors on the board and the P64H does not perform any input biasing. In this case, the P64H can treat memory write and prefetchable memory read transactions as 64-bit transactions on the PCI interface, as discussed previously.

3.2.7. Transaction Termination

This section describes how the P64H returns transaction termination conditions to the initiator. Table 15 and Table 16 summarize the termination initiated by either a master or target.

Table 15. Termination Transaction by the Initiator

Termination Types	Description
Normal termination	Normal termination occurs when the initiator deasserts FRAME# at the beginning of the last data phase and deasserts IRDY# at the end of the last data phase in conjunction with either TRDY# or STOP# assertion from the target.
Master abort	Master aborts occur when no target response is detected. When the initiator does not detect a DEVSEL# from the target within five clock cycles after asserting FRAME#, the initiator terminates the transaction with a master abort. If FRAME# is still asserted, the initiator deasserts FRAME# on the next cycle and then deasserts IRDY# on the following cycle. IRDY# must be asserted in the same cycle that FRAME# deasserts. If FRAME# is already deasserted, IRDY# can be deasserted on the next clock cycle following detection of the master abort condition.

Table 16. Termination Transaction by the Target

Termination Types	Description
Normal termination	TRDY# and DEVSEL# asserted in conjunction with FRAME# deasserted and IRDY# asserted.
Target retry	STOP# and DEVSEL# asserted without TRDY# during the first data phase. No data transfers occur during the transaction. This transaction must be repeated.
Target disconnect (with data transfer)	STOP# and DEVSEL# asserted with TRDY#. Signals that this is the last data transfer of the transaction.
Target disconnect (without data transfer)	STOP# and DEVSEL# asserted without TRDY# after previous data transfers have been made. This indicates that no more data transfers will be made during this transaction.
Target abort	STOP# asserted without DEVSEL# and without TRDY#. This indicates that the target will never be able to complete this transaction. DEVSEL# must be asserted for at least one cycle during the transaction before the target abort is signaled.

3.2.7.1. Master Termination Initiated by the P64H

The P64H, as an initiator, uses normal termination if DEVSEL# is returned by the target within five clock cycles of the P64H's assertion of FRAME# on the target bus. As an initiator, the P64H terminates a transaction when the following conditions are met:

- For a posted write transaction, all write data for the transaction is transferred from P64H data buffers to the target.
- For a burst transfer, with the exception of memory write and invalidate transactions, the master latency timer expires and the P64H's bus grant is deasserted.
- The target terminates the transaction with a retry, disconnect, or target abort.

3.2.7.2. Master Abort Received by the P64H

If the P64H initiates a transaction on the target bus and does not detect DEVSEL# returned by the target within five clock cycles of the P64H's assertion of FRAME#, the P64H terminates the transaction with a master abort. The P64H sets the received master abort bit in the status register corresponding to the target bus.

Note: When the P64H performs a Type 1 to special cycle translation, a master abort is the expected termination for the special cycle on the target bus. In this case, the master abort received bit is *not* set, and the Type 1 configuration transaction is disconnected after the first data phase.

3.2.7.3. Target Termination Received by the P64H

When the P64H initiates a transaction on the PCI bus and the target responds with DEVSEL#, the target can end the transaction with one of the following types of termination:

- Normal termination (upon deassertion of FRAME#)
- Target retry
- Target disconnect
- Target abort

The P64H handles the termination in different ways, depending on the type of transaction being performed.

Posted Write Target Termination Response

When the P64H initiates a posted write transaction, the target termination cannot be passed back to the initiator.

Delayed Read Target Termination Response

When the P64H initiates a delayed read transaction, the abnormal target responses can be passed back to the initiator.

Target Termination Initiated by the P64H

The P64H can return a target retry, target disconnect, or target abort to an initiator for reasons other than detection of that condition at the target interface.

Target Retry

The P64H returns a target retry to the initiator when it cannot accept write data or return read data as a result of internal conditions. The P64H returns a target retry to an initiator when any of the following conditions is met:

- The transaction is being entered into the delayed transaction queue.
- The read request has already been queued, but read data is not yet available.
- Data has been read from the target, but it is not yet at the head of the read data queue.
- The delayed transaction queue is full, and the transaction cannot be queued.
- A delayed read request with the same address and Bus command has already been queued.
- A locked sequence is being propagated downstream and the P64H is the target of a PCI transaction.

An I/O write transaction is treated as delayed transaction.

Target Disconnect

The P64H returns a target disconnect to an initiator when one of the following conditions occurs:

- The P64H cannot accept any more write data.
- The P64H has no more read data to deliver.
- The P64H receives a non-linear memory address.

Target Abort

The P64H returns a target abort to an initiator when one of the following conditions occurs:

- I/O cycle with invalid byte enable bits.
- Memory cycles that have any of the bits AD[63:44] set.
- I/O cycles that have any of the bits AD[31:16] set.
- Memory or I/O cycles that were forwarded to hub interface but was aborted.

LOCK Cycle Response

Locked transactions are not supported for cycles targeted for the I/O APIC. The I/O APIC would treat locked cycles as normal cycles. The P64H will perform either a PCI initiated memory cycle or PCI initiated I/O cycle to the hub interface when it establishes a lock on the secondary (PCI) bus.

For PCI initiated memory cycles that are targeted for the hub interface:

- Memory cycles addresses, with any of its AD[63:44] bits set, are aborted on PCI bus without a delayed transaction being established.
- Memory cycles addresses, without any of its AD[63:44] bits set, are retried without a delayed transaction being established.

For PCI initiated I/O cycles to the hub interface:

- P64H will abort the cycle without establishing delayed transaction if the address decoding indicates that the cycle has to be target aborted.
- P64H responds with TRDY# (and STOP# if the master was attempting a burst) if the I/O cycle has an address parity error.
- I/O read cycle without address parity error is retired without delayed transaction being established.
- P64H responds to I/O write cycle with data parity error only by asserting TRDY# (and STOP# if the master was attempting a burst). No delayed transaction is established and data is discarded.
- I/O write cycle without address or data parity are retried. Delayed transactions are not established.

3.3. Transaction Ordering

To maintain data coherency and consistency, the P64H complies with the ordering rules set forth in the *PCI Local Bus Specification* for transactions crossing the bridge. This section describes the ordering rules that control transaction forwarding across the P64H. For a more detailed transaction ordering information, see Appendix E of the *PCI Local Bus Specification*.

3.3.1. Transactions Governed by Ordering Rules

Ordering relationships are established, for the transaction classes listed in Table 17, crossing the P64H:

Table 17. Transaction Ordering Rules

Transaction	Description
Posted write transactions	Comprised of memory write and applies to upstream, downstream and I/O Write downstream only.
Non-posted write request (downstream only)	Non-posted write requests transaction must complete on the target bus before it completes on the initiator bus.
Non-posted write completion (downstream only)	Non-posted write completion transactions have been completed on the target bus and the target response is queued in the P64H buffers.
Delayed read request	Comprised of all memory upstream transactions.
Delayed read completion	Comprised of all memory upstream transactions. Delayed read completion transactions have been completed on the target bus, and the read data has been queued in the P64H read data buffers.

The P64H does not combine or merge write transactions:

- The P64H does not combine separate write transactions into a single write transaction.
- The P64H does not merge bytes on separate masked write transactions to the same DWord address.
- The P64H does not collapse sequential write transactions to the same address into a single write transaction—the *PCI Local Bus Specification* does not permit this combining of transactions.

3.3.2. General Ordering Guidelines

Independent transactions on the primary and secondary buses have a relationship only when those transactions cross the P64H. The following ordering guidelines govern transactions crossing the P64H:

- The transaction's ordering relationship, with respect to other transactions, is determined when the transaction ends with a termination other than target retry.
- Memory writes can be posted in both directions in a bridge. I/O and configuration writes are not posted. Read transactions (e.g., I/O, configuration or Memory) are not posted.
- Requests terminated with target retry can be accepted and completed in any order with respect to other transactions that have been terminated with target retry. If the order of completion of delayed requests is important, the initiator should not start a second delayed transaction until the first one has been completed. If more than one delayed transaction is initiated, the initiator should repeat all the delayed transaction requests, using some fairness algorithm. Repeating a delayed transaction cannot be contingent on completion of another delayed transaction; otherwise, a deadlock can occur.
- Write transactions flowing in one direction have no ordering requirements with respect to write transactions flowing in the other direction. The P64H can accept posted write transactions on both interfaces at the same time, as well as initiate posted write transactions on both interfaces at the same time.
- The acceptance of a posted memory write transaction as a target can never be contingent on the completion of a non-locked, non-posted transaction as a master. This is true of the P64H and must also be true of other bus agents; otherwise, a deadlock can occur.
- The P64H accepts posted write transactions, regardless of the state of completion of any delayed transactions being forwarded across the P64H.

3.3.3. Ordering Rules

Table 18 lists the ordering relationships of all transactions and refers by number to the ordering rules that follow.

Table 18. Ordering Relation Per Transaction

Row Pass	Posted Write	Delayed Read Request	Delayed Write Request	Delayed Read Completion	Delayed Write Completion
Posted Write	No[1]	Yes[5]	Yes[5]	Yes[7]	Yes[7]
Delayed Read Request	No[2]	Yes	Yes	No	No
Delayed Write Request	No[3]	No	No	No	No
Delayed Read Completion	No[4]	Yes	Yes	No	No
Delayed Write Completion	No	Yes[6]	Yes[6]	No	No

The following ordering rules describe the transaction relationships. Each ordering rule is followed by an explanation, and the ordering rules reference numbers are in the above table. These ordering rules apply to posted write transactions, delayed read requests, and delayed read completion transactions crossing the P64H in the same direction. Note that delayed completion transactions cross the P64H in the direction opposite that of the corresponding delayed requests.

1. Posted write transactions must complete on the target bus in the order in which they were received on the initiator bus.
2. A delayed read request traveling in the same direction as a previously queued posted write transaction must push the posted write data ahead of it. The posted write transaction must complete on the target bus before the delayed read request can be attempted on the target bus. The read transaction can be to the same location as the write data. If the read transaction were to pass the write transaction, it would return stale data.
3. A delayed write request cannot pass previously queued posted write data.
4. A delayed read completion must “pull” write data back to the originating bus of the read transaction. In this case, the read data is traveling in the same direction as the write data, and the initiator of the read transaction is on the same side as the target of the write transaction. The posted write transaction must complete to the target before the read data is returned to the initiator. The read transaction can be to a status register of the initiator of the posted write data and therefore should not complete until the write transaction is complete.
5. A posted memory write must be allowed to pass delayed read and write requests to avoid deadlocks.
6. Delayed write completion must be allowed to pass delayed read and write requests to avoid deadlocks.
7. A downstream posted memory write must be allowed to pass delayed read and write completion to avoid deadlocks.

3.4. Error Handling

The P64H checks and generates parity on both the hub interface and the PCI bus. The parity errors must always be reported to some system level software, typically the device driver or the operating system. To support error reporting on the PCI bus, the P64H implements the following:

- PERR# and SERR# signals on the PCI interface
- SERR# special cycle on the hub Interface
- Primary status (offset 06–07h) and secondary status registers (1E–1Fh)
- The device-specific Error command register (offset 90h)
- The device-specific Error status register (offset 92h)

3.4.1. P64H Error Reporting Model

The P64H does not have the PERR# or SERR# signals on the hub interface. It is not capable of directly generating NMI, SMI, or INTR signals. The P64H reports the errors to MCH via a hub interface SERR# special cycle. Once the MCH detects this SERR cycle (from the P64H), it will forward the error condition to the ICH. The ICH then generates an interrupt (NMI, SMI to the processor.)

An address parity error is considered to be catastrophic by the P64H. When the P64H receives an address parity error, it aborts (generate master abort) further data transfers and generates a SERR cycle to the hub interface. Although data parity is not considered as severe and transactions are not aborted, a SERR# special cycle is also generated on the hub interface.

3.4.2. Address Parity Errors

The P64H checks address parity for all transactions on both the hub interface and PCI bus. When the P64H detects an address parity error on the hub interface, the following events occur:

- Determine if the Parity Error Response bit (in CMD register) is set. The P64H reports the address parity error, to the MCH only if this bit is set.
- The P64H sets the Detected Parity Error bit (DPE bit in the PD_STS register).
- Checks to ensure that SERR# Enable bit (SERR_EN bit in the CMD register) is set.
- The P64H signals the MCH via the SERR cycle and sets the Signal System Error bit (SSE bit in the PD_STS register).
- The P64H generates a master abort cycle to the hub interface.

When the P64H detects an address parity error on the PCI interface, the following events occur:

- Determine if the Parity Error Response bit (PER bit in the CMD register) is set. The P64H reports address parity errors to the MCH only if this bit is set. Otherwise, the transaction is accepted normally.
- The P64H sets the Detected Parity Error (DPE bit) in both the PD_STS and SEC_STS registers.
- Checks to ensure that SERR# Enable bit (SERR_EN bit in the CMD register) is set.
- The P64H signals the MCH via the hub the SERR cycle and sets the Signal System Error bit (SSE bit in the PD_STS register).

3.4.3. Data Parity Errors

The following sections describe, for each type of transaction, the sequence of events that occurs when a data parity error is detected and also describes the way in which the parity condition is forwarded across the P64H.

- Configuration Write Transactions to P64H PCI configuration space
- Read Transactions (upstream and downstream)
- Posted Write Transaction

3.4.3.1. Configuration Write Transactions to P64H Configuration Space

When the P64H detects a data parity error during a Type 0 configuration write transaction to P64H configuration space, the following events occur:

- The P64H writes the data to the configuration register.
- The P64H determines if the Parity Error Response bit (PER bit in the CMD register) is set. The P64H reports the address parity, to the MCH, only if this bit is set.
- The P64H sets the Detected Parity Error (DPE bit) in the PD_STS register.
- The P64H checks to ensure that SERR# Enable bit (SERR_EN bit in the CMD register) is set.
- The P64H signals the MCH via the SERR# special cycle and sets the Signal System Error bit (SSE bit in the PD_STS register).

3.4.3.2. Read Transactions (Data Parity)

When the P64H detects a parity error during a read transaction, the target drives data and data parity, and the initiator checks parity and conditionally asserts PERR#. For downstream transactions, when the P64H detects a read data parity error on the PCI bus, the following events occur:

- The P64H sets the Detected Parity Error (DPE bit in the SEC_STS register) in the secondary status register.
- The P64H sets the Data Parity Detected bit in the SEC_STS register, only if the secondary interface Parity Error Response bit is set in the bridge control register.
- Generate SERR# special cycle on the hub interface if the Parity Error Response bit in the CMD register is set.
- The P64H forwards the parity error with the data back, to the initiator, on the hub interface.

For upstream transactions, when the P64H detects a read data parity error on the hub interface, the following events occur:

- The P64H sets the Detected Parity Error (DPE bit in the PD_STS register) in the primary status register.
- The P64H sets the Data Parity Detected bit in the PD_STS register, only if the primary interface Parity Error Response bit is set in the command register.
- The P64H forwards the parity error with the data back to the initiator, on the PCI bus. If the data with the bad parity is prefetched and is not read by the initiator on the PCI bus, the data is discarded and the data with bad parity is not returned to the initiator.
- The P64H completes the transaction as normal.

The P64H returns to the initiator the data and parity that was received from the target. When the initiator detects a parity error on this read data and is enabled to report it, the initiator asserts PERR# two cycles after the data transfer occurs. It is assumed that the initiator takes responsibility for handling a parity error condition; therefore, when the P64H detects PERR# asserted while returning read data to the initiator, the P64H does not take any further action and completes the transaction normally.

3.4.3.3. Posted Write Transactions (Data Parity)

When the P64H (as a target) detects a data parity error on the initiator (hub interface) during downstream posted write transactions, the following events occur:

- The P64H sets the primary interface parity error detected bit in the status register.
- The P64H captures and forwards the parity error condition to the secondary (PCI) bus
- The P64H initiates a SERR# special cycle after checking that the Parity Error Response bit (in the CMD register) is set.
- The P64H completes the transaction as normal.

Similarly, during upstream posted write transactions, when the P64H, responding as a target, detects a data parity error on the initiator (PCI) bus, the following events occur:

- The P64H asserts PERR# two cycles after the data transfer, if the secondary interface Parity Error Response bit is set in the Bridge Control register.
- The P64H sets the secondary interface Parity Error Detected bit in the secondary status register.
- The P64H captures and forwards the bad parity condition to the primary bus.
- The P64H completes the transaction as normal.

During downstream write transactions, when a data parity error is reported on the target PCI bus by the target's assertion of PERR#, the following events occur:

- The P64H sets the Data Parity Detected bit in the secondary status register, if the secondary interface Parity Error Response bit is set in the bridge control register.
- The P64H initiates a SERR# special cycle and sets the Signaled System Error bit in the status register, only if all of the following conditions are met:
  - The SERR# enable bit is set in the command register.
  - The secondary interface parity error response bit is set in the bridge control register.
  - The primary interface parity error response bit is set in the command register.
  - The P64H did not detect the parity error on the PCI bus; that is, the parity error was not forwarded from the PCI bus.
  - No action is taken during upstream write transactions.

3.4.3.4. I/O Write Transaction

During upstream I/O write transactions, the following events occur. This is applicable only when the P64H is the target, detects data parity error on the PCI bus, and detects the secondary interface parity error response bit is set.

- The delayed transaction for I/O write transaction will not be established.
- The P64H generates TRDY# and PERR# two cycles after the data transfer.
- The P64H sets the secondary interface parity error detected bit in the secondary status register.
- The P64H does not forward parity error and completes the transaction.

3.4.4. Master Aborts

The P64H provides two methods of handling master-abort termination when it is the master of a transaction, as controlled by the master-abort mode bit in the Bridge Control register.

In the default case, the master-abort mode bit is 0. The master-abort will not be considered as an error unless it occurs during an exclusive access transaction. When the master-abort mode bit is 1, the P64H considers master abort an error for all transaction types when it is the master. The only exceptions are special cycle and transaction terminated with a master-abort. Also, master-aborts are never reported if they occur during special cycle transaction.

3.4.4.1. Non-Posted Transactions

When the master-abort mode bit is 0, the P64H operated in a PC-compatibility mode. When a non-exclusive read transaction crosses the P64H and is terminated on the destination bus by a master-abort, the bridge returns FFFF FFFFh to the originating master and terminates the read transaction on the originating bus normally (by asserting TRDY#). When a non-posted, non-exclusive write transaction crosses the P64H and is terminated by a master-abort, the bridge completes the write transaction on the originating bus normally (by asserting TRDY#) and discard the write data. If the P64H is forwarding a transaction upstream and the transaction terminates with a master-abort on the *primary* interface, the P64H sets the Received Master-Abort bit in the Status Register.

Similarly, the Received Master-Abort bit is set if the P64H is forwarding the transaction downstream and the transaction terminates with a Master-Abort on the *secondary* interface. The P64H sets the Received Master-Abort bit in the Secondary Status Register.

When the master-abort mode bit is set, the bridge must signal a target-abort to the originating master. In this case, if the P64H is forwarding the transaction upstream and it terminates with a master-abort on the *primary* interface, the P64H will:

- Set Received Master-Abort bit in the Status register
- Terminate the corresponding transaction on the secondary bus by signaling a target-abort
- Set Signaled Target-Abort bit in the Secondary Status register.

Similarly, if the bridge is forwarding the transaction downstream and it terminates with a Master-Abort on the *secondary* interface, the P64H will:

- Set Received Master-Abort bit in the Secondary Status register
- Terminate the corresponding transaction on the primary bus by signaling a target-abort
- Set Signaled Target-Abort bit in the Status register.

3.4.4.2. Posted Write Transactions

When the master-abort mode bit is set and a posted write transaction, forwarded by the P64H, terminates with a master-abort on the destination bus, the P64H:

- Asserts SERR# on the primary interface.
- Sets the Signaled System Error bit in the Status Register (if enabled by the SERR# Enable bit in the Command register).

If the P64H is forwarding the transaction upstream and it terminates with a Master-Abort on the *primary* interface the bridge, it will also set the received Master-Abort bit in the Status register. If the bridge is forwarding the transaction downstream and it terminates with a Master-Abort on the *secondary* interface, the bridge will set the received Master-Abort bit in the Secondary Status register.

3.4.4.3. Exclusive Access Master-Abort

The P64H terminates a non-posted exclusive access transaction on its primary interface with a target-abort when the transaction terminates with Master-Abort while being forwarded to its PCI interface. This is necessary to avoid potential deadlock conditions. Note that exclusive access transactions are supported as downstream transactions only.

3.4.5. System Error (SERR#) Reporting

The P64H uses the SERR# special cycle in the hub interface to report system error conditions. In compliance with the *PCI-to-PCI Bridge Architecture Specification*, the P64H initiates a SERR# special cycle when it detects the PCI SERR# input asserted and the SERR# Enable bit is set in the Bridge Control register. In addition, the P64H also sets the Received System Error bit in the secondary status register. The P64H also initiates a SERR# special cycle for the following:

- Target abort detected during posted write transaction
- Parity error reported on target bus during posted write transaction
- Master time out on delayed transaction

3.4.5.1. PCI SERR# Pin Assertion

When SERR# is sampled asserted, the P64H sets the received system error bit in the secondary status register. It generates the SERR# special-cycle if the following two conditions are met:

- The SERR# forward enable bit is set in the bridge control register, and
- The primary SERR# enable bit is set in the primary command register.

3.4.5.2. Other System Error

The P64H also conditionally initiates the SERR special cycle for the following:

- Target abort detected on the PCI bus, if primary SERR# enable bit is set and SERR# on receiving target abort bit (bit 2, offset 90h) is set.
- Parity error reported on targeted bus during write transactions.
- Master timeout on delayed transaction, if the primary SERR# enable bit is set and SERR# due to timeout enable bit is set.

3.5. Advanced Interrupt Controller (APIC)

The P64H implements a variation of the APIC known as the I/O APIC. Within this document, the terms I/O APIC and APIC are used interchangeably. The I/O APIC resides on the secondary bus (bus #1).

3.5.1. Interrupt Handling

The I/O APIC handles interrupts differently than the 8259. Briefly, these differences are:

1. **Method of Interrupt Transmission:** The I/O APIC transmits interrupts through a three-wire bus, and interrupts are handled without the processor running an interrupt acknowledge cycle.
2. **Interrupt Priority:** The priority of interrupts in the I/O APIC is independent of the interrupt number. For example, interrupt 10 can be given a higher priority than interrupt 3.
3. **More Interrupts:** The I/O APIC in the P64H supports a total of 24 interrupts.
4. **Multiple Interrupt Controllers:** The I/O APIC interrupt transmission protocol has an arbitration phase; P64H allows for multiple I/O APICs in the system with their own interrupt vectors. The P64H I/O APIC must arbitrate for the APIC bus before transmitting its interrupt message.

3.5.2. Interrupt Mapping

The I/O APIC within the P64H supports 24 APIC interrupts. Each interrupt has its own unique vector assigned by software. The interrupt vectors are mapped in accordance to the multi-processor specification.

3.5.3. APIC Bus Functional Description

3.5.3.1. Physical Characteristics of APIC

The APIC bus is a 3-wire synchronous bus connecting all I/O and local APICs. Two of these wires are used for data transmission and one wire is a clock. For bus arbitration, APIC uses only one of the data wires. The bus is logically a wire-OR and electrically an open-drain connection providing for both bus use arbitration and arbitration for lowest priority. The APIC bus speed is 16.6667 MHz.

3.5.3.2. APIC Bus Arbitration

The I/O APIC uses one wire arbitration to win bus ownership. A rotating priority scheme is used for APIC bus 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 will take the winner's arbitration ID and will increment it by one. Arbitration IDs are changed only for messages that are transmitted successfully (except for the Low Priority messages). A message is transmitted successfully if no CS error or acceptance error was reported for that message.

An APIC agent can use two different priority schemes—Normal or EOI. EOI has the highest priority. EOI priority is used to send EOI messages for level interrupts from a local APIC to an I/O APIC. When an agent requests the bus with EOI priority, all other agents requesting the bus with normal priorities will back off.

When P64H detects a bus idle condition on the APIC Bus and has an interrupt to send to the APIC bus, the P64H drives a start cycle to begin arbitration (by driving bit 0 to a 0 on an APICCLK rising edge). The P64H then samples bit 1. If Bit 1 was a 0, then a local APIC started arbitration for an EOI message on the same clock edge that the P64H started arbitration. Thus, the P64H has lost arbitration and stops driving the APIC bus.

If the P64H did not see an EOI message start, it starts transferring its arbitration ID; the ID is located in bits [27:24] of its Arbitration ID register (ARBID). Starting in Cycle 2, through Cycle 5, the P64H tri-states bit 0, and drives bit 1 to a 0 if ARBID[27] is a 1. If ARBID[27] is a 0, the P64H also tri-states bit 1. At the end of each cycle, the P64H samples the state of bit 1 on the APIC bus. If the P64H did not drive bit 1 (ARBID[27] = 0) and it samples a 0, then another APIC agent started arbitration for the APIC bus at the same time as the P64H, and it has higher priority. The P64H stops driving the APIC bus.

Table 19. Arbitration Cycles

Cycle	Bit 1	Bit 0	Comment
1	EOI	0	Bit 1 = 1: Normal, Bit 1 = 0: EOI
2	NOT (ARBID[27])	1	Arbitration ID. If P64H samples a different value than it sent, it loses the arbitration.
3	NOT (ARBID[26])	1	
4	NOT (ARBID[25])	1	
5	NOT (ARBID[24])	1	

3.5.3.3. Bus Message Formats

After bus arbitration, the winner is granted exclusive use of the bus and drives its message. APIC messages come in four formats, determined by the delivery mode bits. These four messages are of different length, and are known by all APICs on the bus through the transmission of the delivery mode bits.

Table 20. APIC Message Formats

Message	# of Cycles	Delivery Mode Bits	Comments
EOI	14	xxx	End of Interrupt transmission from Local APIC to I/O APIC on Level interrupts. EOI is known by the EOI bit at the start of arbitration
Short	21	001, 010, 100, 101, 111	I/O APIC delivery on Fixed, NMI, SMI, Reset, ExtINT, and Lowest Priority with focus processor messages
Lowest Priority	33	001	Transmission of Lowest Priority interrupts when the status field indicates that the processor doesn't have focus
Remote Read	39	011	Message from one Local APIC to another to read registers.

3.5.3.3.1. EOI Message For Level Triggered Interrupts

EOI messages are used by local APICs to send an EOI cycle occurring for a level-triggered interrupt to an I/O APIC. This message is needed so the I/O APIC can differentiate between a new interrupt on the interrupt line versus the same interrupt on the interrupt line. The target of the EOI is given by the local APIC through the transmission of the priority vector (V7 through V0) of the interrupt. When this message is received, the I/O APIC resets the Remote IRR bit for that interrupt. If the interrupt signal is still active after the IRR bit is reset, the I/O APIC treats it as a new interrupt.

Table 21. EOI Message

Cycle	Bit 1	Bit 0	Comments
1	0	0	EOI message
2–5	ARBID	1	Arbitration ID
6	NOT(V7)	NOT(V6)	Interrupt vector bits V7–V0 from redirection table register
7	NOT(V5)	NOT(V4)	
8	NOT(V3)	NOT(V2)	
9	NOT(V1)	NOT(V0)	
10	NOT(C1)	NOT(C0)	Check Sum from Cycles 6–9
11	1	1	Post-amble
12	NOT(A)	NOT(A)	Status Cycle 0
13	NOT(A1)	NOT(A1)	Status Cycle 1
14	1	1	Idle

3.5.3.3.2. Short Message

Short messages are used for the delivery of Fixed, NMI, SMI, Reset, ExtINT, and Lowest Priority with focus processor interrupts. The delivery mode bits (M2–M0) specify the message. All short messages take 21 cycles, including the idle cycle.

Table 22. Short Message

Cycle	Bit 1	Bit 0	Comments
1	1	0	Normal Arbitration
2–5	ARBID	1	Arbitration ID
6	NOT(DM)	NOT(M2)	DM[1] = Destination mode from bit 11 of the redirection table register
7	NOT(M1)	NOT(M0)	M2–M0 = Delivery mode from bits 10:8 of the redirection table register
8	NOT(L)	NOT(TM)	L = Level, TM = Trigger mode
9	NOT(V7)	NOT(V6)	Interrupt vector bits V7–V0 from redirection table register
10	NOT(V5)	NOT(V4)	
11	NOT(V3)	NOT(V2)	
12	NOT(V1)	NOT(V0)	
13	NOT(D7)	NOT(D6)	Destination field from bits 63:56 of redirection table register[1]
14	NOT(D5)	NOT(D4)	
15	NOT(D3)	NOT(D2)	
16	NOT(D1)	NOT(D0)	
17	NOT(C1)	NOT(C0)	Checksum for Cycles 6–16[2]
18	1	1	Post-amble[3]
19	NOT(A)	NOT(A)	Status Cycle 0.
20	NOT(A1)	NOT(A1)	Status Cycle 1.
21	1	1	Idle

NOTES:

1. If DM is 0 (physical mode), then cycles 15 and 16 are the APIC ID and cycles 13 and 14 are sent as 1. If DM is 1 (logical mode), then cycles 13 through 16 are the 8-bit Destination field. The interpretation of the logical mode 8-bit Destination field is performed by the local units using the Destination Format Register. Shorthand's of "all-incl-self" and "all-excl-self" both use physical destination mode and a destination field containing APIC ID value of all ones. The sending APIC knows whether it should (incl) or should not (excl) respond to its own message.
2. The checksum field is the cumulative add (mod 4) of all data bits (DM, M0–3, L, TM, V0–7, D0–7). The APIC driving the message provides this checksum. This, in essence, is the lower two bits of an adder at the end of the message.
3. This cycle allows all APICs to perform various internal computations based on the information contained in the received message. One of the computations takes the checksum of the data received in cycles 6 through 16 and compares it with the value in cycle 18. If any APIC computes a different checksum than the one passed in cycle 17, then that APIC signals an error on the APIC bus ("00") in cycle 19. If this happens, all APICs will assume the message was never sent and the sender must try sending the message again; P64H includes re-arbitrating for the APIC bus. In lowest priority delivery when the interrupt has a focus processor, the focus processor will signal this by driving a "01" during cycle 19. This tells all the other APICs that the interrupt has been accepted, the arbitration is preempted, and short message format is used. Cycle 19 and 20 indicates the status of the message (i.e., accepted, check sum error, retry, or error). Table 23 shows the status signal combinations and their meanings for all delivery modes.

Table 23: APIC Bus Status Cycle Definition

Delivery Mode	A	Comments	A1	Comments
Fixed, EOI	11	Checksum OK	1x	Error
			01	Accepted
			00	Retry
	10	Error	xx	
	01	Error	xx	
	00	Checksum Error	xx	
NMI, SMM, Reset, ExtINT	11	Checksum OK	1x	Error
			01	Accepted
			00	Error
	10	Error	xx	
	01	Error	xx	
	00	Checksum Error	xx	
Lowest Priority	11	Checksum OK: No Focus Processor	1x	Error
			01	End and Retry
			00	Go for Low Priority Arbitration
	10	Error	xx	
	01	Checksum OK: Focus Processor	xx	
	00	Checksum Error	xx	
Remote Read	11	Checksum OK	xx	
	10	Error	xx	
	01	Error	xx	
	00	Checksum Error	xx	

3.5.3.3.3. Lowest Priority Message Without Focus Processor (FP)

This message format is used to deliver an interrupt in the lowest priority mode; in P64H it does not have a Focus Process. Cycles 1 through 21 for this message are the same as for the short message discussed above. Status cycle 19 identifies if there is a Focus processor (10). A status value of 11 in cycle 20 indicates the need for lowest priority arbitration.

Table 24. Lowest Priority Message Without Focus Processor

Cycle	Bit 1	Bit 0	Comments
1	1	0	Normal Arbitration
2–5	ARBID	1	Arbitration ID
6	NOT(DM)	NOT(M2)	DM = Destination mode from bit 11 of the redirection table register
7	NOT(M1)	NOT(M0)	M2–M0 = Delivery mode from bits 10:8 of the redirection table register
8	NOT(L)	NOT(TM)	L = Level, TM = Trigger mode
9	NOT(V7)	NOT(V6)	Interrupt vector bits V7–V0 from redirection table register
10	NOT(V5)	NOT(V4)	
11	NOT(V3)	NOT(V2)	
12	NOT(V1)	NOT(V0)	
13	NOT(D7)	NOT(D6)	Destination field from bits 63:56 of redirection table register
14	NOT(D5)	NOT(D4)	
15	NOT(D3)	NOT(D2)	
16	NOT(D1)	NOT(D0)	
17	NOT(C1)	NOT(C0)	Checksum for Cycles 6–16
18	1	1	Postamble
19	NOT(A)	NOT(A)	Status Cycle 0.
20	NOT(A1)	NOT(A1)	Status Cycle 1.
21	P7	1	Inverted Processor Priority P7–P0
22	P6	1	
23	P5	1	
24	P4	1	
25	P3	1	
26	P2	1	
27	P1	1	
28	P0	1	
29	ArbID3	1	
30	ArbID2	1	
31	ArbID1	1	
32	ArbID0	1	
33	S	S	Status
34	1	1	Idle

NOTES:

1. Cycles 21–28 are used to arbitrate for the lowest priority processor. The processor that takes part in the arbitration drives the processor priority on the bus. Only the local APICs that have "free interrupt slots" participate in the lowest priority arbitration.
2. Cycles 29 through 32 are used to break a tie in case two more processors have lowest priority. The bus arbitration IDs are used to break the tie.

3.5.3.3.4. Remote Read Message

The Remote Read message is used when a local APIC wishes to read the register in another local APIC. The message format is the same as the Short message for the first 21 cycles.

Table 25. Remote Read Message

Cycle	Bit 1	Bit 0	Comments
1	1	0	Normal Arbitration
2–5	ARBID	1	Arbitration ID
6	NOT(DM)	NOT(M2)	DM = Destination mode from bit 11 of the redirection table register
7	NOT(M1)	NOT(M0)	M2–M0 = Delivery mode from bits 10:8 of the redirection table register
8	NOT(L)	NOT(TM)	L = Level, TM = Trigger mode
9	NOT(V7)	NOT(V6)	Interrupt vector bits V7–V0 from redirection table register
10	NOT(V5)	NOT(V4)	
11	NOT(V3)	NOT(V2)	
12	NOT(V1)	NOT(V0)	
13	NOT(D7)	NOT(D6)	Destination field from bits 63:56 of redirection table register
14	NOT(D5)	NOT(D4)	
15	NOT(D3)	NOT(D2)	
16	NOT(D1)	NOT(D0)	
17	NOT(C1)	NOT(C0)	Checksum for Cycles 6–16
18	1	1	Postamble
19	NOT(A)	NOT(A)	Status Cycle 0.
20	NOT(A1)	NOT(A1)	Status Cycle 1.
21	d31	D31	Remote register data 31:0
22	d29	d28	
23	d27	d26	
24	d25	d24	
25	d23	d22	
26	d21	d20	
27	d19	d18	
28	d17	d16	
29	d15	d14	
30	d13	d12	
31	d11	d10	
32	d09	d08	
33	d07	d06	
34	d05	d04	
35	d03	d02	
36	d01	d00	
37	S	S	Data Status: 00 = valid, 11 = invalid
38	C	C	Check Sum for data d31–d00
39	1	1	Idle

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

3.5.4. Boot Interrupt

The P64H contains a capability to ‘OR’ several of its interrupt inputs together to generate a single interrupt output. This is necessary for systems that do not support the APIC (and to boot). The output of the P64H is the BT_INTR# pin. This pin is generated in the following fashion:

- Each interrupt input is compared with its mask.
- If the interrupt is masked in the P64H APIC, the interrupt will be sent from the P64H component.
- If the interrupt is not masked, the interrupt is being used by the P64H APIC and will not be sent from the component.

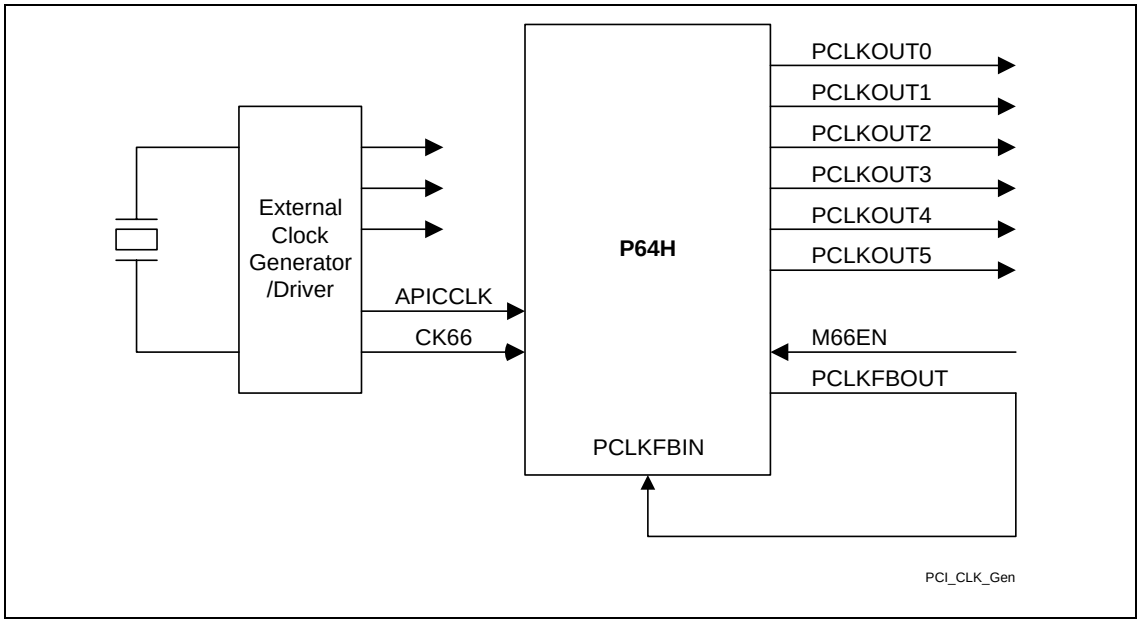
All interrupts that are masked are logically ‘ORed’ together to produce an active low output. If any of the non-masked interrupts are low, the output is low.

3.6. Clocking and Reset

3.6.1. Clocking

The CLK66 clock signal is provided by an external clock generator and is used to run the hub interface. The P64H is responsible for generating the PCI clock (PCLKOUT) to its PCI bus. The P64H includes the M66EN signal; an input to the P64H from the PCI bus. If M66EN is asserted (high), it indicates 66 MHz operation. This signal must be pulled-down (ground) to indicate 33 MHz. The entire PCI bus will operate at 33 MHz if a 33 MHz device is detected. The PCI bus clock speed is dependent on the M66EN input.

Figure 4. PCI Clock Generation and Distribution



The phase of the internal PCI clock is matched to the internal host clock. The internal PCI clock is output through the PCLKOUT pins. One of the clock driver outputs (PCLKFBOUT) must be fed back to the P64H PCLKFBIN; this allows the P64H external PCI clock to phase-lock to the internal PCI clock.

3.6.2. Reset

3.6.2.1. Reset In (RSTIN#)

This signal is connected to the ICH PCIRST# output and has two sources. When asserted, this signal asynchronously resets the P64H. The ICH asserts its PCIRST# during power-up or when a hard reset sequence is initiated through the RC (CF9h) register in the ICH. The ICH PCIRST# is driven inactive a minimum of 1 ms after PWROK is driven active. The ICH PCIRST# is driven for a minimum of 1 ms when initiated through the RC register.

3.6.2.2. Secondary Bus Reset

A PCI reset is generated by the Secondary Bus Reset bit in the Bridge control (BRIDGE_CNT) register. This bit controls the reset on the P64H PCI Bus. If set to 1, the P64H asserts its PCIRST# and sets its data buffers and PCI bus to the reset condition. The hub interface and the configuration registers are not affected by the assertion of PCIRST#.

3.7. Reference Power Voltage

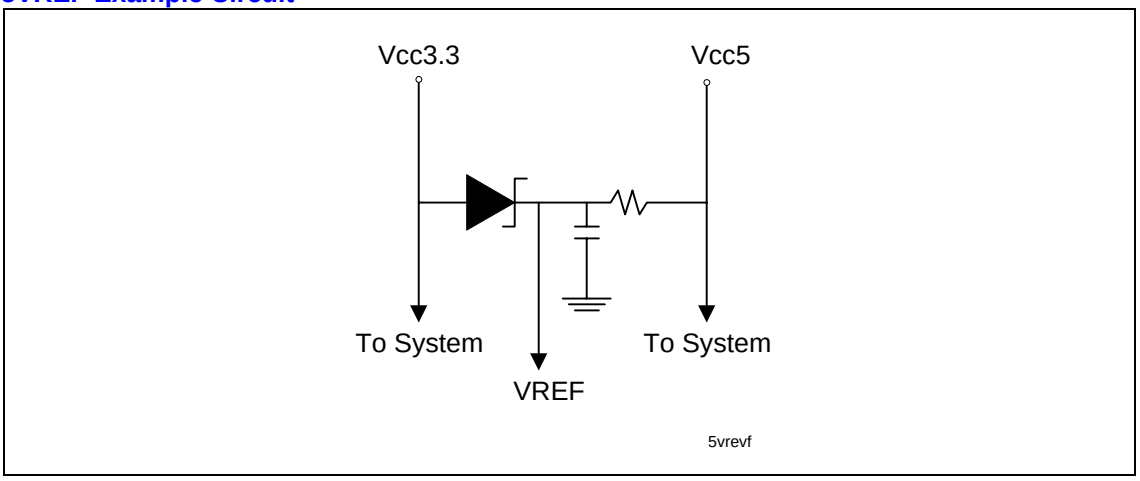
The P64H provides three 5V-reference voltage pins: VCC5REF (2) and PVCC5REF (1). The VCC5REF pins provide 5V voltage reference to the RSTIN#, IRQs, TEST#, and BT_INT#. The PVCC5REF provides 5V reference voltage for the PCI interface. Note that the PCLKIN pin is never 5V tolerant and should have a 3.3V input only.

The VCC5REF pins can be tied to a 5V source even if the interface will be 3.3V only. If the system designer can guarantee that all signals to the P64H are 3.3V only, then the VCC5REF pins can be tied to a 3.3V source. This is also applicable for the PVCC5REF. PVCC5REF should be set at 5V if the design supports 33 MHz, 5V tolerant PCI devices. Otherwise, these signals can be tied to a 3.3V source.

3.7.1. 5V Reference Requirements

The 5VREF reference voltage is for 5V tolerance on inputs in the P64H. 5Vref must be power-up before or simultaneous to Vcc3.3. It must also power-down after or simultaneous to Vcc3.3. Figure 5 shows an example circuit that may be used to ensure the proper 5VREF sequencing.

Figure 5. 5VREF Example Circuit



This implementation is not necessary if the design will only be 3.3V. In an 3.3V-only environment, PVCC5REF and VCC5REF signals can be tied together.

This page is intentionally left blank.

4. Ballout and Package Information

4.1. P64H Ball Assignment

This ballout diagram is a topside view (through the top of the package to the balls).

Figure 6. P64H Ballout (Top View, Left Side)

	1	2	3	4	5	6	7	8
A	VSS	AD7	M66EN	AD14	CBE1#	NC	DEVSEL#	CBE2#
B	AD2	AD6	AD9	AD13	AD15	VSS	STOP#	FRAME#
C	AD1	AD5	VSS	AD12	NC	SERR#	PLOCK#	IRDY#
D	AD0	AD3	CBE0#	AD10	NC	PAR	PERR#	TRDY#
E	PCLKFBIN	AD4	AD8	AD11	VCC3_3	NC	VCC3_3	
F	ACK64#	VSS	REQ64#	VCC5REF	VSS			
G	CBE7#	CBE6#	CBE5#	PAR64	CBE4#			
H	AD63	AD62	AD61	AD60				VSS
J	AD59	AD58	AD57	AD56	VCC3_3			VSS
K	AD55	VSS	AD54	AD53				VSS
L	AD52	AD51	AD50	AD49	VSS			
M	AD48	AD47	AD46	AD45	VCC3_3			
N	AD44	AD43	AD42	AD41	RSV10	VCC3_3	VCC1_8	
P	AD40	VSS	AD39	AD38	RSV12	VSS	PVCC5REF	VCC1_8
R	AD37	AD36	AD35	AD34	RSV13	RSV8	RSV1	RSTIN#
T	AD33	AD32	RSV6	RSV14	RSV11	RSV7	VSS	HLB0
U	VSS	RSV4	RSV5	RSV15	RSV8	RSV3	RSV2	HLB1
	1	2	3	4	5	6	7	8

Figure 7. P64H Ballout (Top View, Right Side)

9	10	11	12	13	14	15	16	17	
AD18	AD22	AD25	AD28	AD31	REQ4#	REQ0#	GNT3#	VSS	A
VSS	AD21	AD24	VSS	AD30	REQ3#	GNT5#	GNT2#	GNT0#	B
AD17	AD20	CBE3#	AD27	AD29	REQ2#	GNT4#	GNT1#	PCIRST#	C
AD16	AD19	AD23	AD26	REQ5#	REQ1#	PCLKOUT6	VSS	PCLKOUT5	D
VCC3_3		VCC3_3	VSS	VCC3_3	VSS	PCLKOUT4	PCLKOUT3	PCLKOUT2	E
				VSS	VCC3_3	PCLKOUT1	VSS	PCLKOUT0	F
				VSS	BT_INTR#	APICD1	APICD0	IRQ23	G
VSS	VSS				IRQ22	IRQ21	IRQ20	VCC5REF	H
VSS	VSS			VCC3_3	IRQ19	IRQ18	TEST#	IRQ17	J
VSS	VSS				IRQ16	IRQ15	IRQ14	APCICLK	K
				VCC3_3	IRQ13	IRQ12	IRQ11	IRQ10	L
				VCC3_3	IRQ9	IRQ8	VSS	IRQ7	M
VCC1_8		VCC1_8	VCC3_3	VSS	IRQ6	IRQ5	IRQ4	IRQ3	N
VSS	VCC1_8	HLB5	VSS	HLB8	HLB11	IRQ2	IRQ1	IRQ0	P
HL2	HLB_STB0	HLB6	HLB17	HLB9	HLB_STB1#	VSS	VSS	CLK66	R
HLB3	VSS	HLB7	HLB19	VSS	HLB_STB1	HLB13	HUBREF	HLBRCOMP	T
HLB_STB0#	HLB4	HLB18	HLB16	HLB10	HLB12	HLB14	HLB15	VSS	U
9	10	11	12	13	14	15	16	17	

Table 26. P64H Alphabetical Ballout Assignment

Signal	Ball #
ACK64#	F1
AD0	D1
AD1	C1
AD2	B1
AD3	D2
AD4	E2
AD5	C2
AD6	B2
AD7	A2
AD8	E3
AD9	B3
AD10	D4
AD11	E4
AD12	C4
AD13	B4
AD14	A4
AD15	B5
AD16	D9
AD17	C9
AD18	A9
AD19	D10
AD20	C10
AD21	B10
AD22	A10
AD23	D11
AD24	B11
AD25	A11
AD26	D12
AD27	C12
AD28	A12
AD29	C13
AD30	B13
AD31	A13
AD32	T2
AD33	T1

Signal	Ball #
AD34	R4
AD35	R3
AD36	R2
AD37	R1
AD38	P4
AD39	P3
AD40	P1
AD41	N4
AD42	N3
AD43	N2
AD44	N1
AD45	M4
AD46	M3
AD47	M2
AD48	M1
AD49	L4
AD50	L3
AD51	L2
AD52	L1
AD53	K4
AD54	K3
AD55	K1
AD56	J4
AD57	J3
AD58	J2
AD59	J1
AD60	H4
AD61	H3
AD62	H2
AD63	H1
APCICLK	K17
APICD0	G16
APICD1	G15
BT_INTR#	G14
CBE0#	D3

Signal	Ball #
CBE1#	A5
CBE2#	A8
CBE3#	C11
CBE4#	G5
CBE5#	G3
CBE6#	G2
CBE7#	G1
CLK66	R17
DEVSEL#	A7
FRAME#	B8
GNT0#	B17
GNT1#	C16
GNT2#	B16
GNT3#	A16
GNT4#	C15
GNT5#	B15
HLB_STB0	R10
HLB_STB0#	U9
HLB_STB1	T14
HLB_STB1#	R14
HLB0	T8
HLB1	U8
HLB2	R9
HLB3	T9
HLB4	U10
HLB5	P11
HLB6	R11
HLB7	T11
HLB8	P13
HLB9	R13
HLB10	U13
HLB11	P14
HLB12	U14
HLB13	T15
HLB14	U15

Signal	Ball #
HLB15	U16
HLB16	U12
HLB17	R12
HLB18	U11
HLB19	T12
HLBRCOMP	T17
HUBREF	T16
IRDY#	C8
IRQ0	P17
IRQ1	P16
IRQ2	P15
IRQ3	N17
IRQ4	N16
IRQ5	N15
IRQ6	N14
IRQ7	M17
IRQ8	M15
IRQ9	M14
IRQ10	L17
IRQ11	L16
IRQ12	L15
IRQ13	L14
IRQ14	K16
IRQ15	K15
IRQ16	K14
IRQ17	J17
IRQ18	J15
IRQ19	J14
IRQ20	H16
IRQ21	H15
IRQ22	H14
IRQ23	G17
M66EN	A3
NC	A6
NC	C5

Signal	Ball #
NC	D5
NC	E6
PAR	D6
PAR64	G4
PCIRST#	C17
PCLKFBIN	E1
PCLKOUT0	F17
PCLKOUT1	F15
PCLKOUT2	E17
PCLKOUT3	E16
PCLKOUT4	E15
PCLKOUT5	D17
PCLKOUT6	D15
PERR#	D7
PLOCK#	C7
PVCC5REF	P7
REQ0#	A15
REQ1#	D14
REQ2#	C14
REQ3#	B14
REQ4#	A14
REQ5#	D13
REQ64#	F3
RSTIN#	R8
RSV1	R7
RSV2	U7
RSV3	U6
RSV4	U2
RSV5	U3
RSV6	T3
RSV7	T6
RSV8	R6
RSV9	U5
RSV10	N5
RSV11	T5
RSV12	P5

Signal	Ball #
RSV13	R5
RSV14	T4
RSV15	U4
SERR#	C6
STOP#	B7
TEST#	J16
TRDY#	D8
VCC1_8	N7
VCC1_8	N9
VCC1_8	N11
VCC1_8	P8
VCC1_8	P10
VCC3_3	E7
VCC3_3	E9
VCC3_3	E11
VCC3_3	J5
VCC3_3	J13
VCC3_3	L13
VCC3_3	M5
VCC3_3	E5
VCC3_3	E13
VCC3_3	F14
VCC3_3	M13
VCC3_3	N6
VCC3_3	N12
VCC5REF	F4
VCC5REF	H17
VSS	A1
VSS	A17
VSS	B6
VSS	B9
VSS	B12
VSS	C3
VSS	D16
VSS	E12
VSS	E14

Signal	Ball #
VSS	F2
VSS	F5
VSS	F13
VSS	F16
VSS	G13
VSS	H8
VSS	H9
VSS	H10
VSS	J8
VSS	J9
VSS	J10
VSS	K2
VSS	K8
VSS	K9
VSS	K10
VSS	L5
VSS	M16
VSS	N13
VSS	P2
VSS	P6
VSS	P9
VSS	P12
VSS	R15
VSS	R16
VSS	T7
VSS	T10
VSS	T13
VSS	U1
VSS	U17

4.2. Package Specification

This section shows the mechanical dimensions for the P64H. The package is a 241 Ball Grid Array (BGA).

Figure 8. Package Dimensions (241-Ball BGA) — Top and Side Views

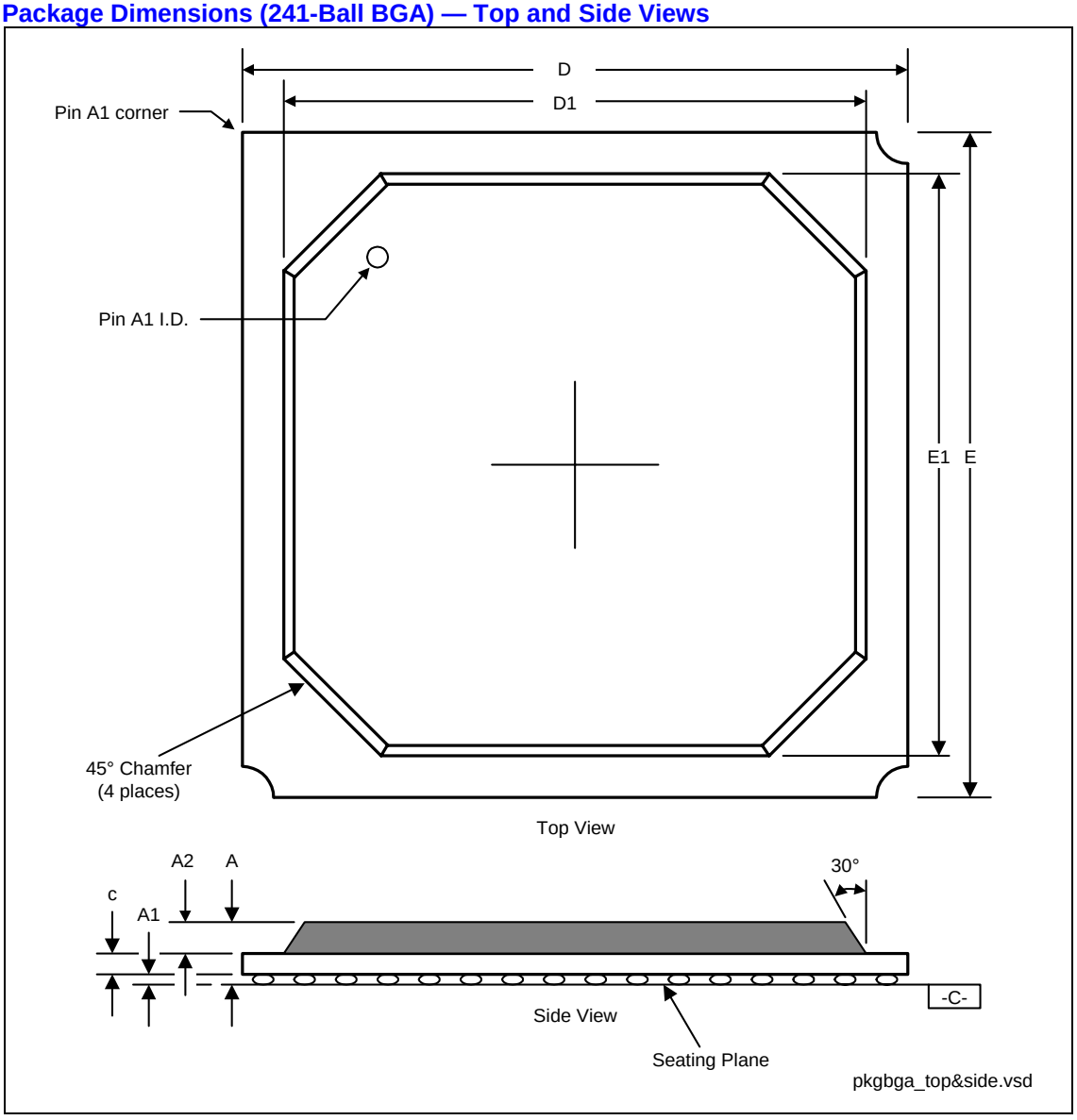


Figure 9. Package Dimensions (241-Ball BGA) — Bottom View

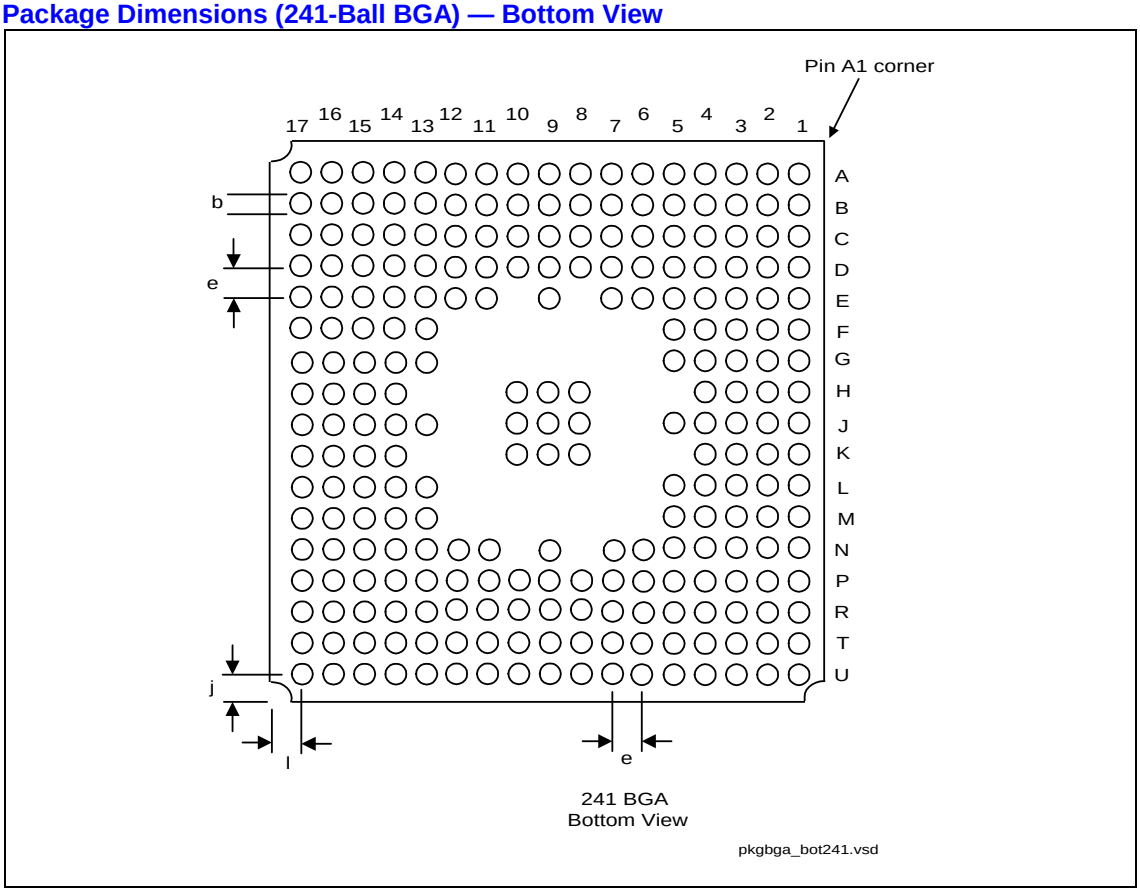


Table 27. BGA Package Dimensions (241-Ball BGA)

Symbol	Min	Nominal	Max	Units	Note
A	2.19	2.38	2.57	mm	
A1	0.50	0.60	0.70	mm	
A2	1.12	1.17	1.22	mm	
D	22.80	23.00	23.20	mm	
D1	19.25	19.50	19.75	mm	
E	22.80	23.00	23.20	mm	
E1	19.25	19.50	19.75	mm	
e	1.27 (solder ball pitch)			mm	
l	1.34 REF.			mm	
J	1.34 REF.			mm	
M	17 x 17 Matrix			mm	
b^2	0.60	0.75	0.90	mm	
c	0.57	0.61	0.65	mm	

NOTES:

1. All dimensions and tolerances conform to ANSI Y14.5-1982
2. Dimension is measured at maximum solder ball diameter parallel to primary datum (-C-)
3. Primary Datum (-C-) and seating plane are defined by the spherical crowns of the solder balls

This page is intentionally left blank.

5. Testability

The P64H supports the following test modes:

Number of Clocks TEST# Driven Low	Test Mode
< 2	No Test Mode Selected
2–5	Reserved. DO NOT ATTEMPT
6	Tri-State
7	NAND Trees
8–26	Reserved. DO NOT ATTEMPT

5.1. Tri-State Mode

The tri-state test mode is activated by asserting the TEST# signal low for six clocks. All outputs and bi-directional pins are tri-stated, including the NAND tree outputs. The tri-state test mode is active during the NAND tree test mode.

5.2. NAND Tree Mode

The P64H has 5 NAND Tree chains. This test mode is activated by asserting the TEST# signals (low) for seven clocks. This test mode can be used to check the connectivity of the pins. The individual NAND chain can be observed with the IRQ pins:

NAND Chain	IRQ Pin
0	IRQ12
1	IRQ13
2	IRQ14
3	IRQ15
4	IRQ17

To perform the NAND tree test, drive all NAND tree inputs pins to 0 and the output of the NAND tree will be a 1 in the order shown below. The corresponding IRQ will reflect the last signal of the NAND tree. Starting at the last signal in the NAND tree, drive a 1 to each pin one at a time. The NAND tree output (IRQXX) will toggle between 0 and 1.

Table 28. NAND Tree Chains

NAND Tree 0	NAND Tree 1	NAND Tree 2	NAND Tree 3	NAND Tree 4	Not In NAND Chain
GNT1#	AD2	RSV14	IRQ6	PCLKOUT0	CLK66
PCIRST#	AD5	RSV15	IRQ2	PCLKOUT1	TEST#
GNT0#	AD1	RSV10	IRQ9	PCLKOUT2	RSTIN#
REQ1#	AD8	RSV12	IRQ5	PCLKOUT3	IRQ12
GNT4#	AD3	RSV13	IRQ1	PCLKOUT4	IRQ13
GNT5v	AD0	RSV11	IRQ8	PCKLOUT5	IRQ14
GNT2#	PCLKFBIN	RSV9	IRQ4	PCLKFBOUT	IRQ15
REQ2v	AD4	RSV8	IRQ0		IRQ17
REQ5#	REQ64#	RSV7	IRQ16		
REQ3#	C/BE4#	RSV3	IRQ3		
REQ0#	PAR64	RSV1	IRQ7		
AD29	ACK64#	RSV2	IRQ11		
AD26	C/BE5#	HLB0	IRQ10		
GNT3#	C/BE6#	HLB1	IRQ19		
REQ4#	C/BE7#	HLB2	APICCLK		
AD30	AD62	HLB3	IRQ18		
AD27	AD60	HLB_STB0#	IRQ22		
AD31	AD61	HLB_STB0	IRQ21		
AD28	AD63	HLB4	IRQ20		
AD23	AD59	HLB5	IRQ23		
C/BE3#	AD58	HLB6	APICD1		
AD24	AD57	HLB7	APICD0		
AD25	AD56	HLB18	BT_INTR#		
AD19	AD55	HLB17			
AD20	AD52	HLB19			
AD17	AD54	HLB16			
AD21	AD51	HLB8			
AD22	AD48	HLB9			
AD18	AD53	HLB10			
AD16	AD50	HLB11			
C/BE2#	AD47	HLB_STB1#			
FRAME#	AD44	HLB_STB1			
IRDY#	AD46	HLB12			
DEVSEL#	AD40	HLB13			
STOP#	AD49	HLB14			

NAND Tree 0	NAND Tree 1	NAND Tree 2	NAND Tree 3	NAND Tree 4	Not In NAND Chain
TRDY#	AD43	HLB15			
LOCK#	AD37	HLBRCOMP			
C/BE1#	AD42				
PERR#	AD45				
SERR#	AD36				
AD14	AD33				
AD15	AD39				
PAR	AD32				
AD13	AD41				
M66EN	AD38				
AD9	AD35				
AD12	AD34				
AD7	RSV4				
AD10	RSV6				
AD6	RSV5				
C/BE0#					
AD11					