OBSOLETE - PART DISCONTINUED

PI7C8154A

2-Port PCI-to-PCI Bridge

REVISION 1.02

3545 North 1st Street, San Jose, CA 95134
Telephone: 1-877-PERICOM, (1-877-737-4266)
Fax: 408-435-1100
Email: solutions@pericom.com
Internet: http://www.pericom.com

LIFE SUPPORT POLICY

Pericom Semiconductor Corporation's products are not authorized for use as critical components in life support devices or systems unless a specific written agreement pertaining to such intended use is executed between the manufacturer and an officer of PSC.

1) Life support devices or system are devices or systems which:
   a) Are intended for surgical implant into the body or
   b) Support or sustain life and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2) A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness. Pericom Semiconductor Corporation reserves the right to make changes to its products or specifications at any time, without notice, in order to improve design or performance and to supply the best possible product. Pericom Semiconductor does not assume any responsibility for use of any circuitry described other than the circuitry embodied in a Pericom Semiconductor product. The Company makes no representations that circuitry described herein is free from patent infringement or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent, patent rights or other rights, of Pericom Semiconductor Corporation.

REVISION HISTORY

Date	Revision Number	Description
07/10/04	0.03	Initial release of preliminary specification
07/26/04	1.00	Initial release of specification to the web Updated Power Dissipation in section 17.9 Updated T_{DELAY} in sections 17.4 and 17.5 Revised V_{IH} parameter in section 17.2
11/24/09	1.01	Updated DC Specifications Updated the Ambient Temperature to be Industrial Temp Compliant
12/03/09	1.02	Added Appendix with Intel 21154 Pinout Comparison

This page intentionally left blank.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

Product Description

The PI7C8154A is Pericom Semiconductor's PCI-to-PCI Bridge, designed to be fully compliant with the 64-bit, 66MHz implementation of the *PCI Local Bus Specification, Revision 2.2.* The PI7C8154A supports synchronous bus transactions between devices on the Primary Bus and the Secondary Buses operating up to 66MHz. The primary and secondary buses can also operate in concurrent mode, resulting in added increase in system performance.

Product Features

- 64-bit Primary and Secondary Ports run up to 66MHz
- Compliant with the *PCI Local Bus Specification,* Revision 2.2
- Compliant with *PCI-to-PCI Bridge Architecture Specification*, Revision 1.1.
  - All I/O and memory commands
  - Type 1 to Type 0 configuration conversion
  - Type 1 to Type 1 configuration forwarding
  - Type 1 configuration write to special cycle conversion
- Compliant with the *PCI Power Management Specification,* Revision 1.1
- Provides internal arbitration for nine secondary bus masters
  - Programmable 2-level priority arbiter
- Supports serial EEPROM interface for register auto-load and VPD access
- *Dynamic Prefetching Control*
- Supports posted write buffers in all directions
- 512 byte upstream posted memory write
- 512 byte downstream posted memory write
- 1024 byte upstream read data buffer
- 1024 byte downstream read data buffer
- Enhanced address decoding
- 32-bit I/O address range
- 32-bit memory-mapped I/O address range
- 64-bit prefetchable address range
- IEEE 1149.1 JTAG interface support
- Industrial temperature range -40°C to 85°C
- 3.3V and 5V signaling
- 304-pin PBGA package
  - Pb-free & Green available

1 SIGNAL DEFINITIONS

1.1 SIGNAL TYPES

Signal Type	Description
I	Input Only
O	Output Only
P	Power
TS	Tri-State bi-directional
STS	Sustained Tri-State. Active LOW signal must be pulled HIGH for 1 cycle when deasserting.
OD	Open Drain

1.2 SIGNALS

Note: Signal names that end with "#" are active LOW.

1.2.1 PRIMARY BUS INTERFACE SIGNALS

Name	Pin #	Type	Description
P_AD[31:0]	U2, U4, U1, V2, V1, V3, W2, W1, W4, Y3, AA1, AA3, Y4, AB3, AA4, Y5, AB8, AA8, AC9, AB9, AA9, AC10, AA10, Y11, AB11, AA11, AA12, AB12, AB13, AA13, Y13, AA14	TS	**Primary Address / Data:** Multiplexed address and data bus. Address is indicated by P_FRAME# assertion. Write data is stable and valid when P_IRDY# is asserted and read data is stable and valid when P_TRDY# is asserted. Data is transferred on rising clock edges when both P_IRDY# and P_TRDY# are asserted. During bus idle, bridge drives P_AD[31:0] to a valid logic level when P_GNT# is asserted.
P_CBE[3:0]	Y2, AB4, AA7, AC11	TS	**Primary Command/Byte Enables:** Multiplexed command field and byte enable field. During address phase, the initiator drives the transaction type on these pins. After that, the initiator drives the byte enables during data phases. During bus idle, bridge drives P_CBE[3:0] to a valid logic level when P_GNT# is asserted.
P_PAR	AB7	TS	**Primary Parity.** P_PAR is even parity of P_AD[31:0] and P_CBE[3:0] (i.e. an even number of 1's). P_PAR is valid and stable one cycle after the address phase (indicated by assertion of P_FRAME#) for address parity. For write data phases, P_PAR is valid one clock after P_IRDY# is asserted. For read data phase, P_PAR is valid one clock after P_TRDY# is asserted. Signal P_PAR is tri-stated one cycle after the P_AD lines are tri-stated. During bus idle, BRIDGE drives P_PAR to a valid logic level when P_GNT# is asserted.
P_FRAME#	AA5	STS	**Primary FRAME (Active LOW).** Driven by the initiator of a transaction to indicate the beginning and duration of an access. The de-assertion of P_FRAME# indicates the final data phase requested by the initiator. Before being tri-stated, it is driven to a de-asserted state for one cycle.

Name	Pin #	Type	Description
P_IRDY#	AC5	STS	**Primary IRDY (Active LOW).** Driven by the initiator of a transaction to indicate its ability to complete current data phase on the primary side. Once asserted in a data phase, it is not de-asserted until the end of the data phase. Before tri-stated, it is driven to a de-asserted state for one cycle.
P_TRDY#	AB5	STS	**Primary TRDY (Active LOW).** Driven by the target of a transaction to indicate its ability to complete current data phase on the primary side. Once asserted in a data phase, it is not de-asserted until the end of the data phase. Before tri-stated, it is driven to a de-asserted state for one cycle.
P_DEVSEL#	AA6	STS	**Primary Device Select (Active LOW).** Asserted by the target indicating that the device is accepting the transaction. As a master, bridge waits for the assertion of this signal within 5 cycles of P_FRAME# assertion; otherwise, terminate with master abort. Before tri-stated, it is driven to a de-asserted state for one cycle.
P_STOP#	AC6	STS	**Primary STOP (Active LOW).** Asserted by the target indicating that the target is requesting the initiator to stop the current transaction. Before tri-stated, it is driven to a de-asserted state for one cycle.
P_LOCK#	AB6	I	**Primary LOCK (Active LOW).** Asserted by an initiator, one clock cycle after the first address phase of a transaction, attempting to perform an operation that may take more than one PCI transaction to complete.
P_IDSEL	Y1	I	**Primary ID Select.** Used as a chip select line for Type 0 configuration access to bridge configuration space.
P_PERR#	AC7	STS	**Primary Parity Error (Active LOW).** Asserted when a data parity error is detected for data received on the primary interface. Before being tri-stated, it is driven to a de-asserted state for one cycle.
P_SERR#	Y7	OD	**Primary System Error (Active LOW).** Can be driven LOW by any device to indicate a system error condition. Bridge drives this pin on: ▪ Address parity error ▪ Posted write data parity error on target bus ▪ Secondary S_SERR# asserted ▪ Master abort during posted write transaction ▪ Target abort during posted write transaction ▪ Posted write transaction discarded ▪ Delayed write request discarded ▪ Delayed read request discarded ▪ Delayed transaction master timeout This signal requires an external pull-up resistor for proper operation.
P_REQ#	U3	TS	**Primary Request (Active LOW):** This is asserted by BRIDGE to indicate that it wants to start a transaction on the primary bus. Bridge de-asserts this pin for at least 2 PCI clock cycles before asserting it again.
P_GNT#	R2	I	**Primary Grant (Active LOW):** When asserted, PI7C8154A can access the primary bus. During idle and P_GNT# asserted, bridge will drive P_AD, P_CBE, and P_PAR to valid logic levels.
P_RESET#	R3	I	**Primary RESET (Active LOW):** When P_RESET# is active, all PCI signals should be asynchronously tri-stated.

Name	Pin #	Type	Description
P_M66EN	AB10	I	**Primary Interface 66MHz Operation.** This input is used to specify if bridge is capable of running at 66MHz. For 66MHz operation on the Primary bus, this signal should be pulled "HIGH". For 33MHz operation on the Primary bus, this signal should be pulled "LOW". In this condition, S_M66EN will be driven "LOW", forcing the secondary bus to run at 33MHz also.

1.2.2 PRIMARY BUS INTERFACE SIGNALS – 64-BIT EXTENSION

Name	Pin #	Type	Description
P_AD[63:32]	AA16, AB16, AA17, AB17, Y17, AB18, AC18, AA18, AC19, AA19, AB20, Y19, AA20, AB21, AC21, AA21, Y20, AA23, Y21, W20, Y23, W21, W23, W22, V21, V23, V22, U23, U20, U22, T23, T22	TS	**Primary Upper 32-bit Address / Data:** Multiplexed address and data bus providing an additional 32 bits to the primary. When a dual address command is used and P_REQ64# is asserted, the initiator drives the upper 32 bits of the 64-bit address. Otherwise, these bits are undefined and driven to valid logic levels. During the data phase of a transaction, the initiator drives the upper 32 bits of the 64-bit write data, or the target drives the upper 32 bits of the 64-bit read data, when P_REQ64# and P_ACK64# are both asserted. Otherwise, these bits are pulled up to a valid logic level through external resistors.
P_CBE[7:4]	AA15, AB15, Y15, AC15	TS	**Primary Upper 32-bit Command/Byte Enables:** Multiplexed command field and byte enable field. During address phase, when the dual address command is used and P_REQ64# is asserted, the initiator drives the transaction type on these pins. Otherwise, these bits are undefined, and the initiator drives a valid logic level onto the pins. For read and write transactions, the initiator drives these bits for the P_AD[63:32] data bits when P_REQ64# and P_ACK64# are both asserted. When not driven, these bits are pulled up to a valid logic level through external resistors.
P_PAR64	T21	TS	**Primary Upper 32-bit Parity:** P_PAR64 carries the even parity of P_AD[63:32] and P_CBE[7:4] for both address and data phases. P_PAR64 is driven by the initiator and is valid 1 cycle after the first address phase when a dual address command is used and P_REQ64# is asserted. P_PAR64 is valid 1 clock cycle after the second address phase of a dual address transaction when P_REQ64# is asserted. P_PAR64 is valid 1 cycle after valid data is driven when both P_REQ64# and P_ACK64# are asserted for that data phase. P_PAR64 is driven by the device driving read or write data 1 cycle after the P_AD lines are driven. P_PAR64 is tri-stated 1 cycle after the P_AD lines are tri-stated. Devices receive data sample P_PAR64 as an input to check for possible parity errors during 64-bit transactions. When not driven, P_PAR64 is pulled up to a valid logic level through external resistors.

Name	Pin #	Type	Description
P_REQ64#	AC14	STS	**Primary 64-bit Transfer Request:** P_REQ64# is asserted by the initiator to indicate that the initiator is requesting a 64-bit data transfer. P_REQ64# has the same timing as P_FRAME#. When P_REQ64# is asserted LOW during reset, a 64-bit data path is supported. When P_REQ64# is HIGH during reset, bridge drives P_AD[63:32], P_CBE[7:4], and P_PAR64 to valid logic levels. When deasserting, P_REQ64# is driven to a deasserted state for 1 cycle and then sustained by an external pull-up resistor.
P_ACK64#	AB14	STS	**Primary 64-bit Transfer Acknowledge:** P_ACK64# is asserted by the target only when P_REQ64# is asserted by the initiator to indicate the target's ability to transfer data using 64 bits. P_ACK64# has the same timing as P_DEVSEL#. When deasserting, P_ACK64# is driven to a deasserted state for 1 cycle and then is sustained by an external pull-up resistor.

1.2.3 SECONDARY BUS INTERFACE SIGNALS

Name	Pin #	Type	Description
S_AD[31:0]	C3, A3, B3, C4, A4, B4, C5, B5, A6, A7, D7, B7, A8, B8, C8, A9, C13, B13, A13, D13, C14, B14, C15, B15, C16, B16, C17, B17, D17, A17, B18, A18	TS	**Secondary Address/Data:** Multiplexed address and data bus. Address is indicated by S_FRAME# assertion. Write data is stable and valid when S_IRDY# is asserted and read data is stable and valid when S_IRDY# is asserted. Data is transferred on rising clock edges when both S_IRDY# and S_TRDY# are asserted. During bus idle, bridge drives S_AD[31:0] to a valid logic level when S_GNT# is asserted respectively.
S_CBE[3:0]	C6, D9, C12, A15	TS	**Secondary Command/Byte Enables:** Multiplexed command field and byte enable field. During address phase, the initiator drives the transaction type on these pins. The initiator then drives the byte enables during data phases. During bus idle, bridge drives S_CBE[3:0] to a valid logic level when the internal grant is asserted.
S_PAR	B12	TS	**Secondary Parity:** S_PAR is an even parity of S_AD[31:0] and S_CBE[3:0] (i.e. an even number of 1's). S_PAR is valid and stable one cycle after the address phase (indicated by assertion of S_FRAME#) for address parity. For write data phases, S_PAR is valid one clock after S_IRDY# is asserted. For read data phase, S_PAR is valid one clock after S_TRDY# is asserted. Signal S_PAR is tri-stated one cycle after the S_AD lines are tri-stated. During bus idle, bridge drives S_PAR to a valid logic level when the internal grant is asserted.
S_FRAME#	B9	STS	**Secondary FRAME (Active LOW):** Driven by the initiator of a transaction to indicate the beginning and duration of an access. The de-assertion of S_FRAME# indicates the final data phase requested by the initiator. Before being tri-stated, it is driven to a de-asserted state for one cycle.
S_IRDY#	C9	STS	**Secondary IRDY (Active LOW):** Driven by the initiator of a transaction to indicate its ability to complete current data phase on the secondary side. Once asserted in a data phase, it is not de-asserted until the end of the data phase. Before tri-stated, it is driven to a de-asserted state for one cycle.

Name	Pin #	Type	Description
S_TRDY#	A10	STS	**Secondary TRDY (Active LOW):** Driven by the target of a transaction to indicate its ability to complete current data phase on the secondary side. Once asserted in a data phase, it is not de-asserted until the end of the data phase. Before tri-stated, it is driven to a de-asserted state for one cycle.
S_DEVSEL#	B10	STS	**Secondary Device Select (Active LOW):** Asserted by the target indicating that the device is accepting the transaction. As a master, bridge waits for the assertion of this signal within 5 cycles of S_FRAME# assertion; otherwise, terminate with master abort. Before tri-stated, it is driven to a de-asserted state for one cycle.
S_STOP#	C10	STS	**Secondary STOP (Active LOW):** Asserted by the target indicating that the target is requesting the initiator to stop the current transaction. Before tri-stated, it is driven to a de-asserted state for one cycle.
S_LOCK#	A11	STS	**Secondary LOCK (Active LOW):** Asserted by an initiator, one clock cycle after the first address phase of a transaction, when it is propagating a locked transaction downstream. Bridge does not propagate locked transactions upstream.
S_PERR#	C11	STS	**Secondary Parity Error (Active LOW):** Asserted when a data parity error is detected for data received on the secondary interface. Before being tri-stated, it is driven to a de-asserted state for one cycle.
S_SERR#	B11	I	**Secondary System Error (Active LOW):** Can be driven LOW by any device to indicate a system error condition.
S_REQ#[8:0]	E1, E3, D2, D1, E4, D3, C2, C1, D4	I	**Secondary Request (Active LOW):** This is asserted by an external device to indicate that it wants to start a transaction on the secondary bus. The input is externally pulled up through a resistor to VDD.
S_GNT#[8:0]	H1, G3, G2, G4, G1, F2, F1, F3, E2	TS	**Secondary Grant (Active LOW):** PI7C8154A asserts these pins to allow external masters to access the secondary bus. Bridge de-asserts these pins for at least 2 PCI clock cycles before asserting it again. During idle and S_GNT# deasserted, PI7C8154A will drive S_AD, S_CBE, and S_PAR.
S_RESET#	H2	O	**Secondary RESET (Active LOW):** Asserted when any of the following conditions are met: 1. Signal P_RESET# is asserted. 2. Secondary reset bit in bridge control register in configuration space is set. 3. The chip reset bit in the chip control register in configuration space is set. When asserted, all control signals are tri-stated and zeroes are driven on S_AD, S_CBE, S_PAR, and S_PAR64.
S_M66EN	A14	I/OD	**Secondary Interface 66MHz Operation:** This input is used to specify if bridge is capable of running at 66MHz on the secondary side. When HIGH, the Secondary bus may run at 66MHz. When LOW, the Secondary bus may only run at 33MHz. If P_M66EN is pulled LOW, the S_M66EN is driven LOW.
S_CFN#	K1	I	**Secondary Bus Central Function Control Pin:** When tied LOW, it enables the internal arbiter. When tied HIGH, an external arbiter must be used. S_REQ#[0] is reconfigured to be the secondary bus grant input, and S_GNT#[0] is reconfigured to be the secondary bus request output.

1.2.4 SECONDARY BUS INTERFACE SIGNALS – 64-EXTENSTION

Name	Pin #	Type	Description
S_AD[63:32]	C20, A21, D20, C21, C23, C22, D21, E20, D22, E21, E23, F21, F23, F22, G20, G22, G21, H23, H22, H21, J23, J20, J22, K23, K22, K21, L23, L21, L22, M22, M23, M21	TS	**Secondary Upper 32-bit Address/Data:** Multiplexed address and data bus. Address is indicated by S_FRAME# assertion. Write data is stable and valid when S_IRDY# is asserted and read data is stable and valid when S_IRDY# is asserted. Data is transferred on rising clock edges when both S_IRDY# and S_TRDY# are asserted. During bus idle, bridge drives S_AD to a valid logic level when S_GNT# is asserted respectively.
S_CBE[7:4]	A19, C19, A20, D19	TS	**Secondary Upper 32-bit Command/Byte Enables:** Multiplexed command field and byte enable field. During address phase, the initiator drives the transaction type on these pins. The initiator then drives the byte enables during data phases. During bus idle, bridge drives S_CBE[7:0] to a valid logic level when the internal grant is asserted.
S_PAR64	N21	TS	**Secondary Upper 32-bit Parity:** S_PAR64 carries the even parity of S_AD[63:32] and S_CBE[7:4] for both address and data phases. S_PAR64 is driven by the initiator and is valid 1 cycle after the first address phase when a dual address command is used and S_REQ64# is asserted. S_PAR64 is valid 1 clock cycle after the second address phase of a dual address transaction when S_REQ64# is asserted. S_PAR64 is valid 1 cycle after valid data is driven when both S_REQ64# and S_ACK64# are asserted for that data phase. S_PAR64 is driven by the device driving read or write data 1 cycle after the S_AD lines are driven. S_PAR64 is tri-stated 1 cycle after the S_AD lines are tri-stated. Devices receive data sample S_PAR64 as an input to check for possible parity errors during 64-bit transactions. When not driven, S_PAR64 is pulled up to a valid logic level through external resistors.
S_REQ64#	B19	STS	**Secondary 64-bit Transfer Request:** S_REQ64# is asserted by the initiator to indicate that the initiator is requesting a 64-bit data transfer. S_REQ64# has the same timing as S_FRAME#. When S_REQ64# is asserted LOW during reset, a 64-bit data path is supported. When S_REQ64# is HIGH during reset, bridge drives S_AD[63:32], S_CBE[7:4], and S_PAR64 to valid logic levels. When deasserting, S_REQ64# is driven to a deasserted state for 1 cycle and then sustained by an external pull-up resistor.
S_ACK64#	C18	STS	**Secondary 64-bit Transfer Acknowledge:** S_ACK64# is asserted by the target only when S_REQ64# is asserted by the initiator to indicate the target's ability to transfer data using 64 bits. S_ACK64# has the same timing as S_DEVSEL#. When deasserting, S_ACK64# is driven to a deasserted state for 1 cycle and then is sustained by an external pull-up resistor.

1.2.5 CLOCK SIGNALS

Name	Pin #	Type	Description
P_CLK	T3	I	**Primary Clock Input:** Provides timing for all transactions on the primary interface.

Name	Pin #	Type	Description
S_CLKIN	J4	I	**Secondary Clock Input:** Provides timing for all transactions on the secondary interface.
S_CLKOUT[9:0]	P1, P2, P3, N1, N3, M2, M1, M3, L3, L2	O	**Secondary Clock Output:** Provides secondary clocks phase synchronous with the P_CLK. When these clocks are used, one of the clock outputs must be fed back to S_CLKIN. Unused outputs may be disabled by: 1. Writing the secondary clock disable bits in the configuration space 2. Using the serial disable mask using the GPIO pins and MSK_IN 3. Terminating them electrically.

1.2.6 MISCELLANEOUS SIGNALS

Name	Pin #	Type	Description
MSK_IN	R21	I	**Secondary Clock Disable Serial Input:** This pin is used by bridge to disable secondary clock outputs. The serial stream is received by MSK_IN, starting when P_RESET is detected deasserted and S_RESET# is detected as being asserted. The serial data is used for selectively disabling secondary clock outputs and is shifted into the secondary clock control configuration register. This pin can be tied LOW to enable all secondary clock outputs or tied HIGH to drive all the secondary clock outputs HIGH.
P_VIO	R20	I	**Primary I/O Voltage:** This pin is used to determine either 3.3V or 5V signaling on the primary bus. P_VIO must be tied to 3.3V only when all devices on the primary bus use 3.3V signaling. Otherwise, P_VIO is tied to 5V.
S_VIO	N22	I	**Secondary I/O Voltage:** This pin is used to determine either 3.3V or 5V signaling on the secondary bus. S_VIO must be tied to 3.3V only when all devices on the secondary bus use 3.3V signaling. Otherwise, S_VIO is tied to 5V.
BPCCE	R4	I	**Bus/Power Clock Control Management Pin:** When this pin is tied HIGH and the bridge is placed in the D2 or $D3_{HOT}$ power state, it enables the bridge to place the secondary bus in the B2 power state. The secondary clocks are disabled and driven to 0. When this pin is tied LOW, there is no effect on the secondary bus clocks when the bridge enters the D2 or $D3_{HOT}$ power state.
CONFIG66	R22	I	**66MHz Configuration:** This pin indicates if the bridge is capable of running at 66MHz operation. Tie HIGH to set bit [21] of offset 04h of the status register.
PMEENA#	D11	I	**Power Management Enable Support:** This pin sets bits [31:27] offset DEh of the Power Management Capabilities Register. When tied LOW, bits [31:27] offset DEh are set to 11111 to indicate that the secondary devices are capable of asserting PME#. When this pin is tied HIGH, bits [31:27] offset DEh are set to 00000 to indicate that PI7C8154A does not support the PME# pin. For Intel 21154, this pin is defined as VDD. For more info, please refer to Appendix on page 113.

EEDATA	A22	I/O	**EEPROM Data:** Serial data interface to the EEPROM. For Intel 21154, this pin is defined as VDD. For more info, please refer to Appendix on page 113.
EECLK	A23	O	**EEPROM Clock:** Clock signal to the EEPROM interface used during the autoload and VPD functions. For Intel 21154, this pin is defined as VSS. For more info, please refer to Appendix on page 113.
EE_EN#	AC22	I	**EEPROM Enable:** Set to LOW to enable EEPROM interface. For Intel 21154, this pin is defined as VDD. For more info, please refer to Appendix on page 113.

1.2.7 GENERAL PURPOSE I/O INTERFACE SIGNALS

Name	Pin #	Type	Description
GPIO[3:0]	K2, K3, L4, L1	TS	**General Purpose I/O Data Pins:** The 4 general-purpose signals are programmable as either input-only or bi-directional signals by writing the GPIO output enable control register in the configuration space.

1.2.8 JTAG BOUNDARY SCAN SIGNALS

Name	Pin #	Type	Description
TCK	N20	I	**Test Clock.** Used to clock state information and data into and out of the bridge during boundary scan.
TMS	P21	I	**Test Mode Select.** Used to control the state of the Test Access Port controller.
TDO	P22	O	**Test Data Output.** Used as the serial output for the test instructions and data from the test logic.
TDI	P23	I	**Test Data Input.** Serial input for the JTAG instructions and test data.
TRST#	N23	I	**Test Reset.** Active LOW signal to reset the Test Access Port (TAP) controller into an initialized state.

1.2.9 POWER AND GROUND

Name	Pin #	Type	Description
VDD	A2, B1, B6, B20, B23, D5, D6, D10, D14, D15, D18, E22, H4, H20, J1, J3, J21, M4, M20, N4, R1, R23, T1, T4, T20, W3, Y6, Y10, Y14, Y18, Y22, AB1, AB19, AB23, AC2, AC3, AC8, AC12, AC16	P	**Power:** +3.3V Digital power.
VSS	A1, A5, A12, A16, B2, B21, B22, C7, D8, D12, D16, D23, F4, F20, G23, H3, J2, K4, K20, L20, N2, P4, P20, T2, U21, V4, V20, Y8, Y9, Y12, Y16, AA2, AA22, AB2, AB22, AC1, AC4, AC13, AC17, AC20, AC23	P	**Ground:** Digital ground.

1.3 PIN LIST

BALL LOCATION	PIN NAME	TYPE	BALL LOCATION	PIN NAME	TYPE
A1	VSS	P	A2	VDD	P
A3	S_AD[30]	TS	A4	S_AD[27]	TS
A5	VSS	P	A6	S_AD[23]	TS
A7	S_AD[22]	TS	A8	S_AD[19]	TS
A9	S_AD[16]	TS	A10	S_TRDY#	STS
A11	S_LOCK#	STS	A12	VSS	P
A13	S_AD13]	TS	A14	SM66EN	I/OD
A15	S_CBE[0]	TS	A16	VSS	P
A17	S_AD[2]	TS	A18	S_AD[0]	TS
A19	S_CBE[7]	TS	A20	S_CBE[5]	TS
A21	S_AD[62]	TS	A22	EEDATA	I/O
A23	EECLK	O	-	-	-
B1	VDD	P	B2	VSS	P
B3	S_AD[29]	TS	B4	S_AD[26]	TS
B5	S_AD[24]	TS	B6	VDD	P
B7	S_AD[20]	TS	B8	S_AD[18]	TS
B9	S_FRAME#	STS	B10	S_DEVSEL#	STS
B11	S_SERR#	I	B12	S_PAR	TS
B13	S_AD[14]	TS	B14	S_AD[10]	TS
B15	S_AD[8]	TS	B16	S_AD[6]	TS
B17	S_AD[4]	TS	B18	S_AD[1]	TS
B19	S_REQ64#	STS	B20	VDD	P
B21	VSS	P	B22	VSS	P
B23	VDD	P	-	-	-
C1	S_REQ#[1]	I	C2	S_REQ#[2]	I
C3	S_AD[31]	TS	C4	S_AD[28]	TS
C5	S_AD[25]	TS	C6	S_CBE[3]	TS
C7	VSS	P	C8	S_AD[17]	TS
C9	S_IRDY#	STS	C10	S_STOP#	STS
C11	S_PERR#	STS	C12	S_CBE[1]	TS
C13	S_AD[15]	TS	C14	S_AD[11]	TS
C15	S_AD[9]	TS	C16	S_AD[7]	TS
C17	S_AD[5]	TS	C18	S_ACK64#	STS
C19	S_CBE[6]	TS	C20	S_AD[63]	TS
C21	S_AD[60]	TS	C22	S_AD[58]	TS
C23	S_AD[59]	TS	-	-	-
D1	S_REQ#[5]	I	D2	S_REQ#[6]	I
D3	S_REQ_[3]	I	D4	S_REQ#[0]	I
D5	VDD	P	D6	VDD	P
D7	S_AD[21]	TS	D8	VSS	P
D9	S_CBE[2]	TS	D10	VDD	P
D11	PMEENA#	I	D12	VSS	P
D13	S_AD[12]	TS	D14	VDD	P
D15	VDD	P	D16	VSS	P
D17	S_AD[3]	TS	D18	VDD	P
D19	S_CBE[4]	TS	D20	S_AD[61]	TS
D21	S_AD[57]	TS	D22	S_AD[55]	TS
D23	VSS	P	-	-	-
E1	S_REQ#[8]	I	E2	S_GNT#[0]	TS
E3	S_REQ#[7]	I	E4	S_REQ#[4]	I
-	-	-	E20	S_AD[56]	TS
E21	S_AD[54]	TS	E22	VDD	P
E23	S_AD[53]	TS	-	-	-
F1	S_GNT#[2]	TS	F2	S_GNT#[3]	TS
F3	S_GNT#[1]	TS	F4	VSS	P
-	-	-	F20	VSS	P

BALL LOCATION	PIN NAME	TYPE	BALL LOCATION	PIN NAME	TYPE
F21	S_AD[52]	TS	F22	S_AD[50]	TS
F23	S_AD[51]	TS	-	-	-
G1	S_GNT#[4]	TS	G2	S_GNT#[6]	TS
G3	S_GNT#[7]	TS	G4	S_GNT#[5]	TS
-	-	-	G20	S_AD[49]	TS
G21	S_AD[47]	TS	G22	S_AD[48]	TS
G23	VSS	P	-	-	-
H1	S_GNT#[8]	TS	H2	S_RESET#	O
H3	VSS	P	H4	VDD	P
-	-	-	H20	VDD	P
H21	S_AD[44]	TS	H22	S_AD[45]	TS
H23	S_AD[46]	TS	-	-	-
J1	VDD	P	J2	VSS	P
J3	VDD	P	J4	S_CLKIN	I
-	-	-	J20	S_AD[42]	TS
J21	VDD	P	J22	S_AD[41]	TS
J23	S_AD[43]	TS	-	-	-
K1	S_CFN#	I	K2	GPIO[3]	TS
K3	GPIO[2]	TS	K4	VSS	P
-	-	-	K20	VSS	P
K21	S_AD[38]	TS	K22	S_AD[39]	TS
K23	S_AD[40]	TS	-	-	-
L1	GPIO[0]	TS	L2	S_CLKOUT[0]	O
L3	S_CLKOUT[1]	O	L4	GPIO[1]	TS
-	-	-	L20	VSS	P
L21	S_AD[36]	TS	L22	S_AD[35]	TS
L23	S_AD[37]	TS	-	-	-
M1	S_CLKOUT[3]	O	M2	S_CLKOUT[4]	O
M3	S_CLKOUT[2]	O	M4	VDD	P
-	-	-	M20	VDD	P
M21	S_AD[32]	TS	M22	S_AD[34]	TS
M23	S_AD[33]	TS	-	-	-
N1	S_CLKOUT[6]	O	N2	VSS	P
N3	S_CLKOUT[5]	O	N4	VDD	P
-	-	-	N20	TCK	I
N21	S_PAR64	TS	N22	S_VIO	I
N23	TRST#	I	-	-	-
P1	S_CLKOUT[9]	O	P2	S_CLKOUT[8]	O
P3	S_CLKOUT[7]	O	P4	VSS	P
-	-	-	P20	VSS	P
P21	TMS	I	P22	TDO	O
P23	TDI	I	-	-	-
R1	VDD	P	R2	P_GNT#	I
R3	P_RESET#	I	R4	BPCCE	I
-	-	-	R20	P_VIO	I
R21	MSK_IN	I	R22	CONFIG66	I
R23	VDD	P	-	-	-
T1	VDD	P	T2	VSS	P
T3	P_CLK	I	T4	VDD	P
-	-	-	T20	VDD	P
T21	P_PAR64	TS	T22	P_AD[32]	TS
T23	P_AD[33]	TS	-	-	-
U1	P_AD[29]	TS	U2	P_AD[31]	TS
U3	P_REQ#	TS	U4	P_AD[30]	TS
-	-	-	U20	P_AD[35]	TS
U21	VSS	P	U22	P_AD[34]	TS
U23	P_AD[36]	TS	-	-	-
V1	P_AD[27]	TS	V2	P_AD[28]	TS

BALL LOCATION	PIN NAME	TYPE	BALL LOCATION	PIN NAME	TYPE
V3	P_AD[26]	TS	V4	VSS	P
-	-	-	V20	Reserved[1]	P
V21	P_AD[39]	TS	V22	P_AD[37]	TS
V23	P_AD[38]	TS	-	-	-
W1	P_AD[24]	TS	W2	P_AD[25]	TS
W3	VDD	P	W4	P_AD[23]	TS
-	-	-	W20	P_AD[44]	TS
W21	P_AD[42]	TS	W22	P_AD[40]	TS
W23	P_AD[41]	TS	-	-	-
Y1	P_IDSEL	I	Y2	P_CBE[3]	TS
Y3	P_AD[22]	TS	Y4	P_AD[19]	TS
Y5	P_AD[16]	TS	Y6	VDD	P
Y7	P_SERR#	OD	Y8	VSS	P
Y9	VSS	P	Y10	VDD	P
Y11	P_AD[8]	TS	Y12	VSS	P
Y13	P_AD[1]	TS	Y14	VDD	P
Y15	P_CBE[5]	TS	Y16	VSS	P
Y17	P_AD[59]	TS	Y18	Reserved[2]	P
Y19	P_AD[52]	TS	Y20	P_AD[47]	TS
Y21	P_AD[45]	TS	Y22	VDD	P
Y23	P_AD[43]	TS	-	-	-
AA1	P_AD[21]	TS	AA2	VSS	P
AA3	P_AD[20]	TS	AA4	P_AD[17]	TS
AA5	P_FRAME#	STS	AA6	P_DEVSEL#	STS
AA7	P_CBE[1]	TS	AA8	P_AD[14]	TS
AA9	P_AD[11]	TS	AA10	P_AD[9]	TS
AA11	P_AD[6]	TS	AA12	P_AD[5]	TS
AA13	P_AD[2]	TS	AA14	P_AD[0]	TS
AA15	P_CBE[7]	TS	AA16	P_AD[63]	TS
AA17	P_AD[61]	TS	AA18	P_AD[56]	TS
AA19	P_AD[54]	TS	AA20	P_AD[51]	TS
AA21	P_AD[48]	TS	AA22	VSS	P
AA23	P_AD[46]	TS	-	-	-
AB1	VDD	P	AB2	VSS	P
AB3	P_AD[18]	TS	AB4	P_CBE[2]	TS
AB5	P_TRDY#	STS	AB6	P_LOCK#	I
AB7	P_PAR	TS	AB8	P_AD[15]	TS
AB9	P_AD[12]	TS	AB10	P_M66EN	I
AB11	P_AD[7]	TS	AB12	P_AD[4]	TS
AB13	P_AD[3]	TS	AB14	P_ACK64#	STS
AB15	P_CBE[6]	TS	AB16	P_AD[62]	TS
AB17	P_AD[60]	TS	AB18	P_AD[58]	TS
AB19	VDD	P	AB20	P_AD[53]	TS
AB21	P_AD[50]	TS	AB22	VSS	P
AB23	VDD	P	-	-	-
AC1	VSS	P	AC2	VDD	P
AC3	VDD	P	AC4	VSS	P
AC5	P_IRDY#	STS	AC6	P_STOP#	STS
AC7	P_PERR#	STS	AC8	VDD	P
AC9	P_AD[13]	TS	AC10	P_AD[10]	TS
AC11	P_CBE[0]	TS	AC12	VDD	P
AC13	VSS	P	AC14	P_REQ64#	STS
AC15	P_CBE[4]	TS	AC16	VDD	P
AC17	VSS	P	AC18	P_AD[57]	TS
AC19	P_AD[55]	TS	AC20	VSS	P
AC21	P_AD[49]	TS	AC22	EE_EN#	I
AC23	VSS	P	-	-	-

[1] Connected to GROUND
[2] Connected to V_{DD}

2 SIGNAL DEFINITIONS

This Chapter offers information about PCI transactions, transaction forwarding across PI7C8154A, and transaction termination. The PI7C8154A has two 128-byte buffers for read data buffering of upstream and downstream transactions. Also, PI7C8154A has two 128-byte buffers for write data buffering of upstream and downstream transactions.

2.1 TYPES OF TRANSACTIONS

This section provides a summary of PCI transactions performed by PI7C8154A. Table 2-1 lists the command code and name of each PCI transaction. The Master and Target columns indicate support for each transaction when PI7C8154A initiates transactions as a master, on the primary and secondary buses, and when PI7C8154A responds to transactions as a target, on the primary and secondary buses.

Table 2-1 PCI TRANSACTIONS

Types of Transactions		Initiates as Master		Responds as Target	
		Primary	Secondary	Primary	Secondary
0000	Interrupt Acknowledge	N	N	N	N
0001	Special Cycle	Y	Y	N	N
0010	I/O Read	Y	Y	Y	Y
0011	I/O Write	Y	Y	Y	Y
0100	Reserved	N	N	N	N
0101	Reserved	N	N	N	N
0110	Memory Read	Y	Y	Y	Y
0111	Memory Write	Y	Y	Y	Y
1000	Reserved	N	N	N	N
1001	Reserved	N	N	N	N
1010	Configuration Read	N	Y	Y	N
1011	Configuration Write	Y (Type 1 only)	Y	Y	Y (Type 1 only)
1100	Memory Read Multiple	Y	Y	Y	Y
1101	Dual Address Cycle	Y	Y	Y	Y
1110	Memory Read Line	Y	Y	Y	Y
1111	Memory Write and Invalidate	Y	Y	Y	Y

As indicated in Table 2-1, the following PCI commands are not supported by PI7C8154A:

- PI7C8154A never initiates a PCI transaction with a reserved command code and, as a target, PI7C8154A ignores reserved command codes.
- PI7C8154A does not generate interrupt acknowledge transactions. PI7C8154A ignores interrupt acknowledge transactions as a target.
- PI7C8154A does not respond to special cycle transactions. PI7C8154A 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, Type 1 configuration write must be used.
- PI7C8154A neither generates Type 0 configuration transactions on the primary PCI bus nor responds to Type 0 configuration transactions on the secondary PCI bus.

2.2 SINGLE ADDRESS PHASE

A 32-bit address uses a single address phase. This address is driven on P_AD[31:0], and the bus command is driven on P_CBE[3:0]. PI7C8154A supports the linear increment address mode only, which is indicated when the lowest two address bits are equal to zero. If either of the lowest two address bits is nonzero, PI7C8154A automatically disconnects the transaction after the first data transfer.

2.3 DUAL ADDRESS PHASE

A 64-bit address uses two address phases. The first address phase is denoted by the asserting edge of FRAME#. The second address phase always follows on the next clock cycle.

For a 32-bit interface, the first address phase contains dual address command code on the CBE[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 CBE[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. See Section 3.3.3 for a discussion of prefetchable address space. The PI7C8154A supports dual address transactions in both the upstream and the downstream direction. The PI7C8154A supports a programmable 64-bit address range in prefetchable memory for downstream forwarding of dual address transactions. Dual address transactions falling outside the prefetchable address range are forwarded upstream, but not downstream. Prefetching and posting are performed in a manner consistent with the guidelines given in this document for each type of memory transaction in prefetchable memory space.

2.4 DEVICE SELECT (DEVSEL#) GENERATION

PI7C8154A always performs positive address decoding (medium decode) when accepting transactions on either the primary or secondary buses. PI7C8154A never does subtractive decode.

2.5 DATA PHASE

The address phase of a PCI transaction is 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 during the same PCI clock cycle. The last data phase of a transaction is indicated when FRAME# is de-asserted and both TRDY# and IRDY# are asserted, or when IRDY# and STOP# are asserted. See Section 2.11 for further discussion of transaction termination.

Depending on the command type, PI7C8154A can support multiple data phase PCI transactions. For detailed descriptions of how PI7C8154A imposes disconnect boundaries, see Section 2.6.4 for write address boundaries and Section 2.7.3 read address boundaries.

2.6 WRITE TRANSACTIONS

Write transactions are treated as either posted write or delayed write transactions. Table 2-2 shows the method of forwarding used for each type of write operation.

Table 2-2 WRITE TRANSACTION FORWARDING

Type of Transaction	Type of Forwarding
Memory Write	Posted (except VGA memory)
Memory Write and Invalidate	Posted
Memory Write to VGA memory	Delayed
I/O Write	Delayed
Type 1 Configuration Write	Delayed

2.6.1 MEMORY WRITE TRANSACTIONS

Posted write forwarding is used for "Memory Write" and "Memory Write and Invalidate" transactions.

When PI7C8154A determines that a memory write transaction is to be forwarded across the bridge, PI7C8154A asserts DEVSEL# with medium decode timing and TRDY# in the next cycle, provided that enough buffer space is available in the posted memory write queue for the address and at least one DWORD of data. Under this condition, PI7C8154A accepts write data without obtaining access to the target bus. The PI7C8154A can accept one DWORD of write data every PCI clock cycle. That is, no target wait state is inserted. The write data is stored in an internal posted write buffers and is subsequently delivered to the target. The PI7C8154A continues to accept write data until one of the following events occurs:

- The initiator terminates the transaction by de-asserting FRAME# and IRDY#.
- An internal write address boundary is reached, such as a cache line boundary or an aligned 4KB boundary, depending on the transaction type.
- The posted write data buffer fills up.

When one of the last two events occurs, the PI7C8154A returns a target disconnect to the requesting initiator on this data phase to terminate the transaction.

Once the posted write data moves to the head of the posted data queue, PI7C8154A asserts its request on the target bus. This can occur while PI7C8154A is still receiving data on the initiator bus. When the grant for the target bus is received and the target bus is detected in the idle condition, PI7C8154A asserts FRAME# and drives the stored write address out on the target bus. On the following cycle, PI7C8154A drives the first DWORD of write data and continues to transfer write data until all write data corresponding to that transaction is delivered, or until a target termination is received. As long as write data exists in the queue, PI7C8154A can drive one DWORD of write data in each PCI clock cycle; that is, no master wait states are inserted. If write data is flowing through PI7C8154A and the initiator stalls, PI7C8154A will signal the last data phase for the current transaction at the target bus if the queue empties. PI7C8154A will restart the follow-on transactions if the queue has new data.

PI7C8154A ends the transaction on the target bus when one of the following conditions is met:

- All posted write data has been delivered to the target.
- The target returns a target disconnect or target retry (PI7C8154A starts another transaction to deliver the rest of the write data).

- The target returns a target abort (PI7C8154A discards remaining write data).
- The master latency timer expires, and PI7C8154A no longer has the target bus grant (PI7C8154A starts another transaction to deliver remaining write data).

Section 2.11.3.2 provides detailed information about how PI7C8154A responds to target termination during posted write transactions.

2.6.2 MEMORY WRITE AND INVALIDATE

Posted write forwarding is used for Memory Write and Invalidate transactions.

The PI7C8154A disconnects Memory Write and Invalidate commands at aligned cache line boundaries. The cache line size value in the cache line size register gives the number of DWORD in a cache line.

If the value in the cache line size register does meet the memory write and invalidate conditions, the PI7C8154A returns a target disconnect to the initiator on a cache line boundary.

2.6.3 DELAYED WRITE TRANSACTIONS

Delayed write forwarding is used for I/O write transactions and Type 1 configuration write transactions.

A delayed write transaction guarantees that the actual target response is returned back to the initiator without holding the initiating bus in wait states. A delayed write transaction is limited to a single DWORD data transfer.

When a write transaction is first detected on the initiator bus, and PI7C8154A forwards it as a delayed transaction, PI7C8154A claims the access by asserting DEVSEL# and returns a target retry to the initiator. During the address phase, PI7C8154A samples the bus command, address, and address parity one cycle later. After IRDY# is asserted, PI7C8154A also samples the first data DWORD, byte enable bits, and data parity. This information is placed into the delayed transaction queue. The transaction is queued only if no other existing delayed transactions have the same address and command, and if the delayed transaction queue is not full. When the delayed write transaction moves to the head of the delayed transaction queue and all ordering constraints with posted data are satisfied. The PI7C8154A initiates the transaction on the target bus. PI7C8154A transfers the write data to the target. If PI7C8154A receives a target retry in response to the write transaction on the target bus, it continues to repeat the write transaction until the data transfer is completed, or until an error condition is encountered.

If PI7C8154A is unable to deliver write data after 2^{24} (default) or 2^{32} (maximum) attempts, PI7C8154A will report a system error. PI7C8154A also asserts P_SERR# if the primary SERR# enable bit is set in the command register. See Section 5.4 for information on the assertion of P_SERR#. When the initiator repeats the same write transaction (same command, address, byte enable bits, and data), and the completed delayed transaction is at the head of the queue, the PI7C8154A claims the access by asserting DEVSEL# and returns TRDY# to the initiator, to indicate that the write data was transferred. If the initiator requests multiple DWORD, PI7C8154A also asserts STOP# in conjunction with TRDY# to signal a target disconnect. Note that only those bytes of write data with valid byte enable bits are compared. If any of the byte enable bits are turned off (driven HIGH), the corresponding byte of write data is not compared.

If the initiator repeats the write transaction before the data has been transferred to the target, PI7C8154A returns a target retry to the initiator. PI7C8154A continues to return a target retry to the initiator until write data is delivered to the target, or until an error condition is encountered. When the write transaction is repeated, PI7C8154A does not make a new entry into the delayed transaction queue. Section 2.11.3.1 provides detailed information about how PI7C8154A responds to target termination during delayed write transactions.

PI7C8154A implements a discard timer that starts counting when the delayed write completion is at the head of the delayed transaction completion queue. The initial value of this timer can be set to the retry counter register offset 78h.

If the initiator does not repeat the delayed write transaction before the discard timer expires, PI7C8154A discards the delayed write completion from the delayed transaction completion queue. PI7C8154A also conditionally asserts P_SERR# (see Section 5.4).

2.6.4 WRITE TRANSACTION ADDRESS BOUNDARIES

PI7C8154A imposes internal address boundaries when accepting write data. The aligned address boundaries are used to prevent PI7C8154A from continuing a transaction over a device address boundary and to provide an upper limit on maximum latency. PI7C78154 returns a target disconnect to the initiator when it reaches the aligned address boundaries under conditions shown in Table 2-3.

Table 2-3 WRITE TRANSACTION DISCONNECT ADDRESS BOUNDARIES

Type of Transaction	Condition	Aligned Address Boundary
Delayed Write	All	Disconnects after one data transfer
Posted Memory Write	Memory write disconnect control bit = 0(1)	4KB aligned address boundary
Posted Memory Write	Memory write disconnect control bit = 1(1)	Disconnects at cache line boundary
Posted Memory Write and Invalidate	Cache line size ≠ 1, 2, 4, 8, 16	4KB aligned address boundary
Posted Memory Write and Invalidate	Cache line size = 1, 2, 4, 8	Cache line boundary if posted memory write data FIFO does not have enough space for the next cache line
Posted Memory Write and Invalidate	Cache line size = 16	16-DWORD aligned address boundary

Note 1. Memory write disconnect control bit is bit 1 of the chip control register at offset 40h in the configuration space.

2.6.5 BUFFERING MULTIPLE WRITE TRANSACTIONS

PI7C8154A continues to accept posted memory write transactions as long as space for at least one DWORD of data in the posted write data buffer remains. If the posted write data buffer fills before the initiator terminates the write transaction, PI7C8154A returns a target disconnect to the initiator.

Delayed write transactions are accepted as long as at least one open entry in the delayed transaction queue exists. Therefore, several posted and delayed write transactions can exist in data buffers at the same time. See Chapter 4 for information about how multiple posted and delayed write transactions are ordered.

2.6.6 FAST BACK-TO-BACK TRANSACTIONS

PI7C8154A is capable of decoding and forwarding fast back-to-back write transactions. When PI7C8154A cannot accept the second transaction because of buffer space limitations, it returns a target retry to the initiator. The fast back-to-back enable bit must be set in the command register for upstream write transactions, and in the bridge control register for downstream write transactions.

2.7 READ TRANSACTIONS

Delayed read forwarding is used for all read transactions crossing PI7C8154A. Delayed read transactions are treated as either prefetchable or non-prefetchable. Table 2-5 shows the read behavior, prefetchable or non-prefetchable, for each type of read operation.

2.7.1 PREFETCHABLE READ TRANSACTIONS

A prefetchable read transaction is a read transaction where PI7C8154A performs speculative DWORD reads, transferring data from the target before it is requested from the initiator. This behavior allows a prefetchable read transaction to consist of multiple data transfers. However, byte enable bits cannot be forwarded for all data phases as is done for the single data phase of the non-prefetchable read transaction. For prefetchable read transactions, PI7C8154A forces all byte enable bits to be on for all data phases.

Prefetchable behavior is used for memory read line and memory read multiple transactions, as well as for memory read transactions that fall into prefetchable memory space.

The amount of data that is prefetched depends on the type of transaction. The amount of prefetching may also be affected by the amount of free buffer space available in PI7C8154A, and by any read address boundaries encountered.

Prefetching should not be used for those read transactions that have side effects in the target device, that is, control and status registers, FIFO's, and so on. The target device's base address register or registers indicate if a memory address region is prefetchable.

2.7.2 NON-PREFETCHABLE READ TRANSACTIONS

A non-prefetchable read transaction is a read transaction where PI7C8154A requests one and only one DWORD from the target and disconnects the initiator after delivery of the first DWORD of read data. Unlike prefetchable read transactions, PI7C8154A forwards the read byte enable information for the data phase.

Non-prefetchable behavior is used for I/O and configuration read transactions, as well as for memory read transactions that fall into non-prefetchable memory space.

If extra read transactions could have side effects, for example, when accessing a FIFO, use non-prefetchable read transactions to those locations. Accordingly, if it is important to retain the value of the byte enable bits during the data phase, use non-prefetchable read transactions. If these locations are mapped in memory space, use the memory read command and map the target into non-prefetchable (memory-mapped I/O) memory space to use non-prefetching behavior.

2.7.3 READ PREFETCH ADDRESS BOUNDARIES

PI7C8154A imposes internal read address boundaries on read prefetched data. When a read transaction reaches one of these aligned address boundaries, the PI7C8154A stops pre-fetched data, unless the target signals a target disconnect before the read prefetched boundary is reached. When PI7C8154A finishes transferring this read data to the initiator, it returns a target disconnect with the last data transfer, unless the initiator completes the transaction before all pre-fetched read data is delivered. Any leftover pre-fetched data is discarded.

Prefetchable read transactions in flow-through mode pre-fetch to the nearest aligned 4KB address boundary, or until the initiator de-asserts FRAME#. Section 2.7.6 describes flow-through mode during read operations.

Table 2-4 shows the read pre-fetch address boundaries for read transactions during non-flow-through mode.

Table 2-4 READ PREFETCH ADDRESS BOUNDARIES

Type of Transaction	Address Space	Cache Line Size (CLS)	Prefetch Aligned Address Boundary
Configuration Read	-	*	One DWORD (no prefetch)
I/O Read	-	*	One DWORD (no prefetch)
Memory Read	Non-Prefetchable	*	One DWORD (no prefetch)
Memory Read	Prefetchable	CLS = 0 or 16	16-DWORD aligned address boundary
Memory Read	Prefetchable	CLS = 1, 2, 4, 8	Cache line address boundary
Memory Read Line	-	CLS = 0 or 16	16-DWORD aligned address boundary
Memory Read Line	-	CLS = 1, 2, 4, 8	Cache line boundary
Memory Read Multiple	-	CLS = 0 or 16	Queue full
Memory Read Multiple	-	CLS = 1, 2, 4, 8	Second cache line boundary

- does not matter if it is prefetchable or non-prefetchable
* don't care

Table 2-5 READ TRANSACTION PREFETCHING

Type of Transaction	Read Behavior
I/O Read	Prefetching never allowed
Configuration Read	Prefetching never allowed
Memory Read	Downstream: Prefetching used if address is prefetchable space
	Upstream: Prefetching used or programmable
Memory Read Line	Prefetching always used
Memory Read Multiple	Prefetching always used

See Section 3.3 for detailed information about prefetchable and non-prefetchable address spaces.

2.7.4 DELAYED READ REQUESTS

PI7C8154A 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.

PI7C8154A accepts a delayed read request, by sampling the read address, read bus command, and address parity. When IRDY# is asserted, PI7C8154A then samples the byte enable bits for the first data phase. This information is entered into the delayed transaction queue. PI7C8154A 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) other than a target retry is received.

2.7.5 DELAYED READ COMPLETION ON TARGET BUS

When delayed read request reaches the head of the delayed transaction queue, PI7C8154A arbitrates for the target bus and initiates the read transaction only if all previously queued posted write transactions have been delivered. PI7C8154A uses the exact read address and read command captured from the initiator during the initial delayed read request to initiate the read transaction. If the read transaction is a non-prefetchable read, PI7C8154A drives the captured byte enable bits during the next cycle. If the transaction is a prefetchable read transaction, it drives all byte enable bits to zero for all data phases. If PI7C8154A receives a target retry in response to the read transaction on the target bus, it continues to repeat the read transaction until at least one data transfer is completed, or until an error condition is encountered. If the transaction is terminated via normal master termination or target disconnect after at least one data transfer has been completed, PI7C8154A does not initiate any further attempts to read more data.

If PI7C8154A is unable to obtain read data from the target after 2^{24} (default) or 2^{32} (maximum) attempts, PI7C8154A will report system error. The number of attempts is programmable. PI7C8154A also asserts P_SERR# if the primary SERR# enable bit is set in the command register. See Section 5.4 for information on the assertion of P_SERR#.

Once PI7C8154A receives DEVSEL# and TRDY# from the target, it transfers the data read to the opposite direction read data queue, pointing toward the opposite inter-face, before terminating the transaction. For example, read data in response to a downstream read transaction initiated on the primary bus is placed in the upstream read data queue. The PI7C8154A can accept one DWORD of read data each PCI clock cycle; that is, no master wait states are inserted. The number of DWORD's transferred during a delayed read transaction matches the prefetch address boundary given in Table 2-4 (assuming no disconnect is received from the target).

2.7.6 DELAYED READ COMPLETION ON INITIATOR BUS

When the transaction has been completed on the target bus, and the delayed read data is at the head of the read data queue, and all ordering constraints with posted write transactions have been satisfied, the PI7C8154A transfers the data to the initiator when the initiator repeats the transaction. For memory read transactions, PI7C8154A aliases memory read line and memory read multiple bus commands to memory read when matching the bus command of the transaction to the bus command in the delayed transaction queue if bit[3] of offset 74h is set to '1'. PI7C8154A returns a target disconnect along with the transfer of the last DWORD of read data to the initiator. If PI7C8154A initiator terminates the transaction before all read data has been transferred, the remaining read data left in data buffers is discarded.

When the master repeats the transaction and starts transferring prefetchable read data from data buffers while the read transaction on the target bus is still in progress and before a read boundary is reached on the target bus, the read transaction starts operating in flow-through mode. Because data is flowing through the data buffers from the target to the initiator, long read bursts can then be sustained. In this case, the read transaction is allowed to continue until the initiator terminates the transaction, or until an aligned 4KB address boundary is reached, or until the buffer fills, whichever comes first. When the buffer empties, PI7C8154A reflects the stalled condition to the initiator by disconnecting the initiator with data. The initiator may retry the transaction later if data are needed. If the initiator does not need any more data, the initiator will not continue the disconnected

transaction. In this case, PI7C8154A will start the master timeout timer. The remaining read data will be discarded after the master timeout timer expires. To provide better latency, if there are any other pending data for other transactions in the RDB (Read Data Buffer), the remaining read data will be discarded even though the master timeout timer has not expired.

PI7C8154A implements a master timeout timer that starts counting when the delayed read completion is at the head of the delayed transaction queue, and the read data is at the head of the read data queue. The initial value of this timer is programmable through configuration transaction. If the initiator does not repeat the read transaction and before the master timeout timer expires (2^{15} default), PI7C8154A discards the read transaction and read data from its queues. PI7C8154A also conditionally asserts P_SERR# (see Section 5.4).

PI7C8154A has the capability to post multiple delayed read requests, up to a maximum of four in each direction. If an initiator starts a read transaction that matches the address and read command of a read transaction that is already queued, the current read command is not posted as it is already contained in the delayed transaction queue.

See Section 4 for a discussion of how delayed read transactions are ordered when crossing PI7C8154A.

2.7.7 FAST BACK-TO-BACK TRANSACTIONS

PI7C8154A is capable of decoding fast back-to-back read transactions on both the primary and secondary. Also, PI7C8154A cannot generate fast back-to-back read transactions on the secondary or primary even though bit[23] of offset 3Ch is set to '1' or bit[9] of offset 04h is set to '1'.

2.8 CONFIGURATION TRANSACTIONS

Configuration transactions are used to initialize a PCI system. Every PCI device has a configuration space that is accessed by configuration commands. All registers are accessible in configuration space only.

In addition to accepting configuration transactions for initialization of its own configuration space, the PI7C8154A 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 two bits of the address set to 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 two address bits set to 01b.

The register number is found in both Type 0 and Type 1 formats and gives the DWORD address of the configuration register 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 addresses of Type 1 configuration

transaction 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.

2.8.1 TYPE 0 ACCESS TO PI7C8154A

The configuration space is accessed by a Type 0 configuration transaction on the primary interface. The configuration space cannot be accessed from the secondary bus. The PI7C8154A responds to a Type 0 configuration transaction by asserting P_DEVSEL# when the following conditions are met during the address phase:

- The bus command is a configuration read or configuration write transaction.
- Lowest two address bits P_AD[1:0] must be 00b.
- Signal P_IDSEL must be asserted.

PI7C8154A limits all configuration access to a single DWORD data transfer and returns target-disconnect with the first data transfer if additional data phases are requested. Because read transactions to configuration space do not have side effects, all bytes in the requested DWORD are returned, regardless of the value of the byte enable bits.

Type 0 configuration write and read transactions do not use data buffers; that is, these transactions are completed immediately, regardless of the state of the data buffers. The PI7C8154A ignores all Type 0 transactions initiated on the secondary interface.

2.8.2 TYPE 1 TO TYPE 0 CONFIGURATION

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 other than the one where the Type 1 transaction is generated.

PI7C8154A performs a Type 1 to Type 0 translation when the Type 1 transaction is generated on the primary bus and is intended for a device attached directly to the secondary bus. PI7C8154A 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; that is, PI7C8154A generates a Type 0 transaction only on the secondary bus, and never on the primary bus.

PI7C8154A responds to a Type 1 configuration transaction and translates it into a Type 0 transaction on the secondary bus when the following conditions are met during the address phase:

- The lowest two address bits on P_AD[1:0] are 01b.
- The bus number in address field P_AD[23:16] is equal to the value in the secondary bus number register in configuration space.
- The bus command on P_CBE[3:0] is a configuration read or configuration write transaction.

When PI7C8154A translates the Type 1 transaction to a Type 0 transaction on the secondary interface, it performs the following translations to the address:

- Sets the lowest two address bits on S_AD[1:0] to 0.

- Decodes the device number and drives the bit pattern specified in Table 2-6 on S_AD[31:16] for the purpose of asserting the device's IDSEL signal.
- Sets S_AD[15:11] to 0.
- Leaves unchanged the function number and register number fields.

PI7C8154A asserts a unique address line based on the device number. These address lines may be used as secondary bus IDSEL signals. The mapping of the address lines depends on the device number in the Type 1 address bits P_AD[15:11]. Table 2-6 presents the mapping that PI7C8154A uses.

Table 2-6 DEVICE NUMBER TO IDSEL S_AD PIN MAPPING

Device Number	P_AD[15:11]	Secondary IDSEL S_AD[31:16]	S_AD
0h	00000	0000 0000 0000 0001	16
1h	00001	0000 0000 0000 0010	17
2h	00010	0000 0000 0000 0100	18
3h	00011	0000 0000 0000 1000	19
4h	00100	0000 0000 0001 0000	20
5h	00101	0000 0000 0010 0000	21
6h	00110	0000 0000 0100 0000	22
7h	00111	0000 0000 1000 0000	23
8h	01000	0000 0001 0000 0000	24
9h	01001	0000 0010 0000 0000	25
Ah	01010	0000 0100 0000 0000	26
Bh	01011	0000 1000 0000 0000	27
Ch	01100	0001 0000 0000 0000	28
Dh	01101	0010 0000 0000 0000	29
Eh	01110	0100 0000 0000 0000	30
Fh	01111	1000 0000 0000 0000	31
10h – 1Eh	10000 – 11110	0000 0000 0000 0000	-
1Fh	11111	Generate special cycle (P_AD[7:2] = 00h) 0000 0000 0000 0000 (P_AD[7:2] = 00h)	-

PI7C8154A 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 8. Because of electrical loading constraints of the PCI bus, more than 16 IDSEL signals should not be necessary. However, if device numbers greater than 16 are desired, some external method of generating IDSEL lines must be used, and no upper address bits are then asserted. The configuration transaction is still translated and passed from the primary bus to the secondary bus. If no IDSEL pin is asserted to a secondary device, the transaction ends in a master abort.

PI7C8154A forwards Type 1 to Type 0 configuration read or write transactions as delayed transactions. Type 1 to Type 0 configuration read or write transactions are limited to a single 32-bit data transfer.

2.8.3 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 PI7C8154A detects a Type 1 configuration transaction intended for a PCI bus downstream from the secondary bus, PI7C8154A 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 lowest two 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.

PI7C8154A also supports Type 1 to Type 1 forwarding of configuration write transactions upstream to support upstream special cycle generation. A Type 1 configuration command is forwarded upstream when the following conditions are met:

- The lowest two address bits are equal to 01b.
- The bus number falls outside the range defined by the lower limit (inclusive) in the secondary bus number register and the upper limit (inclusive) in the subordinate bus number register.
- 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 bus command is a configuration write transaction.

The PI7C8154A forwards Type 1 to Type 1 configuration write transactions as delayed transactions. Types 1 to Type 1 configuration write transactions are limited to a single data transfer.

2.8.4 SPECIAL CYCLES

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

PI7C8154A initiates a special cycle on the target bus when a Type 1 configuration write transaction is being detected on the initiating bus and the following conditions are met during the address phase:

- The lowest two 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 on CBE is a configuration write command.

When PI7C8154A initiates the transaction on the target interface, the bus command is changed from configuration write to special cycle. The address and data are for-warded 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 delayed 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, PI7C8154A responds with TRDY# to the next attempt of the con-figuration transaction from the initiator. If more than one data transfer is requested, PI7C8154A responds with a target disconnect operation during the first data phase.

2.9 64-BIT OPERATION

Both the primary and secondary interfaces of the PI7C8154A support 32-bit operation and 64-bit operation. This chapter describes how to use the 64-bit operations as well as the conditions that go along with it.

2.9.1 64-BIT AND 32-BIT TRANSACTIONS INITIATED BY PI7C8154A

64-bit transactions are requested by asserting P_REQ64# on the primary and S_REQ64# on the secondary during the address phase. REQ64# is asserted and deasserted during the same cycles as FRAME#. Under certain conditions, PI7C8154A does not use the 64-bit extension when initiating transactions. In this case, REQ64# is not asserted.

If REQ64# is not asserted, the transaction is initiated as a 32-bit transaction when any of the following conditions are met:

- P_REQ64# was not asserted by the primary during reset (64-bit extension not supported on the primary) for upstream transactions only
- PI7C8154A is initiating an I/O transaction
- PI7C8154A is initiating a special cycle transaction
- PI7C8154A is initiating a configuration transaction
- PI7C8154A is initiating a nonprefetchable memory read transaction
- The address is not QUADWORD aligned
- The address is near the top of a cache line
- A single DWORD read transaction is being performed
- A single or two-DWORD memory write transaction is being performed
- PI7C8154A is resuming memory write transaction after a target disconnect, and ACK64# was not asserted by the target in the previous transaction – does not apply when the previous target termination was a target retry

2.9.2 64-BIT TRANSACTIONS – ADDRESS PHASE

When a transaction using the primary bus 64-bit extension is a single address cycle, the upper 32-bits of the address, AD[63:32], are assumed to be 0 and CBE[7:4] are not defined but driven to valid logic levels during the address phase.

When a transaction using the primary bus 64-bit extension is a dual address cycle, the upper 32-bit of the address, AD[63:32], contain the upper 32-bits of the address and CBE[7:4] contain 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.

2.9.3 64-BIT TRANSACTIONS – DATA PHASE

PI7C8154A asserts REQ64# to indicate it is initiating a 64-bit transfer during memory write transactions. During the data phase, PI7C8154A asserts the following:

- The low 32 bits of data on AD[31:0]
- The low 4 bits on CBE[3:0]

- The high 32 bits of data on AD[63:32]
- The high 4 bits on CBE[7:4]

Every data phase will consist of 64 bits and 8 byte enable bits when PI7C8154A detects ACK64## asserted by the target at the same time it detects DEVSEL#.

For write transactions, PI7C8154A redirects the write data that it has on the AD[63:32] bus to AD[31:0] during the second data phase if it does not detect ACK64# asserted at the same time that it detects DEVSEL# asserted. Also, the CBE[7:4] is redirected to CBE[3:0] during the second data phase.

For 64-bit memory write transactions that end at an odd DWORD boundary, PI7C8154A 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, PI7C8154A drives 8 bits of byte enables on CBE[7:0] when it has asserted REQ64#. CBE[7:0] is always 0 because the only read transactions that use the 64-bit extension are prefetchable memory reads. 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 consist of 64 bits and the target asserts PAR64, which covers AD[63:32] and CBE[7:4]. All data phase consist of 32-bit transactions when the target does not assert ACK64# and asserts DEVSEL#.

2.9.4 64-BIT TRANSACTIONS – RECEIVED BY PI7C8154A

PI7C8154A does one of 2 things when it is the target of a transaction and REQ64# is asserted. PI7C8154A either asserts ACK64# at the same time it asserts DEVSEL# to indicate its ability to perform 64-bit data transfers, or it does not use the 64-bit extension as a target and does not assert ACK64#. PI7C8154A does not assert ACK64# under any of the following conditions:

- REQ64# was not asserted by the initiator
- PI7C8154A is responding to a non-prefetchable memory read transaction
- PI7C8154A is responding to an I/O transaction
- PI7C8154A is responding to a configuration transaction
- Only 1 DWORD of data was read from the target

If PI7C8154A is the target of a 64-bit memory write transaction, it is able to accept 64 bits of data during each data phase. If PI7C8154A is the target of a memory read transaction, it delivers 64 bits of read data during each data phase and drives PAR64 corresponding to AD[63:32] and CBE[7:4] for each data phase. If an odd number of DWORDS is read from the target and PI7C8154A has asserted ACK64# when returning read data to the initiator, PI7C8154A disconnects before the last DWORD is returned. PI7C8154A may have read an odd number of DWORD's because of either a target disconnect or a master latency timer expiration during 32-bit data transfers on the opposite interface.

2.9.5 64-BIT TRANSACTIONS – SUPPORT DURING RESET

PI7C8154A checks P_REQ64# while P_RESET# is asserted to determine whether the 64-bit extensions are connected. If P_REQ64# is HIGH, PI7C8154A knows that the 64-bit extension signals are not connected so it always drives the 64-bit extension outputs to have valid logic levels on the inputs. PI7C8154A will then treat all transactions on the primary as 32-bit. If P_REQ64# is

LOW, the 64-bit signals should be connected to pull-up resistors on the board and PI7C8154A does not perform any input biasing. PI7C8154A can then treat memory write and prefetchable memory read transactions as 64-bit transactions on the primary.

PI7C8154A always asserts S_REQ64# LOW during S_RESET# to indicate that the 64-bit extension is supported on the secondary bus. Individual pull-up resistors must always be supplied for S_AD[63:32], S_CBE[7:4], and S_PAR64.

2.10 TRANSACTION FLOW THROUGH

Transaction flow through refers to data being removed from the read/write buffers concurrently as data is still being written to the buffer.

For reads, flow through occurs when the initiator repeats the delayed transaction while some read data is in the buffer, but the transaction is still ongoing on the target bus. For read flow through to occur, there can be no other reads or writes previously posted in the same direction.

For writes, flow through occurs when PI7C8154A is able to arbitrate for the target bus, initiate the transaction and receive TRDY# from the target, while receiving data from the same transaction on the initiator bus. Flow through can only occur if the writes that were previously posted in the same direction are completed.

2.11 TRANSACTION TERMINATION

This section describes how PI7C8154A returns transaction termination conditions back to the initiator.

The initiator can terminate transactions with one of the following types of termination:

- **Normal termination**

Normal termination occurs when the initiator de-asserts FRAME# at the beginning of the last data phase, and de-asserts IRDYL at the end of the last data phase in conjunction with either TRDY# or STOP# assertion from the target.

- **Master abort**

A master abort occurs 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 de-asserts FRAME# on the next cycle, and then de-asserts IRDY# on the following cycle. IRDY# must be asserted in the same cycle in which FRAME# deasserts. If FRAME# is already deasserted, IRDY# can be deasserted on the next clock cycle following detection of the master abort condition.

The target can terminate transactions with one of the following types of termination:

- **Normal termination**

TRDY# and DEVSEL# asserted in conjunction with FRAME# deasserted and IRDY# asserted.

- **Target retry**

STOP# and DEVSEL# asserted with TRDY# deasserted during the first data phase. No data transfers occur during the transaction. This transaction must be repeated.

- **Target disconnect with data transfer**

STOP#, DEVSEL# and TRDY# asserted. It signals that this is the last data transfer of the transaction.

- **Target disconnect without data transfer**

STOP# and DEVSEL# asserted with TRDY# de-asserted after previous data transfers have been made, indicating that no more data transfers will be made during this transaction.

- **Target abort**

STOP# asserted with DEVSEL# and TRDY# de-asserted. Indicates that 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.

2.11.1 MASTER TERMINATION INITIATED BY PI7C8154A

PI7C8154A, as an initiator, uses normal termination if DEVSEL# is returned by target within five clock cycles of PI7C8154A's assertion of FRAME# on the target bus. As an initiator, PI7C8154A terminates a transaction when the following conditions are met:

- During a delayed write transaction, a single DWORD is delivered.
- During a non-prefetchable read transaction, a single DWORD is transferred from the target.
- During a prefetchable read transaction, a pre-fetch boundary is reached.
- For a posted write transaction, all write data for the transaction is transferred from data buffers to the target.
- For burst transfer, with the exception of "Memory Write and Invalidate" transactions, the master latency timer expires and the PI7C8154A's bus grant is de-asserted.
- The target terminates the transaction with a retry, disconnect, or target abort.

If PI7C8154A is delivering posted write data when it terminates the transaction because the master latency timer expires, it initiates another transaction to deliver the remaining write data. The address of the transaction is updated to reflect the address of the current DWORD to be delivered.

If PI7C8154A is pre-fetching read data when it terminates the transaction because the master latency timer expires, it does not repeat the transaction to obtain more data.

2.11.2 MASTER ABORT RECEIVED BY PI7C8154A

If the initiator initiates a transaction on the target bus and does not detect DEVSEL# returned by the target within five clock cycles of the assertion of FRAME#, PI7C8154A terminates the transaction with a master abort. This sets the received-master-abort bit in the status register corresponding to the target bus.

For delayed read and write transactions, PI7C8154A is able to reflect the master abort condition back to the initiator. When PI7C8154A detects a master abort in response to a delayed transaction, and when the initiator repeats the transaction, PI7C8154A does not respond to the transaction with DEVSEL#, which induces the master abort condition back to the initiator. The transaction is then removed from the delayed transaction queue. When a master abort is received in response to a posted write transaction, PI7C8154A discards the posted write data and makes no more attempts to deliver the data. PI7C8154A sets the received-master-abort bit in the status register when the master abort is received on the primary bus, or it sets the received master abort bit in the secondary

status register when the master abort is received on the secondary interface. When master abort is detected in posted write transaction with both master-abort-mode bit (bit[5] of bridge control register) and the SERR# enable bit (bit 8 of command register for secondary bus) are set, PI7C8154A asserts P_SERR# if the master-abort-on-posted-write is not set. The master-abort-on-posted-write bit is bit 4 of the P_SERR# event disable register (offset 64h).

Note: When PI7C8154A performs a Type 1 to special cycle conversion, 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.

2.11.3 TARGET TERMINATION RECEIVED BY PI7C8154A

When PI7C8154A initiates a transaction on the target bus and the target responds with DEVSEL#, the target can end the transaction with one of the following types
of termination:

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

PI7C8154A handles these terminations in different ways, depending on the type of transaction being performed.

2.11.3.1 DELAYED WRITE TARGET TERMINATION RESPONSE

When PI7C8154A initiates a delayed write transaction, the type of target termination received from the target can be passed back to the initiator.

Table 2-7 shows the response to each type of target termination that occurs during a delayed write transaction.

PI7C8154A repeats a delayed write transaction until one of the following conditions is met:

- PI7C8154A completes at least one data transfer.
- PI7C8154A receives a master abort.
- PI7C8154A receives a target abort.

PI7C8154A makes 2^{24} (default) or 2^{32} (maximum) write attempts resulting in a response of target retry.

Table 2-7 DELAYED WRITE TARGET TERMINATION RESPONSE

Target Termination	Response
Normal	Returning disconnect to initiator with first data transfer only if multiple data phases requested.
Target Retry	Returning target retry to initiator. Continue write attempts to target
Target Disconnect	Returning disconnect to initiator with first data transfer only if multiple data phases requested.
Target Abort	Returning target abort to initiator. Set received target abort bit in target interface status register. Set signaled target abort bit in initiator interface status register.

After the PI7C8154A makes 2^{24} (default) attempts of the same delayed write trans-action on the target bus, PI7C8154A asserts P_SERR# if the SERR# enable bit (bit 8 of command register for the secondary bus) is set and the delayed-write-non-delivery bit is not set. The delayed-write-non-delivery bit is bit 5 of P_SERR# event disable register (offset 64h). PI7C8154A will report system error. See Section 5.4 for a description of system error conditions.

2.11.3.2 POSTED WRITE TARGET TERMINATION RESPONSE

When PI7C8154A initiates a posted write transaction, the target termination cannot be passed back to the initiator. Table 2-8 shows the response to each type of target termination that occurs during a posted write transaction.

Table 2-8 RESPONSE TO POSTED WRITE TARGET TERMINATION

Target Termination	Repsonse
Normal	No additional action.
Target Retry	Repeating write transaction to target.
Target Disconnect	Initiate write transaction for delivering remaining posted write data.
Target Abort	Set received-target-abort bit in the target interface status register. Assert P_SERR# if enabled, and set the signaled-system-error bit in primary status register.

Note that when a target retry or target disconnect is returned and posted write data associated with that transaction remains in the write buffers, PI7C8154A initiates another write transaction to attempt to deliver the rest of the write data. If there is a target retry, the exact same address will be driven as for the initial write trans-action attempt. If a target disconnect is received, the address that is driven on a subsequent write transaction attempt will be updated to reflect the address of the current DWORD. If the initial write transaction is Memory-Write-and-Invalidate transaction, and a partial delivery of write data to the target is performed before a target disconnect is received, PI7C8154A will use the memory write command to deliver the rest of the write data. It is because an incomplete cache line will be transferred in the subsequent write transaction attempt.

After the PI7C8154A makes 2^{24} (default) write transaction attempts and fails to deliver all posted write data associated with that transaction, PI7C8154A asserts P_SERR# if the primary SERR# enable bit is set (bit 8 of command register for secondary bus) and posted-write-non-delivery bit is not set. The posted-write-non-delivery bit is the bit 2 of P_SERR# event disable register (offset 64h). PI7C8154A will report system error. See Section 5.4 for a discussion of system error conditions.

2.11.3.3 DELAYED READ TARGET TERMINATION RESPONSE

When PI7C8154A initiates a delayed read transaction, the abnormal target responses can be passed back to the initiator. Other target responses depend on how much data the initiator requests. Table

2-9 shows the response to each type of target termination that occurs during a delayed read transaction.

PI7C8154A repeats a delayed read transaction until one of the following conditions is met:

- PI7C8154A completes at least one data transfer.
- PI7C8154A receives a master abort.
- PI7C8154A receives a target abort.

PI7C8154A makes 2^{24} (default) read attempts resulting in a response of target retry.

Table 2-9 RESPONSE TO DELAYED READ TARGET TERMINATION

Target Termination	Response
Normal	If prefetchable, target disconnect only if initiator requests more data than read from target. If non-prefetchable, target disconnect on first data phase.
Target Retry	Re-initiate read transaction to target
Target Disconnect	If initiator requests more data than read from target, return target disconnect to initiator.
Target Abort	Return target abort to initiator. Set received target abort bit in the target interface status register. Set signaled target abort bit in the initiator interface status register.

After PI7C8154A makes 2^{24}(default) attempts of the same delayed read transaction on the target bus, PI7C8154A asserts P_SERR# if the primary SERR# enable bit is set (bit 8 of command register for secondary bus) and the delayed-write-non-delivery bit is not set. The delayed-write-non-delivery bit is bit 5 of P_SERR# event disable register (offset 64h). PI7C8154A will report system error. See Section 5.4 for a description of system error conditions.

2.11.4 TARGET TERMINATION INITIATED BY PI7C8154A

PI7C8154A 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.

2.11.4.1 TARGET RETRY

PI7C8154A returns a target retry to the initiator when it cannot accept write data or return read data as a result of internal conditions. PI7C8154A returns a target retry to an initiator when any of the following conditions is met:

FOR DELAYED WRITE TRANSACTIONS:

- The transaction is being entered into the delayed transaction queue.
- Transaction has already been entered into delayed transaction queue, but target response has not yet been received.
- Target response has been received but has not progressed to the head of the return queue.
- The delayed transaction queue is full, and the transaction cannot be queued.
- A transaction with the same address and command has been queued.
- A locked sequence is being propagated across PI7C8154A, and the write transaction is not a locked transaction.
- The target bus is locked and the write transaction is a locked transaction.
- Use more than 16 clocks to accept this transaction.

FOR DELAYED READ TRANSACTIONS:

- 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 target, but it is not yet at head of the read data queue or a posted write transaction precedes it.
- 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 across PI7C8154A, and the read transaction is not a locked transaction.
- PI7C78154B is currently discarding previously pre-fetched read data.
- The target bus is locked and the write transaction is a locked transaction.
- Use more than 16 clocks to accept this transaction.

FOR POSTED WRITE TRANSACTIONS:

- The posted write data buffer does not have enough space for address and at least one DWORD of write data.
- A locked sequence is being propagated across PI7C8154A, and the write transaction is not a locked transaction.
- When a target retry is returned to the initiator of a delayed transaction, the initiator must repeat the transaction with the same address and bus command as well as the data if it is a write transaction, within the time frame specified by the master timeout value. Otherwise, the transaction is discarded from the buffers.

2.11.4.2 TARGET DISCONNECT

PI7C8154A returns a target disconnect to an initiator when one of the following conditions is met:

- PI7C8154A hits an internal address boundary.
- PI7C8154A cannot accept any more write data.
- PI7C8154A has no more read data to deliver.

See Section 2.6.4 for a description of write address boundaries, and Section 2.7.3 for a description of read address boundaries.

2.11.4.3 TARGET ABORT

PI7C8154A returns a target abort to an initiator when one of the following conditions is met:

- PI7C8154A is returning a target abort from the intended target.
- When PI7C8154A returns a target abort to the initiator, it sets the signaled target abort bit in the status register corresponding to the initiator interface.

3 ADDRESS DECODING

PI7C8154A uses three address ranges that control I/O and memory transaction forwarding. These address ranges are defined by base and limit address registers in the configuration space. This chapter describes these address ranges, as well as ISA-mode and VGA-addressing support.

3.1 ADDRESS RANGES

PI7C8154A uses the following address ranges that determine which I/O and memory transactions are forwarded from the primary PCI bus to the secondary PCI bus, and from the secondary bus to the primary bus:

- Two 32-bit I/O address ranges
- Two 32-bit memory-mapped I/O (non-prefetchable memory) ranges
- Two 32-bit prefetchable memory address ranges

Transactions falling within these ranges are forwarded downstream from the primary PCI bus to the secondary PCI bus. Transactions falling outside these ranges are forwarded upstream from the secondary PCI bus to the primary PCI bus.

No address translation is required in PI7C8154A. The addresses that are not marked for downstream are always forwarded upstream.

3.2 I/O ADDRESS DECODING

PI7C8154A uses the following mechanisms that are defined in the configuration space to specify the I/O address space for downstream and upstream forwarding:

- I/O base and limit address registers
- The ISA enable bit
- The VGA mode bit
- The VGA snoop bit

This section provides information on the I/O address registers and ISA mode Section 3.4 provides information on the VGA modes.

To enable downstream forwarding of I/O transactions, the I/O enable bit must be set in the command register in configuration space. All I/O transactions initiated on the primary bus will be ignored if the I/O enable bit is not set. To enable upstream forwarding of I/O transactions, the master enable bit must be set in the command register. If the master-enable bit is not set, PI7C8154A ignores all I/O and memory transactions initiated on the secondary bus.

The master-enable bit also allows upstream forwarding of memory transactions if it is set.

CAUTION
If any configuration state affecting I/O transaction forwarding is changed by a configuration write operation on the primary bus at the same time that I/O transactions are ongoing on the secondary bus, PI7C8154A response to the secondary bus I/O transactions is not predictable. Configure the I/O base and limit address registers, ISA enable bit, VGA mode bit, and VGA snoop bit before setting I/O enable and master enable bits, and change them subsequently only when the primary and secondary PCI buses are idle.

3.2.1 I/O BASE AND LIMIT ADDRESS REGISTER

PI7C8154A implements one set of I/O base and limit address registers in configuration space that define an I/O address range per port downstream forwarding. PI7C8154A supports 32-bit I/O addressing, which allows I/O addresses downstream of PI7C8154A to be mapped anywhere in a 4GB I/O address space.

I/O transactions with addresses that fall inside the range defined by the I/O base and limit registers are forwarded downstream from the primary PCI bus to the secondary PCI bus. I/O transactions with addresses that fall outside this range are forwarded upstream from the secondary PCI bus to the primary PCI bus.

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. When the I/O range is turned off, all I/O trans-actions are forwarded upstream, and no I/O transactions are forwarded downstream. The I/O range has a minimum granularity of 4KB and is aligned on a 4KB boundary. The maximum I/O range is 4GB in size. The I/O base register consists of an 8-bit field at configuration address 1Ch, and a 16-bit field at address 30h. The top 4 bits of the 8-bit field define bits [15:12] of the I/O base address. The bottom 4 bits read only as 1h to indicate that PI7C8154A supports 32-bit I/O addressing. Bits [11:0] of the base address are assumed to be 0, which naturally aligns the base address to a 4KB boundary. The 16 bits contained in the I/O base upper 16 bits register at configuration offset 30h define AD[31:16] of the I/O base address. All 16 bits are read/write. After primary bus reset or chip reset, the value of the I/O base address is initialized to 0000 0000h.

The I/O limit register consists of an 8-bit field at configuration offset 1Dh and a 16-bit field at offset 32h. The top 4 bits of the 8-bit field define bits [15:12] of the I/O limit address. The bottom 4 bits read only as 1h to indicate that 32-bit I/O addressing is supported. Bits [11:0] of the limit address are assumed to be FFFh, which naturally aligns the limit address to the top of a 4KB I/O address block. The 16 bits contained in the I/O limit upper 16 bits register at configuration offset 32h define AD[31:16] of the I/O limit address. All 16 bits are read/write. After primary bus reset or chip reset, the value of the I/O limit address is reset to 0000 0FFFh.

Note: The initial states of the I/O base and I/O limit address registers define an I/O range of 0000 0000h to 0000 0FFFh, which is the bottom 4KB of I/O space. Write these registers with their appropriate values before setting either the I/O enable bit or the master enable bit in the command register in configuration space.

3.2.2 ISA MODE

PI7C8154A supports ISA mode by providing an ISA enable bit in the bridge control register in configuration space. ISA mode modifies the response of PI7C8154A inside the I/O address range in order to support mapping of I/O space in the presence of an ISA bus in the system. This bit only affects the response of PI7C8154A when the transaction falls inside the address range defined by the I/O base and limit address registers, and only when this address also falls inside the first 64KB of I/O space (address bits [31:16] are 0000h).

When the ISA enable bit is set, PI7C8154A does not forward downstream any I/O transactions addressing the top 768 bytes of each aligned 1KB block. Only those transactions addressing the bottom 256 bytes of an aligned 1KB block inside the base and limit I/O address range are forwarded downstream. Transactions above the 64KB I/O address boundary are forwarded as defined by the address range defined by the I/O base and limit registers.

Accordingly, if the ISA enable bit is set, PI7C8154A forwards upstream those I/O transactions addressing the top 768 bytes of each aligned 1KB block within the first 64KB of I/O space. The master enable bit in the command configuration register must also be set to enable upstream forwarding. All other I/O transactions initiated on the secondary bus are forwarded upstream only if they fall outside the I/O address range.

When the ISA enable bit is set, devices downstream of PI7C8154A can have I/O space mapped into the first 256 bytes of each 1KB chunk below the 64KB boundary, or anywhere in I/O space above the 64KB boundary.

3.3 MEMORY ADDRESS DECODING

PI7C8154A has three mechanisms for defining memory address ranges for forwarding of memory transactions:

- Memory-mapped I/O base and limit address registers
- Prefetchable memory base and limit address registers
- VGA mode

This section describes the first two mechanisms. Section 3.4.1 describes VGA mode. To enable downstream forwarding of memory transactions, the memory enable bit must be set in the command register in configuration space. To enable upstream forwarding of memory transactions, the master-enable bit must be set in the command register. The master-enable bit also allows upstream forwarding of I/O transactions if it is set.

CAUTION
If any configuration state affecting memory transaction forwarding is changed by a configuration write operation on the primary bus at the same time that memory transactions are ongoing on the secondary bus, response to the secondary bus memory transactions is not predictable. Configure the memory-mapped I/O base and limit address registers, prefetchable memory base and limit address registers, and VGA mode bit before setting the memory enable and master enable bits, and change them subsequently only when the primary and secondary PCI buses are idle.

3.3.1 MEMORY-MAPPED I/O BASE AND LIMIT ADDRESS REGISTERS

Memory-mapped I/O is also referred to as non-prefetchable memory. Memory addresses that cannot automatically be pre-fetched but that can be conditionally pre-fetched based on command type should be mapped into this space. Read transactions to non-prefetchable space may exhibit side effects; this space may have non-memory-like behavior. PI7C8154A prefetches in this space only if the memory read line or memory read multiple commands are used; transactions using the memory read command are limited to a single data transfer.

The memory-mapped I/O base address and memory-mapped I/O limit address registers define an address range that PI7C8154A uses to determine when to forward memory commands. PI7C8154A forwards a memory transaction from the primary to the secondary interface if the transaction address falls within the memory-mapped I/O address range. PI7C8154A ignores memory transactions initiated on the secondary interface that fall into this address range. Any transactions that fall outside this address range are ignored on the primary interface and are forwarded upstream from the secondary interface (provided that they do not fall into the prefetchable memory range or are not forwarded downstream by the VGA mechanism).

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

The memory-mapped I/O address range is defined by a 16-bit memory-mapped I/O base address register at configuration offset 20h and by a 16-bit memory-mapped I/O limit address register at offset 22h. The top 12 bits of each of these registers correspond to bits [31:20] of the memory address. The low 4 bits are hardwired to 0. The lowest 20 bits of the memory-mapped I/O base address are assumed to be 0 0000h, which results in a natural alignment to a 1MB boundary. The lowest 20 bits of the memory-mapped I/O limit address are assumed to be FFFFFh, which results in an alignment to the top of a 1MB block.

Note: The initial state of the memory-mapped I/O base address register is 0000 0000h. The initial state of the memory-mapped I/O limit address register is 000F FFFFh. Note that the initial states of these registers define a memory-mapped I/O range at the bottom 1MB 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 memory-mapped I/O address range, write the memory-mapped I/O base address register with a value greater than that of the memory-mapped I/O limit address register.

3.3.2 PREFETCHABLE MEMORY BASE AND LIMIT ADDRESS REGISTERS

Locations accessed in the prefetchable memory address range must have true memory-like behavior and must not exhibit side effects when read. This means that extra reads to a prefetchable memory location must have no side effects. PI7C8154A pre-fetches for all types of memory read commands in this address space.

The prefetchable memory base address and prefetchable memory limit address registers define an address range that PI7C8154A uses to determine when to forward memory commands. PI7C8154A forwards a memory transaction from the primary to the secondary interface if the transaction address falls within the prefetchable memory address range. PI7C8154A ignores memory transactions initiated on the secondary interface that fall into this address range. PI7C8154A does not respond to any transactions that fall outside this address range on the primary interface and forwards those transactions upstream from the secondary interface (provided that they do not fall into the memory-mapped I/O range or are not forwarded by the VGA mechanism).

The prefetchable memory range supports 64-bit addressing and provides additional registers to define the upper 32 bits of the memory address range, the prefetchable memory base address upper 32 bits register, and the prefetchable memory limit address upper 32 bits register. For address comparison, a single address cycle (32-bit address) prefetchable memory transaction is treated like a 64-bit address transaction where the upper 32 bits of the address are equal to 0. This upper 32-bit value of 0 is compared to the prefetchable memory base address upper 32 bits register and the prefetchable memory limit address upper 32 bits register. The prefetchable memory base address upper 32 bits register must be 0 to pass any single address cycle transactions downstream.

Prefetchable memory address range has a granularity and alignment of 1MB. Maximum memory address range is 4GB when 32-bit addressing is being used. Prefetchable memory address range is defined by a 16-bit prefetchable memory base address register at configuration offset 24h and by a 16-bit prefetchable memory limit address register at offset 26h. The top 12 bits of each of these

registers correspond to bits [31:20] of the memory address. The lowest 4 bits are hardwired to 1h. The lowest 20 bits of the prefetchable memory base address are assumed to be 0 0000h, which results in a natural alignment to a 1MB boundary. The lowest 20 bits of the prefetchable memory limit address are assumed to be FFFFFh, which results in an alignment to the top of a 1MB block.

Note: The initial state of the prefetchable memory base address register is 0000 0000h. The initial state of the prefetchable memory limit address register is 000F FFFFh. Note that the initial states of these registers define a prefetchable memory range at the bottom 1MB 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 prefetchable memory base address register with a value greater than that of the prefetchable memory limit address register. The entire base value must be greater than the entire limit value, meaning that the upper 32 bits must be considered. Therefore, to disable the address range, the upper 32 bits registers can both be set to the same value, while the lower base register is set greater than the lower limit register. Otherwise, the upper 32-bit base must be greater than the upper 32-bit limit.

3.3.3 PREFETCHABLE MEMORY 64-BIT ADDRESSING REGISTERS

PI7C8154A supports 64-bit memory address decoding for forwarding of dual address memory transactions. Dual address cycle is used for 64-bit addressing. The first address phase of the dual address cycle contains the low 32 bits of the address and the second address phase contains the high 32 bits. The high 32 bits must never be 0 during a dual address cycle.

The prefetchable memory address range is defined by implementing the prefetchable memory base address upper 32 bits register and the prefetchable memory limit address upper 32 bits register. The prefetchable address space can be defined as either:

- Residing entirely in the first 4GB of memory
- Residing entirely above the first 4GB of memory
- Crossing the first 4GB memory boundary

If the prefetchable memory space on the secondary bus resides entirely in the first 4GB of memory, both upper 32 bit register must be set to 0. PI7C8154A then ignores all dual address cycles initiated on the primary interface and forwards all dual address transactions initiated on the secondary interface upstream.

If the prefetchable memory space on the secondary bus resides entirely above the first 4GB of memory, both the prefetchable memory base address upper 32 bit register and the prefetchable memory limit address upper 32 bit register must be initialized to nonzero values. PI7C8154A ignores all single address memory transactions initiated on the primary and forwards all single address memory transactions initiated on the secondary upstream, unless the memory falls within the memory mapped I/O or VGA memory range. A dual address memory transaction is forwarded downstream from the primary if it falls within the address range defined by the prefetchable memory base address, prefetchable memory base address upper 32 bits, prefetchable memory limit address, and prefetchable memory limit address upper 32 bits. If the dual address cycle initiated on the secondary falls outside this address range, it is forwarded upstream to the primary. PI7C8154A does not respond to a dual address cycle initiated on the primary that falls outside this address range, or to a dual address cycle initiated on the secondary that falls within the address range.

If the prefetchable memory space on the secondary bus resides on top of the 4GB boundary, the prefetchable memory base address upper 32 bit register is set to 0 and the prefetchable memory limit address upper 32 bit register is initialized to a nonzero value. Single address cycle memory transactions are compared to the prefetchable memory base address register only. A transaction initiated on the primary is forwarded downstream if the address is greater than or equal to the base address. A transaction initiated on the secondary is forwarded upstream if the address is less than the base address. Dual address cycles are compared to the prefetchable memory limit address and the prefetchable memory limit address upper 32 bit register. If the address of the dual address cycle is less than or equal to the limit, the transaction is forwarded downstream from the primary and is ignored on the secondary. If the address of the dual address cycle is greater than this limit, the transaction is ignored on the primary and is forwarded upstream from the secondary.

The prefetchable memory base address upper 32 bit register is located at offset 28h of the configuration register and the prefetchable memory limit address upper 32 bit register is located at offset 2Ch. Both registers are reset to 0.

3.4 VGA SUPPORT

PI7C8154A provides two modes for VGA support:

- VGA mode, supporting VGA-compatible addressing
- VGA snoop mode, supporting VGA palette forwarding

3.4.1 VGA MODE

When a VGA-compatible device exists downstream from PI7C8154A, set the VGA mode bit in the bridge control register in configuration space to enable VGA mode. When PI7C8154A 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 base and limit address registers. PI7C8154A ignores transactions initiated on the secondary interface addressing these locations.

The VGA frame buffer consists of the following memory address range:

000A 0000h–000B FFFFh

Read transactions to frame buffer memory are treated as non-prefetchable. PI7C8154A 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 are in the range of 3B0h–3BBh and 3C0h–3DFh I/O. These I/O addresses are aliases every 1KB throughout the first 64KB of I/O space. This means that address bits [5:10] are not decoded and can be any value, while address bits [31:16] must be all 0's. VGA BIOS addresses starting at C0000h are not decoded in VGA mode.

3.4.2 VGA SNOOP MODE

PI7C8154A provides VGA snoop mode, allowing for VGA palette write transactions to be forwarded downstream. This mode is used when a graphics device downstream from PI7C8154A needs to snoop or respond to VGA palette write transactions. To enable the mode, set the VGA

snoop bit in the command register in configuration space. Note that PI7C8154A claims VGA palette write transactions by asserting DEVSEL# in VGA snoop mode.

When VGA snoop bit is set, PI7C8154A forwards downstream transactions within the 3C6h, 3C8h and 3C9h I/O addresses space. Note that these addresses are also forwarded as part of the VGA compatibility mode previously described. Again, address bits [15:10] are not decoded, while address bits [31:16] must be equal to 0, which means that these addresses are aliases every 1KB throughout the first 64KB of I/O space.

Note: If both the VGA mode bit and the VGA snoop bit are set, PI7C8154A behaves in the same way as if only the VGA mode bit were set.

4 TRANSACTION ORDERING

To maintain data coherency and consistency, PI7C8154A complies with the ordering rules set forth in the PCI Local Bus Specification, Revision 2.2, for transactions crossing the bridge. This chapter describes the ordering rules that control transaction forwarding across PI7C8154A.

4.1 TRANSACTIONS GOVERNED BY ORDERING RULES

Ordering relationships are established for the following classes of transactions crossing PI7C8154A:

Posted write transactions, comprised of memory write and memory write and invalidate transactions.
Posted write transactions complete at the source before they complete at the destination; that is, data is written into intermediate data buffers before it reaches the target.

Delayed write request transactions, comprised of I/O write and configuration write transactions.
Delayed write requests are terminated by target retry on the initiator bus and are queued in the delayed transaction queue. A delayed write transaction must complete on the target bus before it completes on the initiator bus.

Delayed write completion transactions, comprised of I/O write and configuration write transactions.
Delayed write completion transactions complete on the target bus, and the target response is queued in the buffers. A delayed write completion transaction proceeds in the direction opposite that of the original delayed write request; that is, a delayed write completion transaction proceeds from the target bus to the initiator bus.

Delayed read request transactions, comprised of all memory read, I/O read, and configuration read transactions.
Delayed read requests are terminated by target retry on the initiator bus and are queued in the delayed transaction queue.

Delayed read completion transactions, comprised of all memory read, I/O read, & configuration read transactions.
Delayed read completion transactions complete on the target bus, and the read data is queued in the read data buffers. A delayed read completion transaction proceeds in the direction opposite that of

the original delayed read request; that is, a delayed read completion transaction proceeds from the target bus to the initiator bus.

PI7C8154A does not combine or merge write transactions:

- PI7C8154A does not combine separate write transactions into a single write transaction—this optimization is best implemented in the originating master.
- PI7C8154A does not merge bytes on separate masked write transactions to the same DWORD address—this optimization is also best implemented in the originating master.
- PI7C8154A 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.

4.2 GENERAL ORDERING GUIDELINES

Independent transactions on primary and secondary buses have a relationship only when those transactions cross PI7C8154A.

The following general ordering guidelines govern transactions crossing PI7C8154A:

- The ordering relationship of a transaction with respect to other transactions is determined when the transaction completes, that is, when a transaction ends with a termination other than target retry.
- 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 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. PI7C8154A 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 for PI7C8154A and must also be true for other bus agents. Otherwise, a deadlock can occur.
- PI7C8154A accepts posted write transactions, regardless of the state of completion of any delayed transactions being forwarded across PI7C8154A.

4.3 ORDERING RULES

Table 4-1 shows the ordering relationships of all the transactions and refers by number to the ordering rules that follow.

Table 4-1 SUMMARY OF TRANSACTION ORDERING

Pass	Posted Write	Delayed Read Request	Delayed Write Request	Delayed Read Completion	Delayed Write Completion
Posted Write	No[1]	Yes[5]	Yes[5]	Yes[5]	Yes[5]
Delayed Read Request	No[2]	No	No	Yes	Yes

Pass	Posted Write	Delayed Read Request	Delayed Write Request	Delayed Read Completion	Delayed Write Completion
Delayed Write Request	No[4]	No	No	Yes	Yes
Delayed Read Completion	No[3]	Yes	Yes	No	No
Delayed Write Completion	Yes	Yes	Yes	No	No

Note: The superscript accompanying some of the table entries refers to any applicable ordering rule listed in this section. Many entries are not governed by these ordering rules; therefore, the implementation can choose whether or not the transactions pass each other.

The entries without superscripts reflect the PI7C8154A's implementation choices.

The following ordering rules describe the transaction relationships. Each ordering rule is followed by an explanation, and the ordering rules are referred to by number in Table 4-1. These ordering rules apply to posted write transactions, delayed write and read requests, and delayed write and read completion transactions crossing PI7C8154A in the same direction. Note that delayed completion transactions cross PI7C8154A 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. The subsequent posted write transaction can be setting a flag that covers the data in the first posted write transaction; if the second transaction were to complete before the first transaction, a device checking the flag could subsequently consume stale data.

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, so if the read transaction were to pass the write transaction, it would return stale data.

3. A delayed read completion must ''pull'' ahead of previously queued posted write data traveling in the same direction. 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 of PI7C8154A 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 a reading to a status register of the initiator of the posted write data and therefore should not complete until the write transaction is complete.

4. Delayed write requests cannot pass previously queued posted write data. For posted memory write transactions, the delayed write transaction can set a flag that covers the data in the posted write transaction. If the delayed write request were to complete before the earlier posted write transaction, a device checking the flag could subsequently consume stale data.

5. Posted write transactions must be given opportunities to pass delayed read and write requests and completions. Otherwise, deadlocks may occur when some bridges which support delayed transactions and other bridges which do not support delayed transactions are being used in the same system. A fairness algorithm is used to arbitrate between the posted write queue and the delayed transaction queue.

4.4 DATA SYNCHRONIZATION

Data synchronization refers to the relationship between interrupt signaling and data delivery. The *PCI Local Bus Specification*, Revision 2.2, provides the following alternative methods for synchronizing data and interrupts:

- The device signaling the interrupt performs a read of the data just written (software).
- The device driver performs a read operation to any register in the interrupting device before accessing data written by the device (software).
- System hardware guarantees that write buffers are flushed before interrupts are forwarded.

PI7C8154A does not have a hardware mechanism to guarantee data synchronization for posted write transactions. Therefore, all posted write transactions must be followed by a read operation, either from the device to the location just written (or some other location along the same path), or from the device driver to one of the device registers.

5 ERROR HANDLING

PI7C8154A checks, forwards, and generates parity on both the primary and secondary interfaces. To maintain transparency, PI7C8154A always tries to forward the existing parity condition on one bus to the other bus, along with address and data. PI7C8154A always attempts to be transparent when reporting errors, but this is not always possible, given the presence of posted data and delayed transactions.

To support error reporting on the PCI bus, PI7C8154A implements the following:

- PERR# and SERR# signals on both the primary and secondary interfaces
- Primary status and secondary status registers
- The device-specific P_SERR# event disable register

This chapter provides detailed information about how PI7C8154A handles errors. It also describes error status reporting and error operation disabling.

5.1 ADDRESS PARITY ERRORS

PI7C8154A checks address parity for all transactions on both buses, for all address and all bus commands. When PI7C8154A detects an address parity error on the primary interface, the following events occur:

- If the parity error response bit is set in the command register, PI7C8154A does not claim the transaction with P_DEVSEL#; this may allow the transaction to terminate in a master abort. If parity error response bit is not set, PI7C8154A proceeds normally and accepts the transaction if it is directed to or across PI7C8154A.
- PI7C8154A sets the detected parity error bit in the status register.
- PI7C8154A asserts P_SERR# and sets signaled system error bit in the status register, if both the following conditions are met:
  - The SERR# enable bit is set in the command register
  - The parity error response bit is set in the command register

When PI7C8154A detects an address parity error on the secondary interface, the following events occur:

- If the parity error response bit is set in the bridge control register, PI7C8154A does not claim the transaction with S_DEVSEL#; this may allow the transaction to terminate in a master

abort. If parity error response bit is not set, PI7C8154A proceeds normally and accepts transaction if it is directed to or across PI7C8154A.
- PI7C8154A sets the detected parity error bit in the secondary status register
- PI7C8154A asserts P_SERR# and sets signaled system error bit in status register, if both of the following conditions are met:

  - The SERR# enable bit is set in the command register
  - The parity error response bit is set in the bridge control register

5.2 DATA PARITY ERRORS

When forwarding transactions, PI7C8154A attempts to pass the data parity condition from one interface to the other unchanged, whenever possible, to allow the master and target devices to handle the error condition.

The following sections describe, for each type of transaction, the sequence of events that occurs when a parity error is detected and the way in which the parity condition is forwarded across PI7C8154A.

5.2.1 CONFIGURATION WRITE TRANSACTIONS TO CONFIGURATION SPACE

When PI7C8154A detects a data parity error during a Type 0 configuration write transaction to PI7C8154A configuration space, the following events occur:

If the parity error response bit is set in the command register, PI7C8154A asserts P_TRDY# and writes the data to the configuration register. PI7C8154A also asserts P_PERR#. If the parity error response bit is not set, PI7C8154A does not assert P_PERR#.

PI7C8154A sets the detected parity error bit in the status register, regardless of the state of the parity error response bit.

5.2.2 READ TRANSACTIONS

When PI7C8154A 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 PI7C8154A detects a read data parity error on the secondary bus, the following events occur:

- PI7C8154A asserts S_PERR# two cycles following the data transfer, if the secondary interface parity error response bit is set in the bridge control register.
- PI7C8154A sets the detected parity error bit in the secondary status register.
- PI7C8154A 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.
- PI7C8154A forwards the bad parity with the data back to the initiator on the primary bus. If the data with the bad parity is pre-fetched and is not read by the initiator on the primary bus, the data is discarded and the data with bad parity is not returned to the initiator.
- PI7C8154A completes the transaction normally.

For upstream transactions, when PI7C8154A detects a read data parity error on the primary bus, the following events occur:

- PI7C8154A asserts P_PERR# 2 cycles following the data transfer, if the primary interface parity error response bit is set in the command register.
- PI7C8154A sets the detected parity error bit in the primary status register.
- PI7C8154A sets the data parity detected bit in the primary status register, if the primary interface parity-error-response bit is set in the command register.
- PI7C8154A forwards the bad parity with the data back to the initiator on the secondary bus. If the data with the bad parity is pre-fetched and is not read by the initiator on the secondary bus, the data is discarded and the data with bad parity is not returned to the initiator.
- PI7C8154A completes the transaction normally.

PI7C8154A 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 PI7C8154A detects PERR# asserted while returning read data to the initiator, PI7C8154A does not take any further action and completes the transaction normally.

5.2.3 DELAYED WRITE TRANSACTIONS

When PI7C8154A detects a data parity error during a delayed write transaction, the initiator drives data and data parity, and the target checks parity and conditionally asserts PERR#.

For delayed write transactions, a parity error can occur at the following times:

- During the original delayed write request transaction
- When the initiator repeats the delayed write request transaction
- When PI7C8154A completes the delayed write transaction to the target

When a delayed write transaction is normally queued, the address, command, address parity, data, byte enable bits, and data parity are all captured and a target retry is returned to the initiator. When PI7C8154A detects a parity error on the write data for the initial delayed write request transaction, the following events occur:

- If the parity-error-response bit corresponding to the initiator bus is set, PI7C8154A asserts TRDY# to the initiator and the transaction is not queued. If multiple data phases are requested, STOP# is also asserted to cause a target disconnect. Two cycles after the data transfer, PI7C8154A also asserts PERR#.
- If the parity-error-response bit is not set, PI7C8154A returns a target retry. It queues the transaction as usual. PI7C8154A does not assert PERR#. In this case, the initiator repeats the transaction.
- PI7C8154A sets the detected-parity-error bit in the status register corresponding to the initiator bus, regardless of the state of the parity-error-response bit.

Note: If parity checking is turned off and data parity errors have occurred for queued or subsequent delayed write transactions on the initiator bus, it is possible that the initiator's re-attempts of the write transaction may not match the original queued delayed write information contained in the delayed transaction queue. In this case, a master timeout condition may occur, possibly resulting in a system error (P_SERR# assertion).

For downstream transactions, when PI7C8154A is delivering data to the target on the secondary bus and S_PERR# is asserted by the target, the following events occur:

- PI7C8154A sets the secondary interface data parity detected bit in the secondary status register, if the secondary parity error response bit is set in the bridge control register.
- PI7C8154A captures the parity error condition to forward it back to the initiator on the primary bus.

Similarly, for upstream transactions, when PI7C8154A is delivering data to the target on the primary bus and P_PERR# is asserted by the target, the following events occur:

- PI7C8154A sets the primary interface data-parity-detected bit in the status register, if the primary parity-error-response bit is set in the command register.
- PI7C8154A captures the parity error condition to forward it back to the initiator on the secondary bus.

A delayed write transaction is completed on the initiator bus when the initiator repeats the write transaction with the same address, command, data, and byte enable bits as the delayed write command that is at the head of the posted data queue. Note that the parity bit is not compared when determining whether the transaction matches those in the delayed transaction queues.

Two cases must be considered:

- When parity error is detected on the initiator bus on a subsequent re-attempt of the transaction and was not detected on the target bus.
- When parity error is forwarded back from the target bus

For downstream delayed write transactions, when the parity error is detected on the initiator bus and PI7C8154A has write status to return, the following events occur:

- PI7C8154A first asserts P_TRDY# and then asserts P_PERR# two cycles later, if the primary interface parity-error-response bit is set in the command register.
- PI7C8154A sets the primary interface parity-error-detected bit in the status register.
- Because there was not an exact data and parity match, the write status is not returned and the transaction remains in the queue.

Similarly, for upstream delayed write transactions, when the parity error is detected on the initiator bus and PI7C8154A has write status to return, the following events occur:

- PI7C8154A first asserts S_TRDY# and then asserts S_PERR# two cycles later; if the secondary interface parity-error-response bit is set in the bridge control register (offset 3Ch).
- PI7C8154A sets the secondary interface parity-error-detected bit in the secondary status register.
- Because there was not an exact data and parity match, the write status is not returned and the transaction remains in the queue.

For downstream transactions, where the parity error is being passed back from the target bus and the parity error condition was not originally detected on the initiator bus, the following events occur:

- Bridge asserts P_PERR# two cycles after the data transfer, if the following are both true:
  - The parity-error-response bit is set in the command register of the primary interface
  - The parity-error-response bit is set in the bridge control register of the secondary interface

- Bridge completes the transaction normally.

For upstream transactions, when the parity error is being passed back from the target bus and the parity error condition was not originally detected on the initiator bus, the following events occur:

- Bridge asserts S_PERR# two cycles after the data transfer, if the following are both true:
  - The parity error response bit is set in the command register of the primary interface.
  - The parity error response bit is set in the bridge control register of the secondary interface.
- Bridge completes the transaction normally.

5.2.4 POSTED WRITE TRANSACTIONS

During downstream posted write transactions, when the bridge responds as a target, it detects a data parity error on the initiator (primary) bus and the following events occur:

- Bridge asserts P_PERR# two cycles after the data transfer, if the parity error response bit is set in the command register of primary interface.
- Bridge sets the parity error detected bit in the status register of the primary interface.
- Bridge captures and forwards the bad parity condition to the secondary bus.
- Bridge completes the transaction normally.

Similarly, during upstream posted write transactions, when the bridge responds as a target, it detects a data parity error on the initiator (secondary) bus, the following events occur:

- Bridge asserts S_PERR# two cycles after the data transfer, if the parity error response bit is set in the bridge control register of the secondary interface.
- Bridge sets the parity error detected bit in the status register of the secondary interface.
- Bridge captures and forwards the bad parity condition to the primary bus.
- Bridge completes the transaction normally.

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

- Bridge sets the data parity detected bit in the status register of secondary interface, if the parity error response bit is set in the bridge control register of the secondary interface.
- Bridge asserts P_SERR# and sets the signaled system error bit in the status register, if all the following conditions are met:
  - The SERR# enable bit is set in the command register.
  - The posted write parity error bit of P_SERR# event disable register is not set.
  - The parity error response bit is set in the bridge control register of the secondary interface.
  - The parity error response bit is set in the command register of the primary interface.
  - Bridge has not detected the parity error on the primary (initiator) bus which the parity error is not forwarded from the primary bus to the secondary bus.

During upstream write transactions, when a data parity error is reported on the target (primary) bus by the target's assertion of P_PERR#, the following events occur:

- Bridge sets the data parity detected bit in the status register, if the parity error response bit is set in the command register of the primary interface.
- Bridge asserts P_SERR# and sets the signaled system error bit in the status register, if all the following conditions are met:

  - The SERR# enable bit is set in the command register
  - The parity error response bit is set in the bridge control register of the secondary interface
  - The parity error response bit is set in the command register of the primary interface
  - Bridge has not detected the parity error on the secondary (initiator) bus, which the parity error is not forwarded from the secondary bus to the primary bus

Assertion of P_SERR# is used to signal the parity error condition when the initiator does not know that the error occurred. Because the data has already been delivered with no errors, there is no other way to signal this information back to the initiator. If the parity error has forwarded from the initiating bus to the target bus, P_SERR# will not be asserted.

5.3 DATA PARITY ERROR REPORTING

In the previous sections, the responses of the bridge to data parity errors are presented according to the type of transaction in progress. This section organizes the responses of the bridge to data parity errors according to the status bits that the bridge sets and the signals that it asserts.

Table 5-1 shows setting the detected parity error bit in the status register, corresponding to the primary interface. This bit is set when PI7C8154A detects a parity error on the primary interface.

Table 5-1 SETTING THE PRIMARY INTERFACE DETECTED PARITY ERROR BIT (bit 31 of offset 04h)

Primary Detected Parity Error Bit	Transaction Type	Direction	Bus Where Error Was Detected	Primary/ Secondary Parity Error Response Bits
0	Read	Downstream	Primary	x / x
0	Read	Downstream	Secondary	x / x
1	Read	Upstream	Primary	x / x
0	Read	Upstream	Secondary	x / x
1	Posted Write	Downstream	Primary	x / x
0	Posted Write	Downstream	Secondary	x / x
0	Posted Write	Upstream	Primary	x / x
0	Posted Write	Upstream	Secondary	x / x
1	Delayed Write	Downstream	Primary	x / x
0	Delayed Write	Downstream	Secondary	x / x
0	Delayed Write	Upstream	Primary	x / x
0	Delayed Write	Upstream	Secondary	x / x

Note: x=don't care

Table 5-2 shows setting the detected parity error bit in the secondary status register, corresponding to the secondary interface. This bit is set when PI7C8154A detects a parity error on the secondary interface.

Table 5-2 SETTING THE SECONDARY INTERFACE DETECTED PARITY ERROR BIT

Secondary Detected Parity Error Bit	Transaction Type	Direction	Bus Where Error Was Detected	Primary/ Secondary Parity Error Response Bits
0	Read	Downstream	Primary	x / x
1	Read	Downstream	Secondary	x / x

Secondary Detected Parity Error Bit	Transaction Type	Direction	Bus Where Error Was Detected	Primary/ Secondary Parity Error Response Bits
0	Read	Upstream	Primary	x / x
0	Read	Upstream	Secondary	x / x
0	Posted Write	Downstream	Primary	x / x
0	Posted Write	Downstream	Secondary	x / x
0	Posted Write	Upstream	Primary	x / x
1	Posted Write	Upstream	Secondary	x / x
0	Delayed Write	Downstream	Primary	x / x
0	Delayed Write	Downstream	Secondary	x / x
0	Delayed Write	Upstream	Primary	x / x
1	Delayed Write	Upstream	Secondary	x / x

Note: x=don't care

Table 5-3 shows setting data parity detected bit in the primary interface's status register. This bit is set under the following conditions:

- PI7C8154A must be a master on the primary bus.
- The parity error response bit in the command register, corresponding to the primary interface, must be set.
- The P_PERR# signal is detected asserted or a parity error is detected on the primary bus.

Table 5-3 SETTING THE PRIMARY INTERFACE DATA PARITY DETECTED BIT (bit 24 of offset 04h)

Primary Data Parity Bit	Transaction Type	Direction	Bus Where Error Was Detected	Primary / Secondary Parity Error Response Bits
0	Read	Downstream	Primary	x / x
0	Read	Downstream	Secondary	x / x
1	Read	Upstream	Primary	1 / x
0	Read	Upstream	Secondary	x / x
0	Posted Write	Downstream	Primary	x / x
0	Posted Write	Downstream	Secondary	x / x
1	Posted Write	Upstream	Primary	1 / x
0	Posted Write	Upstream	Secondary	x / x
0	Delayed Write	Downstream	Primary	x / x
0	Delayed Write	Downstream	Secondary	x / x
1	Delayed Write	Upstream	Primary	1 / x
0	Delayed Write	Upstream	Secondary	x / x

Note: x=don't care

Table 5-4 shows setting the data parity detected bit in the status register of secondary interface. This bit is set under the following conditions:

- The PI7C8154A must be a master on the secondary bus.
- The parity error response bit must be set in the bridge control register of secondary interface.
- The S_PERR# signal is detected asserted or a parity error is detected on the secondary bus.

Table 5-4 SETTING THE SECONDARY INTERFACE DATA PARITY DETECTED BIT

Secondary Detected Parity Detected Bit	Transaction Type	Direction	Bus Where Error Was Detected	Primary / Secondary Parity Error Response Bits
0	Read	Downstream	Primary	x / x
1	Read	Downstream	Secondary	x / 1
0	Read	Upstream	Primary	x / x
0	Read	Upstream	Secondary	x / x
0	Posted Write	Downstream	Primary	x / x
1	Posted Write	Downstream	Secondary	x / 1

Secondary Detected Parity Detected Bit	Transaction Type	Direction	Bus Where Error Was Detected	Primary / Secondary Parity Error Response Bits
0	Posted Write	Upstream	Primary	x / x
0	Posted Write	Upstream	Secondary	x / x
0	Delayed Write	Downstream	Primary	x / x
1	Delayed Write	Downstream	Secondary	x / 1
0	Delayed Write	Upstream	Primary	x / x
0	Delayed Write	Upstream	Secondary	x / x

Note: x=don't care

Table 5-5 shows assertion of P_PERR#. This signal is set under the following conditions:

- PI7C8154A is either the target of a write transaction or the initiator of a read transaction on the primary bus.
- The parity-error-response bit must be set in the command register of primary interface.
- PI7C8154A detects a data parity error on the primary bus or detects S_PERR# asserted during the completion phase of a downstream delayed write transaction on the target (secondary) bus.

Table 5-5 ASSERTION OF P_PERR#

P_PERR#	Transaction Type	Direction	Bus Where Error Was Detected	Primary/ Secondary Parity Error Response Bits
1 (de-asserted)	Read	Downstream	Primary	x / x
1	Read	Downstream	Secondary	x / x
0 (asserted)	Read	Upstream	Primary	1 / x
1	Read	Upstream	Secondary	x / x
0	Posted Write	Downstream	Primary	1 / x
1	Posted Write	Downstream	Secondary	x / x
1	Posted Write	Upstream	Primary	x / x
1	Posted Write	Upstream	Secondary	x / x
0	Delayed Write	Downstream	Primary	1 / x
0[2]	Delayed Write	Downstream	Secondary	1 / 1
1	Delayed Write	Upstream	Primary	x / x
1	Delayed Write	Upstream	Secondary	x / x

Notes: x=don't care

[2]=The parity error was detected on the target (secondary) bus but not on the initiator (primary) bus.

Table 5-6 shows assertion of S_PERR# that is set under the following conditions:

- PI7C8154A is either the target of a write transaction or the initiator of a read transaction on the secondary bus.
- The parity error response bit must be set in the bridge control register of secondary interface.
- PI7C8154A detects a data parity error on the secondary bus or detects P_PERR# asserted during the completion phase of an upstream delayed write transaction on the target (primary) bus.

Table 5-6 ASSERTION OF S_PERR#

S_PERR#	Transaction Type	Direction	Bus Where Error Was Detected	Primary/ Secondary Parity Error Response Bits
1 (de-asserted)	Read	Downstream	Primary	x / x
0 (asserted)	Read	Downstream	Secondary	x / 1
1	Read	Upstream	Primary	x / x
1	Read	Upstream	Secondary	x / x
1	Posted Write	Downstream	Primary	x / x
1	Posted Write	Downstream	Secondary	x / x
1	Posted Write	Upstream	Primary	x / x
0	Posted Write	Upstream	Secondary	x / 1
1	Delayed Write	Downstream	Primary	x / x

S_PERR#	Transaction Type	Direction	Bus Where Error Was Detected	Primary/ Secondary Parity Error Response Bits
1	Delayed Write	Downstream	Secondary	x / x
0[2]	Delayed Write	Upstream	Primary	1 / 1
0	Delayed Write	Upstream	Secondary	x / 1

Note: x=don't care

[2]=The parity error was detected on the target (secondary) bus but not on the initiator (primary) bus.

Table 5-7 shows assertion of P_SERR#. This signal is set under the following conditions:

- PI7C8154A has detected P_PERR# asserted on an upstream posted write transaction or S_PERR# asserted on a downstream posted write transaction.
- PI7C8154A did not detect the parity error as a target of the posted write transaction.
- The parity error response bit on the command register and the parity error response bit on the bridge control register must both be set.
- The SERR# enable bit must be set in the command register.

Table 5-7 ASSERTION OF P_SERR# FOR DATA PARITY ERRORS

P_SERR#	Transaction Type	Direction	Bus Where Error Was Detected	Primary / Secondary Parity Error Response Bits
1 (de-asserted)	Read	Downstream	Primary	x / x
1	Read	Downstream	Secondary	x / x
1	Read	Upstream	Primary	x / x
1	Read	Upstream	Secondary	x / x
1	Posted Write	Downstream	Primary	x / x
0[2] (asserted)	Posted Write	Downstream	Secondary	1 / 1
0[3]	Posted Write	Upstream	Primary	1 / 1
1	Posted Write	Upstream	Secondary	x / x
1	Delayed Write	Downstream	Primary	x / x
1	Delayed Write	Downstream	Secondary	x / x
1	Delayed Write	Upstream	Primary	x / x
1	Delayed Write	Upstream	Secondary	x / x

Note: x=don't care

5.4 SYSTEM ERROR (SERR#) REPORTING

PI7C8154A uses the P_SERR# signal to report conditionally a number of system error conditions in addition to the special case parity error conditions described in Section 5.2.3.

Whenever assertion of P_SERR# is discussed in this document, it is assumed that the following conditions apply:

- For the bridge to assert P_SERR# for any reason, the SERR# enable bit must be set in the command register.
- Whenever the bridge asserts P_SERR#, PI7C8154A must also set the signaled system error bit in the status register.

In compliance with the PCI-to-PCI Bridge Architecture Specification, the bridge asserts P_SERR# when it detects the secondary SERR# input, S_SERR#, asserted and the SERR# forward enable bit is set in the bridge control register. In addition, the bridge also sets the received system error bit in the secondary status register.

The bridge also conditionally asserts P_SERR# for any of the following reasons:

- Target abort detected during posted write transaction.
- Master abort detected during posted write transaction.
- Posted write data discarded after 2^{24} (default) attempts to deliver (2^{24} target retries received).
- Parity error reported on target bus during posted write transaction (see previous section)
- Delayed write data discarded after 2^{24} (default) attempts to deliver (2^{24} target retries received)
- Delayed read data cannot be transferred from target after 2^{24} (default) attempts (2^{24} target retries received)
- Master timeout on delayed transaction

The device-specific P_SERR# status register reports the reason for the assertion of P_SERR#. Most of these events have additional device-specific disable bits in the P_SERR# event disable register that make it possible to mask out P_SERR# assertion for specific events. The master timeout condition has a SERR# enable bit for that event in the bridge control register and therefore does not have a device-specific disable bit.

6 EXCLUSIVE ACCESS

This chapter describes the use of the LOCK# signal to implement exclusive access to a target for transactions that cross the bridge.

6.1 CONCURRENT LOCKS

The primary and secondary bus lock mechanisms operate concurrently except when a locked transaction crosses the bridge. A primary master can lock a primary target without affecting the status of the lock on the secondary bus, and vice versa. This means that a primary master can lock a primary target at the same time that a secondary master locks a secondary target.

6.2 ACQUIRING EXCLUSIVE ACCESS ACROSS PI7C8154A

For any PCI bus, before acquiring access to the LOCK# signal and starting a series of locked transactions, the initiator must first check that both of the following conditions are met:

- The PCI bus must be idle.
- The LOCK# signal must be de-asserted.

The initiator leaves the LOCK# signal de-asserted during the address phase and asserts LOCK# one clock cycle later. Once a data transfer is completed from the target, the target lock has been achieved.

6.2.1 LOCKED TRANSACTIONS IN DOWNSTREAM DIRECTION

Locked transactions can cross the bridge only in the downstream direction, from the primary bus to the secondary bus.

When the target resides on another PCI bus, the master must acquire not only the lock on its own PCI bus but also the lock on every bus between its bus and the target's bus. When the bridge detects on the primary bus, an initial locked transaction intended for a target on the secondary bus,

the bridge samples the address, transaction type, byte enable bits, and parity, as described in Section 2.7.4. It also samples the lock signal. If there is a lock established between 2 ports or the target bus is already locked by another master, then the current lock cycle is retried without forward. Because a target retry is signaled to the initiator, the initiator must relinquish the lock on the primary bus, and therefore the lock is not yet established.

The first locked transaction must be a memory read transaction. Subsequent locked transactions can be memory read or memory write transactions. Posted memory write transactions that are a part of the locked transaction sequence are still posted. Memory read transactions that are a part of the locked transaction sequence are not pre-fetched.

When the locked delayed memory read request is queued, the bridge does not queue any more transactions until the locked sequence is finished. The bridge signals a target retry to all transactions initiated subsequent to the locked read transaction that are intended for targets on the other side of the bridge. The bridge allows any transactions queued before the locked transaction to complete before initiating the locked transaction.

When the locked delayed memory read request transaction moves to the head of the delayed transaction queue, the bridge initiates the transaction as a locked read transaction by de-asserting LOCK# on the target bus during the first address phase, and by asserting LOCK# one cycle later. If LOCK# is already asserted (used by another initiator), PI7C8154A waits to request access to the secondary bus until LOCK# is de-asserted when the target bus is idle. Note that the existing lock on the target bus could not have crossed PI7C8154A. Otherwise, the pending queued locked transaction would not have been queued. When PI7C8154A is able to complete a data transfer with the locked read transaction, the lock is established on the secondary bus.

When the initiator repeats the locked read transaction on the primary bus with the same address, transaction type, and byte enable bits, PI7C8154A transfers the read data back to the initiator, and the lock is then also established on the primary bus.

For PI7C8154A to recognize and respond to the initiator, the initiator's subsequent attempts of the read transaction must use the locked transaction sequence (de-assert LOCK# during address phase, and assert LOCK# one cycle later). If the LOCK# sequence is not used in subsequent attempts, a master timeout condition may result. When a master timeout condition occurs, SERR# is conditionally asserted (see Section 5.4), the read data and queued read transaction are discarded, and the LOCK# signal is de-asserted on the target bus.

Once the intended target has been locked, any subsequent locked transactions initiated on the initiator bus that are forwarded by the bridge are driven as locked transactions on the target bus.

The first transaction to establish LOCK# must be Memory Read. If the first transaction is not Memory read, the following transactions behave accordingly:

- Type 0 Configuration Read/Write induces master abort.
- Type 1 Configuration Read/Write induces master abort.
- I/O Read induces master abort.
- I/O Write induces master abort.
- Memory Write induces master abort.

When the bridge receives a target abort or a master abort in response to the delayed locked read transaction, this status is passed back to the initiator, and no locks are established on either the target or the initiator bus. The bridge resumes forwarding unlocked transactions in both directions.

6.2.2 LOCKED TRANSACTION IN UPSTREAM DIRECTION

The bridge ignores upstream lock and transactions. The bridge will pass these transactions as normal transactions without lock established.

6.3 ENDING EXCLUSIVE ACCESS

After the lock has been acquired on both initiator and target buses, the bridge must maintain the lock on the target bus for any subsequent locked transactions until the initiator relinquishes the lock.

The only time a target-retry causes the lock to be relinquished is on the first transaction of a locked sequence. On subsequent transactions in the sequence, the target retry has no effect on the status of the lock signal.

An established target lock is maintained until the initiator relinquishes the lock. The bridge does not know whether the current transaction is the last one in a sequence of locked transactions until the initiator de-asserts the LOCK# signal at end of the transaction.

When the last locked transaction is a delayed transaction, the bridge has already completed the transaction on the target bus. In this example, as soon as the bridge detects that the initiator has relinquished the LOCK# signal by sampling it in the de-asserted state while FRAME# is de-asserted, the bridge de-asserts the LOCK# signal on the target bus as soon as possible. Because of this behavior, LOCK# may not be de-asserted until several cycles after the last locked transaction has been completed on the target bus. As soon as the bridge has de-asserted LOCK# to indicate the end of a sequence of locked transactions, it resumes forwarding unlocked transactions.

When the last locked transaction is a posted write transaction, the bridge de-asserts LOCK# on the target bus at the end of the transaction because the lock was relinquished at the end of the write transaction on the initiator bus.

When the bridge receives a target abort or a master abort in response to a locked delayed transaction, the bridge returns a target abort or a master abort when the initiator repeats the locked transaction. The initiator must then de-assert LOCK# at the end of the transaction. The bridge sets the appropriate status bits, flagging the abnormal target termination condition (see Section 2.11). Normal forwarding of unlocked posted and delayed transactions is resumed.

When PI7C8154A receives a target abort or a master abort in response to a locked posted write transaction, PI7C8154A cannot pass back that status to the initiator. PI7C8154A asserts SERR# on the initiator bus when a target abort or a master abort is received during a locked posted write transaction, if the SERR# enable bit is set in the command register. Signal SERR# is asserted for the master abort condition if the master abort mode bit is set in the bridge control register (see Section 5.4).

7 PCI BUS ARBITRATION

The bridge must arbitrate for use of the primary bus when forwarding upstream transactions. Also, it must arbitrate for use of the secondary bus when forwarding downstream transactions. The arbiter for the primary bus resides external to the bridge, typically on the motherboard. For the

secondary PCI bus, the bridge implements an internal arbiter. This arbiter can be disabled, and an external arbiter can be used instead. This chapter describes primary and secondary bus arbitration.

7.1 PRIMARY PCI BUS ARBITRATION

The bridge implements a request output pin, P_REQ#, and a grant input pin, P_GNT#, for primary PCI bus arbitration. The bridge asserts P_REQ# when forwarding transactions upstream; that is, it acts as initiator on the primary PCI bus. As long as at least one pending transaction resides in the queues in the upstream direction, either posted write data or delayed transaction requests, the bridge keeps P_REQ# asserted. However, if a target retry, target disconnect, or a target abort is received in response to a transaction initiated by the bridge on the primary PCI bus, the bridge de-asserts P_REQ# for two PCI clock cycles.

For all cycles through the bridge, P_REQ# is not asserted until the transaction request has been completely queued. When P_GNT# is asserted LOW by the primary bus arbiter after the bridge has asserted P_REQ#, PI7C8154A initiates a transaction on the primary bus during the next PCI clock cycle. When P_GNT# is asserted to PI7C8154A when P_REQ# is not asserted, the bridge parks P_AD, P_CBE, and P_PAR by driving them to valid logic levels. When the primary bus is parked at the bridge and the bridge has a transaction to initiate on the primary bus, the bridge starts the transaction if P_GNT# was asserted during the previous cycle.

7.2 SECONDARY PCI BUS ARBITRATION

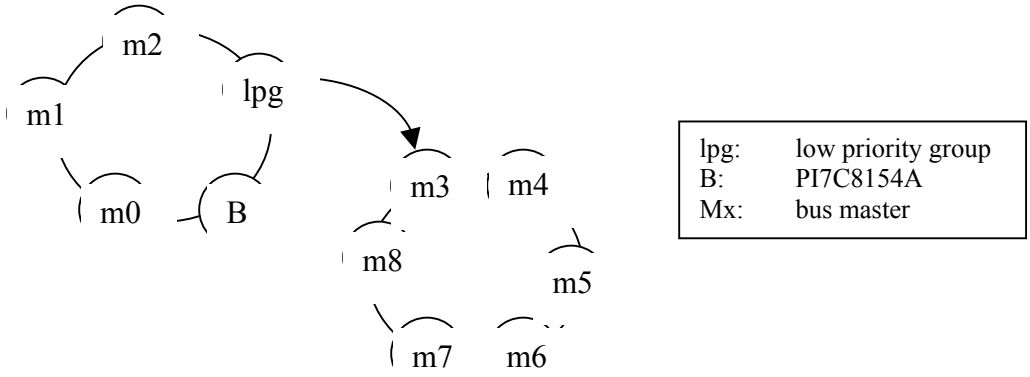
The bridge implements an internal secondary PCI bus arbiter. This arbiter supports eight external masters on the secondary bus in addition to PI7C8154A. The internal arbiter can be disabled, and an external arbiter can be used instead for secondary bus arbitration.

7.2.1 SECONDARY BUS ARBITRATION USING THE INTERNAL ARBITER

To use the internal arbiter, the secondary bus arbiter enable pin, S_CFN#, must be tied LOW. PI7C8154A has nine secondary bus request input pins, S_REQ#[8:0], and has nine secondary bus output grant pins, S_GNT#[8:0], to support external secondary bus masters.
The secondary bus request and grant signals are connected internally to the arbiter and are not brought out to external pins when S_CFN# is LOW.

The secondary arbiter supports a 2-sets programmable 2-level rotating algorithm with each set taking care of 4 requests / grants. Each set of masters can be assigned to a high priority group and a low priority group. The low priority group as a whole represents one entry in the high priority group; that is, if the high priority group consists of n masters, then in at least every n+1 transactions the highest priority is assigned to the low priority group. Priority rotates evenly among the low priority group. Therefore, members of the high priority group can be serviced n transactions out of n+1, while one member of the low priority group is serviced once every n+1 transactions. Figure 7-1 shows an example of an internal arbiter where four masters, including the bridge, are in the high priority group, and five masters are in the low priority group. Using this example, if all requests are always asserted, the highest priority rotates among the masters in the following fashion (high priority members are given in italics, low priority members, in boldface type): *B, m0, m1, m2,* **m3**, *B, m0, m1, m2,* **m4**, *B, m0, m1, m2,* **m5**, *B, m0, m1, m2,* **m6** and so on.

Figure 7-1 SECONDARY ARBITER EXAMPLE



Each bus master, including PI7C8154A, can be configured to be in either the low priority group or the high priority group by setting the corresponding priority bit in the arbiter-control register. The arbiter-control register is located at offset 40h. Each master has a corresponding bit. If the bit is set to 1, the master is assigned to the high priority group. If the bit is set to 0, the master is assigned to the low priority group. If all the masters are assigned to one group, the algorithm defaults to a straight rotating priority among all the masters. After reset, all external masters are assigned to the low priority group, and PI7C8154A is assigned to the high priority group. PI7C8154A receives highest priority on the target bus every other transaction and priority rotates evenly among the other masters.

Priorities are re-evaluated every time S_FRAME# is asserted at the start of each new transaction on the secondary PCI bus. From this point until the time that the next transaction starts, the arbiter asserts the grant signal corresponding to the highest priority request that is asserted. If a grant for a particular request is asserted, and a higher priority request subsequently asserts, the arbiter de-asserts the asserted grant signal and asserts the grant corresponding to the new higher priority request on the next PCI clock cycle. When priorities are re-evaluated, the highest priority is assigned to the next highest priority master relative to the master that initiated the previous transaction. The master that initiated the last transaction now has the lowest priority in its group.

If PI7C8154A detects that an initiator has failed to assert S_FRAME# after 16 cycles of both grant assertion and a secondary idle bus condition, the arbiter de-asserts the grant.

To prevent bus contention, if the secondary PCI bus is idle, the arbiter never asserts one grant signal in the same PCI cycle in which it de-asserts another. It de-asserts one grant and asserts the next grant, no earlier than one PCI clock cycle later. If the secondary PCI bus is busy, that is, S_FRAME# or S_IRDY# is asserted, the arbiter can be de-asserted one grant and asserted another grant during the same PCI clock cycle.

7.2.2 PREEMPTION

Preemption can be programmed to be either on or off, with the default to on (offset 4Ch, bit 31=0). Time-to-preempt can be programmed to 0, 1, 2, 4, 8, 16, 32, or 64 (default is 0) clocks. If the current master occupies the bus and other masters are waiting, the current master will be preempted by removing its grant (GNT#) after the next master waits for the time-to-preempt.

7.2.3 SECONDARY BUS ARBITRATION USING AN EXTERNAL ARBITER

The internal arbiter is disabled when the secondary bus central function control pin, S_CFN#, is tied HIGH. An external arbiter must then be used.

When S_CFN# is tied HIGH, PI7C8154A reconfigures two pins to be external request and grant pins. The S_GNT#[0] pin is reconfigured to be the external request pin because it's an output. The S_REQ#[0] pin is reconfigured to be the external grant pin because it's an input. When an external arbiter is used, PI7C8154A uses the S_GNT#[0] pin to request the secondary bus. When the reconfigured S_REQ#[0] pin is asserted LOW after PI7C8154A has asserted S_GNT#[0], PI7C8154A initiates a transaction on the secondary bus one cycle later. If grant is asserted and PI7C8154A has not asserted the request, PI7C8154A parks AD, CBE and PAR pins by driving them to valid logic levels.

The unused secondary bus grant outputs, S_GNT#[8:1] are driven HIGH. The unused secondary bus request inputs, S_REQ#[8:1], should be pulled HIGH.

7.2.4 BUS PARKING

Bus parking refers to driving the AD[31:0], CBE[3:0], and PAR lines to a known value while the bus is idle. In general, the device implementing the bus arbiter is responsible for parking the bus or assigning another device to park the bus. A device parks the bus when the bus is idle, its bus grant is asserted, and the device's request is not asserted. The AD[31:0] and CBE[3:0] signals should be driven first, with the PAR signal driven one cycle later. The AD[63:32] and CBE[7:4] are not driven and need to be pulled up to a valid logic level through external resistors.

PI7C8154A parks the primary bus only when P_GNT# is asserted, P_REQ# is de-asserted, and the primary PCI bus is idle. When P_GNT# is de-asserted, PI7C8154A 3-states the P_AD, P_CBE, and P_PAR signals on the next PCI clock cycle. If PI7C8154A is parking the primary PCI bus and wants to initiate a transaction on that bus, then PI7C8154A can start the transaction on the next PCI clock cycle by asserting P_FRAME# if P_GNT# is still asserted.

If the internal secondary bus arbiter is enabled, the secondary bus is always parked at the last master that used the PCI bus. That is, PI7C8154A keeps the secondary bus grant asserted to a particular master until a new secondary bus request comes along. After reset, PI7C8154A parks the secondary bus at itself until transactions start occurring on the secondary bus. Offset 48h, bit 1, can be set to 1 to park the secondary bus at PI7C8154A. By default, offset 48h, bit 1, is set to 0. If the internal arbiter is disabled, PI7C8154A parks the secondary bus only when the reconfigured grant signal, S_REQ#[0], is asserted and the secondary bus is idle.

8 GENERAL PURPOSE I/O INTERFACE

The PI7C8154A implements a 4-pin general purpose I/O interface. During normal operation, device specific configuration registers control the GPIO interface. The GPIO interface can be used for the following functions:

- During secondary interface reset, the GPIO interface can be used to shift in a 16-bit serial stream that serves as a secondary bus clock disable mask.
- Along with the GPIO[3] pin, a live insertion bit can be used to bring the PI7C8154A to a halt through hardware, permitting live insertion of option cards behind the PI7C8154A.

8.1 GPIO CONTROL REGISTERS

During normal operation, the following device specific configuration registers control the GPIO interface:

- The GPIO output data register
- The GPIO output enable control register
- The GPIO input data register

These registers consist of five 4-bit fields:

- Write-1-to-set output data field
- Write-1-to-clear output data field
- Write-1-to-set signal output enable control field
- Write-1-to-clear signal output enable control field
- Input data field

The bottom four bits of the output enable fields control whether each GPIO signal is input only or bi-directional. Each signal is controlled independently by a bit in each output enable control field. If a 1 is written to the write-1-to-set field, the corresponding pin is activated as an output. If a 1 is written to the write-1-to-clear field, the output driver is tri-stated, and the pin is then input only. Writing zeroes to these registers has no effect. The reset for these signals is input only.

The input data field is read only and reflects the current value of the GPIO pins. A type 0 configuration read operation to this address is used to obtain the values of these pins. All pins can be read at any time, whether configured as input only or as bi-directional.

The output data fields also use the write-1-to-set and write-1-to-clear mode. If a 1 is written to the write-1-to-set field and the pin is enabled as an output, the corresponding GPIO output is driven HIGH. If a 1 is written to the write-1-to-clear field and the pin is enabled as an output, the corresponding GPIO output is driven LOW. Writing zeros to these registers has no effect. The value written to the output register will be driven only when the GPIO signal is configured as bi-directional. A type 0 configuration write operation is used to program these fields. The rest value for the output is 0.

8.2 SECONDARY CLOCK CONTROL

The PI7C8154A uses the GPIO pins and the MSK_IN signal to input a 16-bit serial data stream. This data stream is shifted into the secondary clock control register and is used for selectively disabling secondary clock outputs.

The serial data stream is shifted in as soon as P_RESET# is detected deasserted and the secondary reset signal, S_RESET#, is detected asserted. The deassertion of S_RESET# is delayed until the PI7C8154A completes shifting in the clock mask data, which takes 23 clock cycles. After that, the GPIO pins can be used as general-purpose I/O pins.

An external shift register should be used to load and shift the data. The GPIO pins are used for shift register control and serial data input. Table 8-1 shows the operation of the GPIO pins.

Table 8-1 GPIO OPERATION

GPIO Pin	Operation
GPIO[0]	Shift register clock output at 33MHz max frequency
GPIO[1]	Not used
GPIO[2]	Shift register control 0: Load 1: Shift
GPIO[3]	Not used

The data is input through the dedicated input signal, MSK_IN.

The shift register circuitry is not necessary for correct operation of PI7C8154A. The shift register can be eliminated, and MSK_IN can be tied LOW to enable all secondary clock outputs or tied HIGH to force all secondary clock outputs HIGH. Table 8-2 shows the format of the serial stream.

Table 8-2 GPIO SERIAL DATA FORMAT

Bit	Description	S_CLKOUT
[1:0]	Slot 0 PRSNT#[1:0] or device 0	0
[3:2]	Slot 1 PRSNT#[1:0] or device 1	1
[5:4]	Slot 2 PRSNT#[1:0] or device 2	2
[7:6]	Slot 3 PRSNT#[1:0] or device 3	3
[8]	Device 4	4
[9]	Device 5	5
[10]	Device 6	6
[11]	Device 7	7
[12]	Device 8	8
[13]	PI7C8154A S_CLKIN	9
[14]	Reserved	NA
[15]	Reserved	NA

The first 8 bits contain the PRSNT#[1:0] signal values for four slots, and these bits control the S_CLKOUT[3:0] outputs. If one or both of the PRSNT#[1:0] signals are 0, that indicates that a card is present in the slot and therefore the secondary clock for that slot is not masked. If these clocks are connected to devices and not to slots, one or both of the bits should be tied low to enable the clock.

The next 5 bits are the clock mask for devices; each bit enables or disables the clock for one device. These bits control the S_CLKOUT[8:4] outputs: 0 enables the clock, and 1 disables the clock.

Bit 13 is the clock enable bit for S_CLKOUT[9], which is connected to PI7C8154A's S_CLKIN input.

If desired, the assignment of S_CLKOUT outputs to slots, devices, and PI7C8154A's S_CLKIN input can be rearranged from the assignment shown here. However, it is important that the serial data stream format match the assignment of S_CLKOUT.

The 8 least significant bits are connected to the PRSNT# pins for the slots. The next 5 bits are tied high to disable their respective secondary clocks because those clocks are not connected to anything. The next bit is tied LOW because that secondary clock output is connected to the bridge S_CLKIN input. When the secondary reset signal, S_RESET#, is detected asserted and the primary reset signal, P_RESET#, is detected deasserted, the bridge drives GPIO[2] LOW for one cycle to load the clock mask inputs into the shift register. On the next cycle, PI7C8154A drives GPIO[2] HIGH to perform a shift operation. This shifts the clock mask into MSK_IN; the most significant bit is shifted in first, and the least significant bit is shifted in last.

After the shift operation is complete, the bridge tri-states the GPIO signals and deasserts S_RESET#. PI7C8154A then ignores MSK_IN. Control of the GPIO signal now reverts to PI7C8154A GPIO control registers. The clock disable mask can be modified subsequently through a configuration write command to the secondary clock control register in device-specific configuration space.

8.3 LIVE INSERTION

The GPIO[3] pin can be used, along with a live insertion mode bit, to disable transaction forwarding.

To enable live insertion mode, the live insertion mode bit in the chip control register must be set to 1, and the output enable control for GPIO[3] must be set to input only in the GPIO output enable control register. When live insertion mode is enabled, whenever GPIO[3] is driven to a value of 1, the I/O enable, the memory enable, and the master enable bits are internally masked to 0. This means that, as a target, PI7C8154A no longer accepts any I/O or memory transactions, on either interface. When read, the register bits still reflect the value originally written by a configuration write command; when GPIO[3] is deasserted, the internal enable bits return to their original value (as they appear when read from the command register). When this mode is enabled, as a master, PI7C8154A completes any posted write or delayed request transactions that have already been queued.

Delayed completion transactions are not returned to the master in this mode because the bridge is not responding to any I/O or memory transactions during this time. PI7C8154A continues to accept Type 0 configuration transactions in live insertion mode. Once live insertion mode brings the bridge to a halt and queued transactions are completed, the secondary reset bit in the bridge control register can be used to assert S_RESET#, if desired, to reset and tri-state secondary bus devices, and to enable any live insertion hardware.

9 EEPROM INTERFACE

The EEPROM interface consists of three pins: EECLK (EEPROM clock output), EEPD (EEPROM bi-directional serial data), and EE_EN# (EEPROM enable on a LOW input). The bridge may control an ISSI IS24C02 or compatible part, which is organized into 256x8 bits. The EEPROM is used to initialize a select number of registers. This is accomplished after P_RESET# is deasserted, at which time the data from the EEPROM will be loaded. The EEPROM interface is organized into a 16-bit base, and the bridge supplies a 7-bit EEPROM word address. The bridge does not control the EEPROM address input. It can only access the EEPROM with address input set to 0.

9.1 AUTO MODE EEPROM ACCESS

The bridge may access the EEPROM in a WORD format by utilizing the auto mode through a hardware sequencer. The EEPROM start control, address, and read/write commands can be accessed through the configuration register. Before each access, the software should check the Start EEPROM bit before issuing the next start.

9.2 EEPROM MODE AT RESET

During a reset, the bridge will autoload information/data from the EEPROM if the automatic load condition is met. The first offset in the EEPROM contains a signature. If the signature is recognized, the autoload will initiate right after the reset.

During the autoload, the bridge will read sequential words from the EEPROM and write to the appropriate registers. Before the bridge registers can be accessed through the host, the autoload condition should be verified by reading bit[3] offset 54h (EEPROM Autoload Status). The host access is allowed only after the status of this bit becomes '1' which signifies that the autoload initialization sequence has completed successfully.

9.3 EEPROM DATA STRUCTURE

The bridge will access the EEPROM one WORD at a time. The bit order during the address phase is reverse that of the data phase. The data order starts with the MSB to the LSB during the address phase, but starts with the LSB to the MSB during the data phase.

9.4 EEPROM CONTENT

EEPROM BYTE ADDRESS	CONFIGURATION OFFSET	DESCRIPTION
00 – 01h		EEPROM SIGNATURE Autoload will only proceed if it reads a value of 1516h on the first word loaded.
02h		REGION ENABLE Enables or disables certain regions of the PCI configuration space from being loaded with contents in the EEPROM. bit[0]: reserved bit[4:1]: 0000 = stop autoload at offset 03h 0001 = stop autoload at offset 0Fh 0011 = stop autoload at offset 2Bh other combinations are undefined bit[7:5]: reserved
03h		ENABLE MISCELLANEOUS FUNCTIONS bit[0]: ISA enable control bit write protect – When this it is set, bridge will change bit[2] offset 3Eh into Read Only, and the ISA enable feature will not be available.
04 – 05h	00 – 01h	Vendor ID
06 – 07h	02 – 03h	Device ID
08h		Reserved
09h	09h	Class Code – low byte of Class Code register
0A – 0Bh	0A – 0Bh	Class Code – upper bytes of Class Code register
0Ch	0Eh	Header Type
0Dh	0Fh	BIST
0E – 0Fh		Reserved
10 – 11h	42 – 43h	Arbiter Control Register
12h	48h	Memory Read Flow/Underflow Control
13h	4Ah	Upstream Memory Base and Limit Enable
14h	4Fh	Arbiter Pre-emption Control (only bit[31:28])
15 – 16h	58 – 59h	Upstream Memory Base Register
17 – 18h	5A – 5Bh	Upstream Memory Limit Register
19 – 1Ch	5C – 5Fh	Upstream Memory Base Upper 32-bit Register
1D – 20h	60 – 63h	Upstream Memory Limit Upper 32-bit Register

EEPROM BYTE ADDRESS	CONFIGURATION OFFSET	DESCRIPTION
21 – 22h	74 – 75h	Port Option Register
23 – 24h	80 – 81h	Secondary Master Timeout Counter
25 – 26h	82 – 83h	Primary Master Timeout Counter
27 – 28h	DE – DFh	Power Management Capabilities
29 – 2Ah	E0 – E1h	Power Management Control Status Register
2Bh	E3h	Power Management Data
2C – 3Fh		Reserved – MUST BE SET TO 0

10 VITAL PRODUCT DATA (VPD)

The bridge contains the Vital Product Data registers as specified in the *PCI Local Bus Specification,* Revision 2.2. The bridge provides 192 bytes of storage in the EEPROM for the VPD data starting at offset ECh of the configuration space.

11 CLOCKS

This chapter provides information about the clocks.

11.1 PRIMARY AND SECONDARY CLOCK INPUTS

PI7C8154A implements a primary clock input for the PCI interface. The primary interface is synchronized to the primary clock input, P_CLK, and the secondary interface is synchronized to the secondary clock input. The secondary clock operates at either the same frequency as the primary clock or at half of the frequency of the primary clock. PI7C8154A operates at a maximum frequency of 66 MHz.

11.2 SECONDARY CLOCK OUTPUTS

The bridge has 10 secondary clock outputs, S_CLKOUT[9:0], that can be used as clock inputs for up to nine external secondary bus devices. The S_CLKOUT[9:0] outputs are derived from P_CLK. These are the rules for using secondary clocks:

- Each secondary clock output is limited to no more than one load
- One of the secondary clock outputs must be used to feedback to S_CLKIN

12 PCI POWER MANAGEMENT

PI7C8154A incorporates functionality that meets the requirements of the *PCI Power Management Specificatio*n, *Revision 1.0*. These features include:

- PCI Power Management registers using the Enhanced Capabilities Port (ECP) address mechanism
- Support for D0, $D3_{HOT}$ and $D3_{COLD}$ power management states

- Support for D0, D1, D2, $D3_{HOT}$, and $D3_{COLD}$ power management states for devices behind the bridge
- Support of the B2 secondary bus power state when in the $D3_{HOT}$ power management state

Table 12-1 shows the states and related actions that the bridge performs during power management transitions. (No other transactions are permitted.)

Table 12-1 POWER MANAGEMENT TRANSITIONS

Current Status	Next State	Action
D0	$D3_{COLD}$	Power has been removed from PI7C8154A. A power-up reset must be performed to bring PI7C8154A to D0.
D0	$D3_{HOT}$	If enabled to do so by the BPCCE pin, PI7C8154A will disable the secondary clocks and drive them LOW.
D0	D2	Unimplemented. PI7C8154A will ignore the write to the power state bits. Power state will remain at D0.
D0	D1	Unimplemented. PI7C8154A will ignore the write to the power state bits. Power state will remain at D0.
$D3_{HOT}$	D0	PI7C8154A enables secondary clock outputs and performs an internal chip reset. Signal S_RST# will not be asserted. All registers will be returned to the reset values and buffers will be cleared.
$D3_{COLD}$	$D3_{COLD}$	Power has been removed from PI7C8154A. A power-up reset must be performed to bring PI7C8154A to D0.
$D3_{COLD}$	D0	Power-up reset. PI7C8154A performs the standard power-up reset functions as described in Section 11.

PME# signals are routed from downstream devices around PCI-to-PCI bridges. PME# signals do not pass through PCI-to-PCI bridges.

13 RESET

This chapter describes the primary interface, secondary interface, and chip reset mechanisms.

13.1 PRIMARY INTERFACE RESET

PI7C8154A has a reset input, P_RESET#. When P_RESET# is asserted, the following events occur:

- PI7C8154A immediately tri-states all primary PCI interface signals. S_AD[31:0] and S_CBE[3:0] are driven LOW on the secondary interface and other control signals are tri-stated.
- PI7C8154A performs a chip reset.
- Registers that have default values are reset.
- PI7C8154A samples P_REQ64# to determine whether the 64-bit extension is enabled on the primary.

P_RESET# asserting and de-asserting edges can be asynchronous to P_CLK and S_CLKOUT. PI7C8154A is not accessible during P_RESET#. After P_RESET# is de-asserted, PI7C8154A remains inaccessible for 16 PCI clocks before the first configuration transaction can be accepted.

13.2 SECONDARY INTERFACE RESET

The bridge is responsible for driving the secondary bus reset signals, S_RESET#. Bridge asserts S_RESET# when any of the following conditions are met:

Signal P_RESET# is asserted. Signal S_RESET# remains asserted as long as P_RESET# is asserted and does not de-assert until P_RESET# is de-asserted.

The secondary reset bit in the bridge control register is set. Signal S_RESET# remains asserted until a configuration write operation clears the secondary reset bit.

The chip reset bit in the diagnostic control register is set. S_RESET# remains asserted until a configuration write operation clears the secondary reset bit. The S_RESET# in asserting and de-asserting edges can be asynchronous to P_CLK.

When S_RESET# is asserted, all secondary PCI interface control signals, including the secondary grant outputs, are immediately tri-stated. Signals S_AD[31:0], S_CBE[3:0], S_PAR are driven low for the duration of S_RESET# assertion. S_REQ64# is asserted LOW to indicate 64-bit extension support on the secondary. All posted write and delayed transaction data buffers are reset. Therefore, any transactions residing inside the buffers at the time of secondary reset are discarded.

When S_RESET# is asserted by means of the secondary reset bit, PI7C8154A remains accessible during secondary interface reset and continues to respond to accesses to its configuration space from the primary interface.

13.3 CHIP RESET

The chip reset bit in the diagnostic control register can be used to reset the PI7C8154A and the secondary bus.

When the chip reset bit is set, all registers and chip state are reset and all signals are tri-stated. S_RESET# is asserted and the secondary reset bit is automatically set. S_RESET# remains asserted until a configuration write operation clears the secondary reset bit. Within 20 PCI clock cycles after completion of the configuration write operation, PI7C8154A's reset bit automatically clears and PI7C8154A is ready for configuration.

During reset, PI7C8154A is inaccessible.

14 CONFIGURATION REGISTERS

PCI configuration defines a 64 DWORD space to define various attributes of PI7C8154A as shown below.

Table 14-1 CONFIGURATION SPACE MAP

31-24	23-16	15-8	7-0	DWORD Address
Device ID		Vendor ID		00h
Primary Status		Command		04h
Class Code			Revision ID	08h
Reserved	Header Type	Primary Latency Timer	Cache Line Size	0Ch
Reserved				10h
Reserved				14h
Secondary Latency Timer	Subordinate Bus Number	Secondary Bus Number	Primary Bus Number	18h
Secondary Status		I/O Limit Address	I/O Base Address	1Ch
Memory Limit Address		Memory Base Address		20h
Prefetchable Memory Limit Address		Prefetchable Memory Base Address		24h
Prefetchable Memory Base Address Upper 32-bit				28h
Prefetchable Memory Limit Address Upper 32-bit				2Ch
I/O Limit Address Upper 16-bit		I/O Base Address Upper 16-bit		30h
Reserved			Capability Pointer	34h
Reserved				38h
Bridge Control		Interrupt Pin (not supported)	Interrupt Line (not supported)	3Ch
Arbiter Control		Diagnostic / Chip Control		40h
Reserved				44h
Upstream Memory Control		Extended Chip Control		48h
Secondary Bus Arbiter Preemption Control	Hot Swap Switch Time Slot			4Ch
EEPROM Autoload Control/Status		Reserved		50h
EEPROM Data		EEPROM Address/Control		54h
Upstream (S to P) Memory Limit		Upstream (S to P) Memory Base		58h
Upstream (S to P) Memory Base Upper 32-bit				5Ch
Upstream (S to P) Memory Limit Upper 32-bit				60h
GPIO Data and Control			P_SERR# Event Disable	64h
Reserved	P_SERR# Status	Secondary Clock Control		68h
Reserved				6Ch - 70h
Reserved		Port Option		74h
Reserved				78h
Reserved				7Ch
Primary Master Timeout Counter		Secondary Master Timeout Counter		80h
Reserved				84h – ACh
Chassis Number	Slot Number	Next Pointer	Capability ID	B0h
Reserved				B4h – BFh
				C0h - CFh
Reserved				D0h –D8h
Power Management Capabilities		Next Item Pointer	Capability ID	DCh
Data	PPB Support Extensions	Power Management Data		E0h
Hot Swap Control and Status		Next Item Pointer	Capability ID	E4h
VPD		Next Item Pointer	Capability ID	E8h
VPD Data				ECh

14.1.1 SIGNAL TYPES

Signal Type	Description
R/O	Read Only
R/W	Read / Write
R/WC	Read / Write 1 to Clear
R/WR	Read / Write 1 to Reset (about 20 clocks)
R/WS	Read / Write 1 to Set

14.1.2 VENDOR ID REGISTER – OFFSET 00h

Bit	Function	Type	Description
15:0	Vendor ID	R/O	Identifies Pericom as vendor of this device. Hardwired as 12D8h.

14.1.3 DEVICE ID REGISTER – OFFSET 00h

Bit	Function	Type	Description
31:16	Device ID	R/O	Identifies this device as the PI7C8154A. Hardwired as 8154h.

14.1.4 COMMAND REGISTER – OFFSET 04h

Bit	Function	Type	Description
0	I/O Space Enable	R/W	Controls response to I/O access on the primary interface 0: ignore I/O transactions on the primary interface 1: enable response to I/O transactions on the primary interface Reset to 0
1	Memory Space Enable	R/W	Controls response to memory accesses on the primary interface 0: ignore memory transactions on the primary interface 1: enable response to memory transactions on the primary interface Reset to 0
2	Bus Master Enable	R/W	Controls ability to operate as a bus master on the primary interface 0: do not initiate memory or I/O transactions on the primary interface and disable response to memory and I/O transactions on the secondary interface 1: enables bridge to operate as a master on the primary interfaces for memory and I/O transactions forwarded from the secondary interface Reset to 0
3	Special Cycle Enable	R/O	No special cycles defined. Bit is defined as read only and returns 0 when read
4	Memory Write And Invalidate Enable	R/O	Bridge does not generate Memory Write and Invalidate except forwarding a transaction for another master. Bit is implemented as read only and returns 0 when read.

Bit	Function	Type	Description
5	VGA Palette Snoop Enable	R/W	Controls response to VGA compatible palette accesses 0: ignore VGA palette accesses on the primary 1: enable positive decoding response to VGA palette writes on the primary interface with I/O address bits AD[9:0] equal to 3C6h, 3C8h, and 3C9h (inclusive of ISA alias; AD[15:10] are not decoded and may be any value)
6	Parity Error Response	R/W	Controls response to parity errors 0: Bridge may ignore any parity errors that it detects and continue normal operation 1: Bridge must take its normal action when a parity error is detected Reset to 0
7	Wait Cycle Control	R/O	Controls the ability to perform address / data stepping 0: disable address/data stepping (affects primary and secondary) Reset to 0
8	P_SERR# enable	R/W	Controls the enable for the P_SERR# pin 0: disable the P_SERR# driver 1: enable the P_SERR# driver Reset to 0
9	Fast Back-to-Back Enable	R/W	Controls bridge's ability to generate fast back-to-back transactions to different devices on the primary interface. 0: no fast back-to-back transactions 1: enable fast back-to-back transactions Reset to 0
15:10	Reserved	R/O	Returns 000000 when read

14.1.5 STATUS REGISTER – OFFEST 04h

Bit	Function	Type	Description
19:16	Reserved	R/O	Reset to 0
20	Capabilities List	R/O	Set to 1 to enable support for the capability list (offset 34h is the pointer to the data structure) Reset to 1
21	66MHz Capable	R/O	Set to 1 to enable 66MHz operation on the primary interface Reset to 1
22	Reserved	R/O	Reset to 0
23	Fast Back-to-Back Capable	R/O	Set to 1 to indicate bridge is capable of decoding fast back-to-back transactions on the primary interface to different targets Reset to 1
24	Data Parity Error Detected	R/WC	0: No parity error detected on the primary interface (bridge is the primary bus master) 1: Parity error detected on the primary interface (bridge is the primary bus master) Reset to 0
26:25	DEVSEL# timing	R/O	DEVSEL# timing (medium decoding) 01: medium DEVSEL# decoding Reset to 01

Bit	Function	Type	Description
27	Signaled Target Abort	R/WC	0: Bridge does not signal target abort on the primary interface 1: Bridge signals target abort on the primary interface Reset to 0
28	Received Target Abort	R/WC	0: Bridge does not detect target abort on the primary interface 1: Bridge detects target abort on the primary interface Reset to 0
29	Received Master Abort	R/WC	0: Bridge does not detect master abort on the primary interface 1: Bridge detects master abort on the primary interface Reset to 0
30	Signaled System Error	R/WC	0: Bridge does not assert SERR# on the primary interface 1: Bridge asserts SERR# on the primary interface Reset to 0
31	Detected Parity Error	R/WC	0: Address of data parity error not detected by the bridge on the primary interface 1: Address of data parity error detected by the bridge on the primary interface Reset to 0

14.1.6 REVISION ID REGISTER – OFFSET 08h

Bit	Function	Type	Description
7:0	Revision	R/O	Indicates revision number of device. Hardwired to 02h

14.1.7 CLASS CODE REGISTER – OFFSET 08h

Bit	Function	Type	Description
15:8	Programming Interface	R/O	Read as 0 to indicate no programming interfaces have been defined for PCI-to-PCI bridges
23:16	Sub-Class Code	R/O	Read as 04h to indicate device is PCI-to-PCI bridge
31:24	Base Class Code	R/O	Read as 06h to indicate device is a bridge device

14.1.8 CACHE LINE SIZE REGISTER – OFFSET 0Ch

Bit	Function	Type	Description
7:0	Cache Line Size	R/W	Designates the cache line size for the system and is used when terminating memory write and invalidate transactions and when prefetching memory read transactions. Only cache line sizes (in units of 4-byte) which are a power of two are valid (only one bit can be set in this register; only 00h, 01h, 02h, 04h, 08h, and 10h are valid values). Reset to 0

14.1.9 PRIMARY LATENCY TIMER REGISTER – OFFSET 0Ch

Bit	Function	Type	Description
15:8	Primary Latency timer	R/W	This register sets the value for the Master Latency Timer, which starts counting when the master asserts FRAME#. Reset to 0

14.1.10 HEADER TYPE REGISTER – OFFSET 0Ch

Bit	Function	Type	Description
23:16	Header Type	R/O	Read as 01h to indicate that the register layout conforms to the standard PCI-to-PCI bridge layout.

14.1.11 PRIMARY BUS NUMBER REGISTER – OFFSET 18h

Bit	Function	Type	Description
7:0	Primary Bus Number	R/W	Indicates the number of the PCI bus to which the primary interface is connected. The value is set in software during configuration. Reset to 0

14.1.12 SECONDARY BUS NUMBER REGISTER – OFFSET 18h

Bit	Function	Type	Description
15:8	Secondary Bus Number	R/W	Indicates the number of the PCI bus to which the secondary interface is connected. The value is set in software during configuration. Reset to 0

14.1.13 SUBORDINATE BUS NUMBER REGISTER – OFFSET 18h

Bit	Function	Type	Description
23:16	Subordinate Bus Number	R/W	Indicates the number of the PCI bus with the highest number that is subordinate to the bridge. The value is set in software during configuration. Reset to 0

14.1.14 SECONDARY LATENCY TIMER – OFFSET 18h

Bit	Function	Type	Description
31:24	Secondary Latency Timer	R/W	Designated in units of PCI bus clocks. Latency timer checks for master accesses on the secondary bus interfaces that remain unclaimed by any target. Reset to 0

14.1.15 I/O BASE REGISTER – OFFSET 1Ch

Bit	Function	Type	Description
1:0	32-bit Indicator	R/O	Read as 01h to indicate 32-bit I/O addressing
3:2	Reserved	R/O	Returns 00 when read. Reset to 00.

Bit	Function	Type	Description
7:4	I/O Base Address [15:12]	R/W	Defines the bottom address of the I/O address range for the bridge to determine when to forward I/O transactions from one interface to the other. The upper 4 bits correspond to address bits [15:12] and are writable. The lower 12 bits corresponding to address bits [11:0] are assumed to be 0. The upper 16 bits corresponding to address bits [31:16] are defined in the I/O base address upper 16 bits address register Reset to 0

14.1.16 I/O LIMIT REGISTER – OFFSET 1Ch

Bit	Function	Type	Description
9:8	32-bit Indicator	R/O	Read as 01h to indicate 32-bit I/O addressing
11:10	Reserved	R/O	Returns 00 when read. Reset to 00
15:12	I/O Limit Address [15:12]	R/W	Defines the top address of the I/O address range for the bridge to determine when to forward I/O transactions from one interface to the other. The upper 4 bits correspond to address bits [15:12] and are writable. The lower 12 bits corresponding to address bits [11:0] are assumed to be FFFh. The upper 16 bits corresponding to address bits [31:16] are defined in the I/O limit address upper 16 bits address register Reset to 0

14.1.17 SECONDARY STATUS REGISTER – OFFSET 1Ch

Bit	Function	Type	Description
20:16	Reserved	R/O	Reset to 0
21	66MHz Capable	R/O	Set to 1 to enable 66MHz operation on the secondary interface Reset to 1
22	Reserved	R/O	Reset to 0
23	Fast Back-to-Back Capable	R/O	Set to 1 to indicate bridge is capable of decoding fast back-to-back transactions on the secondary interface to different targets Reset to 1
24	Data Parity Error Detected	R/WC	Set to 1 when S_PERR# is asserted and bit 6 of command register is set Reset to 0
26:25	DEVSEL# timing	R/O	DEVSEL# timing (medium decoding) 01: medium DEVSEL# decoding Reset to 01
27	Signaled Target Abort	R/WC	Set to 1 (by a target device) whenever a target abort cycle occurs on its secondary interface Reset to 0
28	Received Target Abort	R/WC	Set to 1 (by a master device) whenever transactions on its secondary interface are terminated with target abort Reset to 0
29	Received Master Abort	R/WC	Set to 1 (by a master) when transactions on its secondary interface are terminated with Master Abort Reset to 0
30	Received System Error	R/WC	Set to 1 when S_SERR# is asserted Reset to 0

Bit	Function	Type	Description
31	Detected Parity Error	R/WC	Set to 1 when address or data parity error is detected on the secondary interface Reset to 0

14.1.18 MEMORY BASE REGISTER – OFFSET 20h

Bit	Function	Type	Description
3:0	Reserved	R/O	Lower four bits of register are read only and return 0. Reset to 0
15:4	Memory Base Address [15:4]	R/W	Defines the bottom address of an address range for the bridge to determine when to forward memory transactions from one interface to the other. The upper 12 bits correspond to address bits [31:20] and are writable. The lower 20 bits corresponding to address bits [19:0] are assumed to be 0. Reset to 0

14.1.19 MEMORY LIMIT REGISTER – OFFSET 20h

Bit	Function	Type	Description
19:16	Reserved	R/O	Lower four bits of register are read only and return 0. Reset to 0
31:20	Memory Limit Address [31:20]	R/W	Defines the top address of an address range for the bridge to determine when to forward memory transactions from one interface to the other. The upper 12 bits correspond to address bits [31:20] and are writable. The lower 20 bits corresponding to address bits [19:0] are assumed to be FFFFFh.

14.1.20 PREFETCHABLE MEMORY BASE ADDRESS REGISTER – OFFSET 24h

Bit	Function	Type	Description
3:0	64-bit addressing	R/O	Indicates 64-bit addressing 0000: 32-bit addressing 0001: 64-bit addressing Reset to 0001
15:4	Prefetchable Memory Base Address [31:20]	R/W	Defines the bottom address of an address range for the bridge to determine when to forward memory read and write transactions from one interface to the other. The upper 12 bits correspond to address bits [31:20] and are writable. The lower 20 bits are assumed to be 0.

14.1.21 PREFETCHABLE MEMORY LIMIT ADDRESS REGISTER – OFFSET 24h

Bit	Function	Type	Description
19:16	64-bit addressing	R/O	Indicates 64-bit addressing 0000: 32-bit addressing 0001: 64-bit addressing Reset to 1
31:20	Prefetchable Memory Limit Address [31:20]	R/W	Defines the top address of an address range for the bridge to determine when to forward memory read and write transactions from one interface to the other. The upper 12 bits correspond to address bits [31:20] and are writable. The lower 20 bits are assumed to be FFFFFh.

14.1.22 PREFETCHABLE MEMORY BASE ADDRESS UPPER 32-BITS REGISTER – OFFSET 28h

Bit	Function	Type	Description
31:0	Prefetchable Memory Base Address, Upper 32-bits [63:32]	R/W	Defines the upper 32-bits of a 64-bit bottom address of an address range for the bridge to determine when to forward memory read and write transactions from one interface to the other. Reset to 0

14.1.23 PREFETCHABLE MEMORY LIMIT ADDRESS UPPER 32-BITS REGISTER – OFFSET 2Ch

Bit	Function	Type	Description
31:0	Prefetchable Memory Limit Address, Upper 32-bits [63:32]	R/W	Defines the upper 32-bits of a 64-bit top address of an address range for the bridge to determine when to forward memory read and write transactions from one interface to the other. Reset to 0

14.1.24 I/O BASE ADDRESS UPPER 16-BITS REGISTER – OFFSET 30h

Bit	Function	Type	Description
15:0	I/O Base Address, Upper 16-bits [31:16]	R/W	Defines the upper 16-bits of a 32-bit bottom address of an address range for the bridge to determine when to forward I/O transactions from one interface to the other. Reset to 0

14.1.25 I/O LIMIT ADDRESS UPPER 16-BITS REGISTER – OFFSET 30h

Bit	Function	Type	Description
31:16	I/O Limit Address, Upper 16-bits [31:16]	R/W	Defines the upper 16-bits of a 32-bit top address of an address range for the bridge to determine when to forward I/O transactions from one interface to the other. Reset to 0

14.1.26 CAPABILITY POINTER REGISTER – OFFSET 34h

Bit	Function	Type	Description
7:0	Enhanced Capabilities Port Pointer	R/O	Enhanced capabilities port offset pointer. Read as DCh to indicate that the first item resides at that configuration offset. Reset to DCh.

14.1.27 INTERRUPT LINE REGISTER – OFFSET 3Ch

Bit	Function	Type	Description
7:0	Interrupt Line	R/W	For POST to program to FFh, indicating that the bridge does not implement an interrupt pin. Reset to 0.

14.1.28 INTERRUPT PIN REGISTER – OFFSET 3Ch

Bit	Function	Type	Description
15:8	Interrupt Pin	R/O	Interrupt pin not supported on the bridge. Reset to 0.

14.1.29 BRIDGE CONTROL REGISTER – OFFSET 3Ch

Bit	Function	Type	Description
16	Parity Error Response	R/W	Controls the bridge's response to parity errors on the secondary interface. 0: ignore address and data parity errors on the secondary interface 1: enable parity error reporting and detection on the secondary interface Reset to 0
17	S_SERR# enable	R/W	Controls the forwarding of S_SERR# to the primary interface. 0: disable the forwarding of S_SERR# to primary interface 1: enable the forwarding of S_SERR# to primary interface Reset to 0
18	ISA enable	R/W	Modifies the bridge's response to ISA I/O addresses, applying only to those addresses falling within the I/O base and limit address registers and within the first 64KB of PCI I/O space. 0: forward all I/O addresses in the range defined by the I/O base and I/O limit registers 1: blocks forwarding of ISA I/O addresses in the range defined by the I/O base and I/O limit registers that are in the first 64KB of I/O space that address the last 768 bytes in each 1KB block. Secondary I/O transactions are forwarded upstream if the address falls within the last 768 bytes in each 1KB block Reset to 0

Bit	Function	Type	Description
19	VGA enable	R/W	Controls the bridge's response to VGA compatible addresses. 0: does not forward VGA compatible memory and I/O addresses from primary to secondary 1: forward VGA compatible memory and I/O addresses from primary to secondary regardless of other settings Reset to 0
20	Reserved	R/O	Reserved. Returns 0 when read. Reset to 0
21	Master Abort Mode	R/W	Control's bridge's behavior responding to master aborts on secondary interface. 0: does not report master aborts (returns FFFF_FFFFh on reads and discards data on writes) 1: reports master aborts by signaling target abort if possible or by the assertion of P_SERR# if enabled Reset to 0
22	Secondary Interface Reset	R/W	Controls the assertion of S_RESET# signal pin on the secondary interface 0: does not force the assertion of S_RESET# pin 1: forces the assertion of S_RESET# Reset to 0
23	Fast Back-to-Back Enable	R/W	Controls bridge's ability to generate fast back-to-back transactions on the secondary interface. 0: does not allow fast back-to-back transactions on the secondary 1: enables fast back-to-back transactions on the secondary Reset to 0
24	Primary Master Timeout	R/W	Determines the maximum number of PCI clock cycles the bridge waits for an initiator on the primary interface to repeat a delayed transaction request. 0: Primary discard timer counts 2^{15} PCI clock cycles. 1: Primary discard timer counts 2^{10} PCI clock cycles. Reset to 0
25	Secondary Master Timeout	R/W	Determines the maximum number of PCI clock cycles the bridge waits for an initiator on the primary interface to repeat a delayed transaction request. 0: Primary discard timer counts 2^{15} PCI clock cycles. 1: Primary discard timer counts 2^{10} PCI clock cycles. Reset to 0
26	Master Timeout Status	R/WC	This bit is set to 1 when either the primary master timeout counter or secondary master timeout counter expires. Reset to 0
27	Discard Timer P_SERR# enable	R/W	This bit is set to 1 and P_SERR# is asserted when either the primary discard timer or the secondary discard timer expire. 0: P_SERR# is not asserted on the primary interface as a result of the expiration of either the Primary Discard Timer or the Secondary Discard Timer. 1: P_SERR# is asserted on the primary interface as a result of the expiration of either the Primary Discard Timer or the Secondary Discard Timer. Reset to 0

Bit	Function	Type	Description
31-28	Reserved	R/O	Reserved. Returns 0 when read. Reset to 0.

14.1.30 DIAGNOSTIC / CHIP CONTROL REGISTER – OFFSET 40h

Bit	Function	Type	Description
0	Reserved	R/O	Reserved. Returns 0 when read. Reset to 0
1	Memory Write Disconnect Control	R/W	Controls when the bridge (as a target) disconnects memory write transactions. 0: memory write disconnects at 4KB aligned address boundary 1: memory write disconnects at cache line aligned address boundary Reset to 0
3:2	Reserved	R/O	Reserved. Returns 0 when read. Reset to 0.
4	Secondary Bus Prefetch Disable	R/W	Controls the bridge's ability to prefetch during upstream memory read transactions 0: Bridge prefetches and does not forward byte enable bits during upstream memory read transactions. 1: Bridge requests only 1 DWORD from the target and forwards read byte enable bits during upstream memory reads. Reset to 0
5	Live Insertion Mode	R/W	Enables control of transaction forwarding 0: GPIO[3] has no effect on the I/O, memory, and master enable bits 1: If GPIO[3] is set to input only, this bit enables GPIO[3] to mask the I/O enable, memory enable, and master enable bits to 0. These bits are masked when GPIO[3] is driven HIGH. As a result, PI7C8154 stops accepting I/O and memory transactions. Reset to 0
7:6	Reserved	R/O	Reserved. Returns 0 when read. Reset to 0
8	Chip Reset	R/WR	Controls the chip and secondary bus reset. 0: Bridge is ready for operation 1: Causes Bridge to perform a chip reset
15:9	Reserved	R/O	Reserved. Returns 0 when read. Reset to 0.

14.1.31 ARBITER CONTROL REGISTER – OFFSET 40h

Bit	Function	Type	Description
24:16	Arbiter Control	R/W	Each bit controls whether a secondary bus master is assigned to the high priority group or the low priority group. Bits [24:16] correspond to request inputs S_REQ[8:0] 0: low priority 1: high priority Reset to 0
25	Priority of Secondary Interface	R/W	Controls whether the secondary interface of the bridge is in the high priority group or the low priority group. 0: low priority 1: high priority Reset to 1

Bit	Function	Type	Description
26	Broken Master Timeout Enable	R/W	0: **Broken master timeout off**. If a master receives its GNT# active but does not initiate any transactions for more than 16 clocks, the arbiter will consider the master as broken for only two clocks. The current GNT# will be de-asserted if another master asserts its REQ# or automatic preemption is on (bit[27] offset 40h); otherwise the current GNT# will be kept asserted. 1: **Broken master timeout on**. If a master receives its GNT# active but does not initiate any transactions for more than 16 clocks, the arbiter will consider the master as broken and the REQ# of the current master will be ignored for arbitration until de-assertion of its REQ#. The current GNT# will be de-asserted if another master asserts its REQ# or automatic preemption is on (bit[27] offset 40h); otherwise the current GNT# will be kept asserted. Reset to 0
27	Automatic Preemption Control	R/W	0: **Automatic preemption off**. If the preemption timer expires (bit[31:28] offset 4Ch) and another master asserts REQ#, the GNT# of the current master will be de-asserted and the GNT# of the next master will be asserted. If no other master asserts REQ#, the current GNT# will remain asserted. 1: **Automatic preemption on**. If the preemption timer expires (bit[31:28] offset 4Ch), the GNT# to the current master will be de-asserted for one clock. The same GNT# will be asserted again if no other master asserts its REQ#. If another master asserts its REQ#, the arbiter will generate a GNT# for the next master with the highest priority. Reset to 0
31:28	Reserved	R/O	Returns 0000 when read. Reset to 0000.

14.1.32 EXTENDED CHIP CONTROL REGISTER – OFFSET 48h

Bit	Function	Type	Description
0	Memory Read Flow Through Disable	R/W	Controls ability to do memory read flow through 0: Enable flow through during a memory read transaction 1: Disables flow through during a memory read transaction Reset to 0
1	Park	R/W	Controls bus arbiter's park function 0: Park to last master 1: Park to the bridge Reset to 0
2	Downstream (P to S) Memory Read Dynamic Prefetching	R/W	0: Enable downstream memory read prefetching dynamic control 1: Disable downstream memory read prefetching dynamic control Reset to 0
3	Upstream (S to P) Memory Read Dynamic Prefetching	R/W	0: Enable upstream memory read prefetching dynamic control 1: Disable upstream memory read prefetching dynamic control Reset to 0

Bit	Function	Type	Description
4	Memory Read Underflow Control	R/W	0: Bridge will start returning memory read data to the source bus after the 2nd data is in the data buffer. If the data buffer is read as empty (underflow), bridge will insert target wait states (up to 7 wait states) on the source bus and prefetch more data in the data buffer. If there is no further data coming into the data buffer and the number of wait states reaches 7, the bridge will assert STOP# to disconnect the master and terminate the transaction. 1: Bridge will not start returning memory read data to the source bus until 1 cache line of data is accumulated in the data buffer. If the data buffer is read as empty (underflow), the bridge will stop prefetching at the destination bus and signal a disconnect to the external master on the source bus. The transaction entry and the associated data will be discarded. Reset to 0
15:5	Reserved	R/O	Returns 0 when read. Reset to 0.

14.1.33 UPSTREAM MEMORY CONTROL REGISTER – OFFSET 48h

Bit	Function	Type	Description
16	Upstream (S to P) Memory Base and Limit Enable	R/W	0: Upstream memory range is the entire range except the downstream memory channel 1: Upstream memory range is confined to the upstream Memory Base and Limit *see Offset 58h, 5Ch, and 60h for upstream memory range
31:17	Reserved	R/O	Returns 0 when read. Reset to 0.

14.1.34 SECONDARY BUS ARBITER PREEMPTION CONTROL REGISTER – OFFSET 4Ch

Bit	Function	Type	Description
31:28	Secondary bus arbiter preemption contorl	R/W	Controls the number of clock cycles after frame is asserted before preemption is enabled. 1xxx: Preemption off 0000: Preemption enabled after 0 clock cycles after FRAME asserted 0001: Preemption enabled after 1 clock cycle after FRAME asserted 0010: Preemption enabled after 2 clock cycles after FRAME asserted 0011: Preemption enabled after 4 clock cycles after FRAME asserted 0100: Preemption enabled after 8 clock cycles after FRAME asserted 0101: Preemption enabled after 16 clock cycles after FRAME asserted 0110: Preemption enabled after 32 clock cycles after FRAME asserted 0111: Preemption enabled after 64 clock cycles after FRAME asserted

14.1.35 HOT SWAP SWITCH TIME SLOT REGISTER – OFFSET 4Ch

Bit	Function	Type	Description
27:0	Hot Swap Switch Time Slot Register	R/W	Hot Swap switch time slot set to 0003A98h (15K PCI clocks). Reset to 0003A98h.

14.1.36 EEPROM AUTOLOAD CONTROL / STATUS REGISTER – OFFSET 50h

Bit	Function	Type	Description
15:0	Reserved	R/O	Returns 0 when read. Reset to 0.
16	EEPROM Autoload Control	R/W	0: Enable EEPROM autoload 1: Disable EEPROM autoload Reset to 0
17	Fast EEPROM Autoload Control	R/W	0: Normal speed of EEPROM autoload 1: Speeds up EEPROM autoload by 32X Reset to 0
18	EEPROM Autoload Status	R/O	0: EEPROM autoload is not ongoing 1: EEPROM autoload is on going Reset to 0
31:19	Reserved	R/O	Returns 0 when read. Reset to 0.

14.1.37 EEPROM ADDRESS / CONTROL REGISTER – OFFSET 54h

Bit	Function	Type	Description
0	EEPROM Read or Write Cycle Start	R/W	
1	Command for EEPROM	R/W	Controls the command sent to the EEPROM 0: Read 1: Write Reset to 0
2	EEPROM Error	R/O	0: EEPROM acknowledge is always received during the EEPROM cycle. 1: EEPROM acknowledge is not received during the EEPROM cycle. Reset to 0
3	EEPROM Autoload Complete Status	R/O	0: EEPROM autoload is not successfully completed 1: EEPROM autoload is successfully completed. Reset to 0
5:4	Reserved	R/O	Returns 0 when read. Reset to 0
7:6	EEPROM Clock Frequency	R/W	00: Primary clock / 1024 01: Primary clock / 512 10: Primary clock / 256 11: Primary clock / 32 Reset to 0
8	Reserved	R/O	Returns 0 when read. Reset to 0.
15:9	EEPROM Word Address	R/W	Stores the EEPROM word address for the EEPROM cycle. Reset to 0

14.1.38 EEPROM DATA REGISTER – OFFSET 54h

Bit	Function	Type	Description
31:16	EEPROM Data	R/W	Stores the EEPROM data to be written into the EEPROM or receives the data from the EEPROM after an EEPROM read cycle is completed.

14.1.39 UPSTREAM (S TO P) MEMORY BASE ADDRESS REGISTER – OFFSET 58h

Bit	Function	Type	Description
3:0	64-bit Addressing	R/O	0000: 32-bit addressing 0001: 64-bit addressing Reset to 0
15:4	Upstream Memory Base	R/W	Defines the bottom address of an address range used by the bridge to determine when to forward upstream memory transactions. Reset to 0

14.1.40 UPSTREAM (S TO P) MEMORY LIMIT ADDRESS REGISTER – OFFSET 58h

Bit	Function	Type	Description
19:16	64-bit Addressing	R/O	0000: 32-bit addressing 0001: 64-bit addressing Reset to 0
31:20	Upstream Memory Limit	R/W	Defines the top address of an address range used by the bridge to determine when to forward upstream memory transactions. Reset to 0

14.1.41 UPSTREAM (S TO P) MEMORY BASE ADDRESS UPPER 32-BIT REGISTER – OFFSET 5Ch

Bit	Function	Type	Description
31:0	Upstream Memory Base Upper 32-bits	R/W	Defines the upper 32-bits of a 64-bit bottom address of an address range for the bridge to determine when to forward upstream memory read and write transactions. Reset to 0

14.1.42 UPSTREAM (S TO P) MEMORY LIMIT ADDRESS UPPER 32-BIT REGISTER – OFFSET 60h

Bit	Function	Type	Description
31:0	Upstream Memory Base Upper 32-bits	R/W	Defines the upper 32-bits of a 64-bit top address of an address range for the bridge to determine when to forward upstream memory read and write transactions. Reset to 0

14.1.43 P_SERR# EVENT DISABLE REGISTER – OFFSET 64h

Bit	Function	Type	Description
0	Reserved	R/O	Returns 0 when read. Reset to 0

Bit	Function	Type	Description
1	Posted Write with Parity Error	R/W	0: P_SERR# is asserted if a parity error is detected on the target bus during a posted write transaction and the SERR# enable bit in the command register is set. 1: P_SERR# is not asserted, although a parity error is detected on the target bus during a posted write transaction and the SERR# enable bit in the command register is set. Reset to 0
2	Posted Write with Non-Delivery Data	R/W	0: P_SERR# is asserted if the bridge is not able to transfer any posted write data after 2^{24} attempts and the SERR# enable bit in the command register is set. 1: P_SERR# is not asserted although the bridge is not able to transfer any posted write data after 2^{24} attempts and the SERR# enable bit in the command register is set. Reset to 0
3	Target Abort During Posted Write	R/W	0: P_SERR# is asserted if the bridge receives a target abort when attempting to deliver posted write data and the SERR# enable bit in the command register is set. 1: P_SERR# is not asserted even though the bridge receives a target abort when attempting to deliver posted write data and the SERR# enable bit in the command register is set. Reset to 0
4	Master Abort During Posted Write	R/W	0: P_SERR# is asserted if the bridge receives a master abort when attempting to deliver posted write data and the SERR# enable bit in the command register is set. 1: P_SERR# is not asserted even though the bridge receives a master abort when attempting to deliver posted write data and the SERR# enable bit in the command register is set. Reset to 0
5	Delayed Write with Non-Delivery	R/W	0: P_SERR# is asserted if the bridge is not able to transfer any delayed write data after 2^{24} attempts and the SERR# enable bit in the command register is set. 1: P_SERR# is not asserted even though the bridge is not able to transfer any delayed write data after 2^{24} attempts and the SERR# enable bit in the command register is set. Reset to 0
6	Delayed Read Without Data From Target	R/W	0: P_SERR# is asserted if the bridge is not able to transfer any read data from the target after 2^{24} attempts and the SERR# enable bit in the command register is set. 1: P_SERR# is not asserted even though the bridge is not able to transfer and read data from the target after 2^{24} attempts and the SERR# enable bit in the command register is set. Reset to 0.
7	Reserved	R/O	Returns 0 when read. Reset to 0.

14.1.44 GPIO DATA AND CONTROL REGISTER – OFFSET 64h

Bit	Function	Type	Description
11:8	GPIO output write-1-to-clear	R/WC	Setting any of these bits to 1 drives the corresponding bits LOW on the GPIO[3:0] bus if it is programmed as bi-directional. Data is driven on the PCI clock cycle following completion of the configuration write to this register. The bit positions corresponding to the GPIO pins that are programmed as input only are not driven. Writing 0 to theses bits has no effect and will return the last written value when read. Bits [11:8] correspond to GPIO [3:0]. Reset to 0
15:12	GPIO output write-1-to-set	R/WS	Setting any of these bits to 1 drives the corresponding bits HIGH on the GPIO[3:0] bus if it is programmed as bi-directional. Data is driven on the PCI clock cycle following completion of the configuration write to this register. The bit positions corresponding to the GPIO pins that are programmed as input only are not driven. Writing 0 to theses bits has no effect and will return the last written value when read. Bits [15:12] correspond to GPIO [3:0]. Reset to 0
19:16	GPIO output enable write-1-to-clear	R/WC	Setting any of these bits to 1 configures the corresponding bits on the GPIO[3:0] bus as input only. As a result, the output driver is tri-stated. Writing 0 to theses bits has no effect and will return the last written value when read. Bits [19:16] correspond to GPIO [3:0]. Reset to 0
23:20	GPIO output enable write-1-to-set	R/WS	Setting any of these bits to 1 configures the corresponding bits on the GPIO[3:0] bus as bi-directional; the output driver is enabled and drives the value set in the output data register (offset 65h). Writing 0 to theses bits has no effect and will return the last written value when read. Bits [23:20] correspond to GPIO [3:0]. Reset to 0
27:24	Reserved	R/O	Returns 0 when read. Reset to 0
31:28	GPIO Input Data Register	R/O	Contains the state of the GPIO[3:0] pins. State is updated on the PCI clock cycle after any change to the state of the GPIO[3:0] pins. Reset to 0.

14.1.45 SECONDARY CLOCK CONTROL REGISTER – OFFSET 68h

Bit	Function	Type	Description
1:0	S_CLKOUT[0] disable	R/W	S_CLKOUT[0] (slot 0) Enable 00: enable S_CLKOUT[0] 01: enable S_CLKOUT[0] 10: enable S_CLKOUT[0] 11: disable S_CLKOUT[0] and driven HIGH Reset to 00
3:2	S_CLKOUT[1] disable	R/W	S_CLKOUT[1] (slot 1) Enable 00: enable S_CLKOUT[1] 01: enable S_CLKOUT[1] 10: enable S_CLKOUT[1] 11: disable S_CLKOUT[1] and driven HIGH Reset to 00

Bit	Function	Type	Description
5:4	S_CLKOUT[2] disable	R/W	S_CLKOUT[2] (slot 2) Enable 00: enable S_CLKOUT[2] 01: enable S_CLKOUT[2] 10: enable S_CLKOUT[2] 11: disable S_CLKOUT[2] and driven HIGH Reset to 00
7:6	S_CLKOUT[3] disable	R/W	S_CLKOUT[3] (slot 3) Enable 00: enable S_CLKOUT[3] 01: enable S_CLKOUT[3] 10: enable S_CLKOUT[3] 11: disable S_CLKOUT[3] and driven HIGH Reset to 00
8	S_CLKOUT[4] disable	R/W	S_CLKOUT[4] (device 1) Enable 0: enable S_CLKOUT[4] 1: disable S_CLKOUT[4] and driven HIGH Reset to 0
9	S_CLKOUT[5] disable	R/W	S_CLKOUT[5] (device 2) Enable 0: enable S_CLKOUT[5] 1: disable S_CLKOUT[5] and driven HIGH Reset to 0
10	S_CLKOUT[6] disable	R/W	S_CLKOUT[6] (device 3) Enable 0: enable S_CLKOUT[6] 1: disable S_CLKOUT[6] and driven HIGH Reset to 0
11	S_CLKOUT[7] disable	R/W	S_CLKOUT[7] (device 4) Enable 0: enable S_CLKOUT[7] 1: disable S_CLKOUT[7] and driven HIGH Reset to 0
12	S_CLKOUT[8] disable	R/W	S_CLKOUT[8] (device 5) Enable 0: enable S_CLKOUT[8] 1: disable S_CLKOUT[8] and driven HIGH Reset to 0
13	S_CLKOUT[9] disable	R/W	S_CLKOUT[9] (Bridge) Enable 0: enable S_CLKOUT[4] 1: disable S_CLKOUT[4] and driven HIGH This bit is initialized upon secondary reset by shifting in a serial data stream. The bit is assigned to correspond to the Bridge secondary clock input (S_CLKIN). Reset to 0
15:14	Reserved	RO	Returns 11 when read. Reset to 11.

14.1.46 P_SERR# STATUS REGISTER – OFFSET 68h

Bit	Function	Type	Description
16	Address Parity Error	R/WC	1: Signal P_SERR# was asserted because an address parity error was detected on P or S bus. Reset to 0
17	Posted Write Data Parity Error	R/WC	1: Signal P_SERR# was asserted because a posted write data parity error was detected on the target bus. Reset to 0
18	Posted Write Non-delivery	R/WC	1: Signal P_SERR# was asserted because the bridge was unable to deliver post memory write data to the target after 2^{24} attempts. Reset to 0
19	Target Abort during Posted Write	R/WC	1: Signal P_SERR# was asserted because the bridge received a target abort when delivering post memory write data. Reset to 0.
20	Master Abort during Posted Write	R/WC	1: Signal P_SERR# was asserted because the bridge received a master abort when attempting to deliver post memory write data Reset to 0.
21	Delayed Write Non-delivery	R/WC	1: Signal P_SERR# was asserted because the bridge was unable to deliver delayed write data after 2^{24} attempts. Reset to 0
22	Delayed Read – No Data from Target	R/WC	1: Signal P_SERR# was asserted because the bridge was unable to read any data from the target after 2^{24} attempts. Reset to 0.
23	Delayed Transaction Master Timeout	R/WC	1: Signal P_SERR# was asserted because a master did not repeat a read or write transaction before master timeout. Reset to 0.
31:24	Reserved	R/O	Returns 0 when read. Reset to 0

14.1.47 PORT OPTION REGISTER – OFFSET 74h

Bit	Function	Type	Description
0	Reserved	R/O	Returns 0 when read. Reset to 0.
1	Primary Memory Read Command Alias Enable	R/W	0: exact matching for non-posted memory write retry cycles from initiator on the primary interface 1: alias MEMRL or MEMRM to MEMR for memory read retry cycles from the initiator on the primary interface Reset to 1
2	Primary Memory Write Command Alias Enable	R/W	Reserved Reset to 0
3	Secondary Memory Read Command Alias Enable	R/W	0: exact matching for memory read retry cycles from initiator on the secondary interface 1: alias MEMRL or MEMRM to MEMR for memory read retry cycles from initiator on the secondary interface Reset to 1

Bit	Function	Type	Description
4	Secondary Memory Write Command Alias Enable	R/W	0: exact matching for non-posted memory write retry cycles from initiator on the secondary interface 1: alias MEMWI to MEMW for non-posted memory write retry cycles from initiator on the secondary interface Reset to 0
5	Primary Memory Read Line/Multiple Alias Enable	R/W	0: Exact matching for memory read line/multiple retry cycles from initiator on the primary interface 1: alias MEMRL to MEMRM or MEMRM to MEMRL for memory read retry cycles from initiator on the primary interface Reset to 1
6	Secondary Memory Read Line/Multiple Alias Enable	R/W	0: Exact matching for memory read line/multiple retry cycles from initiator on the secondary interface 1: alias MEMRL to MEMRM or MEMRM to MEMRL for memory read retry cycles from initiator on the secondary interface Reset to 1
7	Primary Memory Write and Invalidate Command Alias Disable	R/W	0: When accepting MEMWI commands on primary, bridge converts MEMWI to MEMW on destination bus 1: When accepting MEMWI commands on primary, bridge does not convert MEMWI to MEMW on destination bus Reset to 0
8	Secondary Memory Write and Invalidate Command Alias Disable	R/W	0: When accepting MEMWI commands on secondary, bridge converts MEMWI to MEMW on destination bus 1: When accepting MEMWI commands on secondary, bridge does not convert MEMWI to MEMW on destination bus Reset to 0
9	Enable Long Request	R/W	0: normal lock operation 1: enable long request for lock cycle Reset to 0
10	Enable Secondary To Hold Request Longer	R/W	0: internal secondary master will release REQ# after FRAME# assertion 1: internal secondary master will hold REQ# until there is no transactions pending in FIFO or until terminated by target Reset to 1
11	Enable Primary To Hold Request Longer	R/W	0: internal Primary master will release REQ# after FRAME# assertion 1: internal Primary master will hold REQ# until there is no transactions pending in FIFO or until terminated by target Reset to 1
12	Ordering Rules Control 1	R/W	0: Enable the out of order capability between two DTR requests from two FIFO's 1: Disable the out of order capability between two DTR requests from two FIFO's Reset to 0

Bit	Function	Type	Description
13	Ordering Rules Control 2	R/W	0: Keep the ordering rule requirements between delay read completion and posted write transactions 1: Disregard the ordering rule requirements between delay read completion and posted write transactions Reset to 0
15:14	Reserved	R/W	Reset to 0

14.1.48 SECONDARY MASTER TIMEOUT COUNTER REGISTER – OFFSET 80h

Bit	Function	Type	Description
15:0	Secondary Master Timeout	R/W	Secondary timeout occurs after 2^{15} PCI clocks. Reset to 8000h.

14.1.49 PRIMARY MASTER TIMEOUT COUNTER REGISTER – OFFSET 80h

Bit	Function	Type	Description
31:16	Primary Master Timeout	R/W	Primary timeout occurs after 2^{15} PCI clocks. Reset to 8000h.

14.1.50 CAPABILITY ID REGISTER – OFFSET B0h

Bit	Function	Type	Description
7:0	Enhanced Capabilities ID	R/O	Read as 04h to indicate that these are Slot Indentification registers.

14.1.51 NEXT POINTER REGISTER – OFFSET B0h

Bit	Function	Type	Description
15:8	Next Item Pointer	R/O	Read as E8h. Points to Vital Products Data register.

14.1.52 SLOT NUMBER REGISTER – OFFSET B0h

Bit	Function	Type	Description
20:16	Expansion Slot Number	R/W	Indicates expansion slot number Reset to 0
21	First in Chassis	R/W	First in chassis Reset to 0
23:22	Reserved	R/O	Returns 0 when read. Reset to 0

14.1.53 CHASSIS NUMBER REGISTER – OFFSET B0h

Bit	Function	Type	Description
31:24	Chassis Number	R/W	Indicates chassis number Reset to 0

14.1.54 CAPABILITY ID REGISTER – OFFSET DCh

Bit	Function	Type	Description
7:0	Enhanced Capabilities ID	R/O	Read as 01h to indicate that these are power management enhanced capability registers.

14.1.55 NEXT ITEM POINTER REGISTER – OFFSET DCh

Bit	Function	Type	Description
15:8	Next Item Pointer	R/O	Read as B0h. Points to slot number register.

14.1.56 POWER MANAGEMENT CAPABILITIES REGISTER – OFFSET DCh

Bit	Function	Type	Description
18:16	Power Management Revision	R/O	Read as 001 to indicate the device is compliant to Revision 1.0 of *PCI Power Management Interface Specifications.*
19	PME# Clock	R/O	Read as 0 to indicate Bridge does not support the PME# pin.
20	Auxiliary Power	R/O	Read as 0 to indicate bridge does not support the PME# pin or an auxiliary power source.
21	Device Specific Initialization	R/O	Read as 0 to indicate bridge does not have device specific initialization requirements.
24:22	Reserved	R/O	Read as 0
25	D1 Power State Support	R/O	Read as 0 to indicate bridge does not support the D1 power management state.
26	D2 Power State Support	R/O	Read as 0 to indicate bridge does not support the D2 power management state.
31:27	PME# Support	R/O	Read as 0 to indicate bridge does not support the PME# pin.

14.1.57 POWER MANAGEMENT DATA REGISTER – OFFSET E0h

Bit	Function	Type	Description
1:0	Power State	R/W	Indicates the current power state of bridge. If an unimplemented power state is written to this register, bridge completes the write transaction, ignores the write data, and does not change the value of the field. Writing a value of D0 when the previous state was D3 cause a chip reset without asserting S_RESET# 00: D0 state 01: D1 state (supported if bit[25] offset DCh is HIGH) 10: D2 state (supported if bit[26] offset DCh is HIGH) 11: D3 state Reset to 0
7:2	Reserved	R/O	Read as 0
8	PME_L Enable	R/O	Read as 0 as bridge does not support the PME# pin.

Bit	Function	Type	Description
12:9	Data Select	R/O	Read as 0 as the data register is not implemented.
14:13	Data Scale	R/O	Read as 0 as the data register is not implemented.
15	PME status	R/O	Read as 0 as the PME# pin is not implemented.

14.1.58 PPB SUPPORT EXTENSIONS REGISTER – OFFSET E0h

Bit	Function	Type	Description
21:16	Reserved	R/O	Reserved. Reset to 0
22	B2_B3	R/O	**B2_B3 Support for $D3_{HOT}$:** When BPCCE is read as 1, this bit is driven as a logic level 1 to indicate that the secondary bus clock outputs will be stopped and driven LOW when the device is placed in $D3_{HOT}$. This bit is undefined when BPCCE is read as 0.
23	Bus Power/Clock Control Enable	R/O	**Bus Power / Clock Control Enable:** When the BPCCE pin is tied HIGH, this bit is read as a 1 to indicate that the bus power/clock control mechanism is enabled. When the BPCCE pin is tied LOW, this bit is read as a 0 to indicate that the bus power / clock control mechanism is disabled.

14.1.59 DATA REGISTER – OFFSET E0h

Bit	Function	Type	Description
31:24	Data	R/O	**Data Register:** Register is not implemented and is read as 00h. Reset to 0.

14.1.60 CAPABILITY ID REGISTER – OFFSET E4h

Bit	Function	Type	Description
7:0	Capability ID	R/O	Read as 06h to indicate these are CompactPCI Hot Swap registers

14.1.61 NEXT POINTER REGISTER – OFFSET E4h

Bit	Function	Type	Description
15:8	Next Pointer	R/O	Read as 00h to indicate end of pointer

14.1.62 HOT SWAP CONTROL AND STATUS REGISTER – OFFSET E4h

Bit	Function	Type	Description
16	Device Hiding	R/W	0: Device hiding not armed 1: Device hiding armed Reset to 0
17	ENUM# Signal Mask	R/W	0: Mask ENUM# signal 1: Enable ENUM# signal Reset to 0
18	Pending Insertion / Extraction	R/O	0: INS is not armed and neither INS nor EXT has a value of 1 1: either INS or EXT has a value of 1 or INS is armed Reset to 0
19	LED on/off	R/W	0: LED on 1: LED off Reset to 0

Bit	Function	Type	Description
21:20	PI (Programming Interface)	R/O	Read as 01 to indicate in addition to the features of Programming Interface 0, Device Hiding, the DHA bit and the PIE bit are implemented
22	EXT (ENUM# Status – Extraction)	R/WC	0: ENUM# is not asserted 1: ENUM# is asserted Reset to 0
23	INS (ENUM# Status – Insertion)	R/WC	0: ENUM# is not asserted 1: ENUM# is asserted Reset to 0
31:24	Reserved	R/O	Returns 0 when read. Reset to 0

14.1.63 CAPABILITY ID REGISTER – OFFSET E8h

Bit	Function	Type	Description
7:0	Capability ID	R/O	Read as 03h to indicate these are VPD registers

14.1.64 NEXT POINTER REGISTER – OFFSET E8h

Bit	Function	Type	Description
15:8	Next Pointer	R/O	E4: HS_EN is 1 00: HS_EN is 0

14.1.65 VPD REGISTER – OFFSET E8h

Bit	Function	Type	Description
17:16	Reserved	R/O	Returns 0 when read. Reset to 0
23:18	VPD Address	R/W	VPD address for read / write cycle
30:24	Reserved	R/O	Returns 0 when read. Reset to 0
31	VPD Operation	R/W	Writing a 0 to this bit generates a read cycle from the EEPROM at the VPD address specified in bits[7:2] of this register. This bit remains 0 until EEPROM cycle is finished, after which it will be set to 1. Data for reads are available at offset ECh. Writing a 1 to this bit generates a write cycle to the EEPROM at the VPD address specified in bits[7:2] of this register. This bit remains at 1 until EEPROM cycle is finished, after which it will be cleared to 0. Reset to 0

14.1.66 VPD DATA REGISTER – OFFSET ECh

Bit	Function	Type	Description
31:0	VPD Data	R/W	VPD data (EEPROM data [address + 0x40]. The least significant byte of this register corresponds to the byte of VPD at the address specified by the VPD address register. The data from or written to this register uses the normal PCI byte transfer capabilities. Reset to 0

15 BRIDGE BEHAVIOR

A PCI cycle is initiated by asserting the FRAME# signal. In a bridge, there are a number of possibilities. Those possibilities are summarized in the table below:

15.1 BRIDGE ACTIONS FOR VARIOUS CYCLE TYPES

Initiator	Target	Response
Master on Primary	Target on Primary	PI7C8154A does not respond. It detects this situation by decoding the address as well as monitoring the P_DEVSEL# for other fast and medium devices on the Primary Port.
Master on Primary	Target on Secondary	PI7C8154A asserts P_DEVSEL#, terminates the cycle normally if it is able to be posted, otherwise return with a retry. It then passes the cycle to the appropriate port. When the cycle is complete on the target port, it will wait for the initiator to repeat the same cycle and end with normal termination.
Master on Primary	Target not on Primary nor Secondary Port	PI7C8154A does not respond and the cycle will terminate as master abort.
Master on Secondary	Target on the same Secondary Port	PI7C8154A does not respond.
Master on Secondary	Target on Primary or the other Secondary Port	PI7C8154A asserts S_DEVSEL#, terminates the cycle normally if it is able to be posted, otherwise returns with a retry. It then passes the cycle to the appropriate port. When cycle is complete on the target port, it will wait for the initiator to repeat the same cycle and end with normal termination.
Master on Secondary	Target not on Primary nor the other Secondary Port	PI7C8154A does not respond.

15.2 ABNORMAL TERMINATION (INITIATED BY BRIDGE MASTER)

15.2.1 MASTER ABORT

Master abort indicates that when PI7C8154A acts as a master and receives no response (i.e., no target asserts DEVSEL# or S_DEVSEL#) from a target, the bridge deasserts FRAME# and then de-asserts IRDY#.

15.2.2 PARITY AND ERROR REPORTING

Parity must be checked for all addresses and write data. Parity is defined on the P_PAR, P_PAR64, S_PAR, and S_PAR64 signals. Parity should be even (i. e. an even number of '1's) across AD, CBE, and PAR. Parity information on PAR is valid the cycle after AD and CBE are valid. For reads, even parity must be generated using the initiators CBE signals combined with the read data. Again, the PAR signal corresponds to read data from the previous data phase cycle.

15.2.3 REPORTING PARITY ERRORS

For all address phases, if a parity error is detected, the error should be reported on the P_SERR# signal by asserting P_SERR# for one cycle and then tri-stating two cycles after the bad address. P_SERR# can only be asserted if bit 6 and 8 in the Command Register are both set to 1. For write data phases, a parity error should be reported by asserting the P_PERR# signal two cycles after the data phase and should remain asserted for one cycle when bit 6 in the Command register is set to a 1. The target reports any type of data parity errors during write cycles, while the master reports data parity errors during read cycles.

Detection of an address parity error will cause the PCI-to-PCI Bridge target to not claim the bus (P_DEVSEL# remains inactive) and the cycle will then terminate with a Master Abort. When the bridge is acting as master, a data parity error during a read cycle results in the bridge master initiating a Master Abort.

15.2.4 SECONDARY IDSEL MAPPING

When PI7C8154A detects a Type 1 configuration transaction for a device connected to the secondary, it translates the Type 1 transaction to Type 0 transaction on the downstream interface. Type 1 configuration format uses a 5-bit field at P_AD[15:11] as a device number. This is translated to S_AD[31:16] by PI7C8154A.

16 IEEE 1149.1 COMPATIBLE JTAG CONTROLLER

An IEEE 1149.1 compatible Test Access Port (TAP) controller and associated TAP pins are provided to support boundary scan in PI7C8154A for board-level continuity test and diagnostics. The TAP pins assigned are TCK, TDI, TDO, TMS and TRST#. All digital input, output, input/output pins are tested except TAP pins.

The IEEE 1149.1 Test Logic consists of a TAP controller, an instruction register, and a group of test data registers including Bypass and Boundary Scan registers. The TAP controller is a synchronous 16-state machine driven by the Test Clock (TCK) and the Test Mode Select (TMS) pins. An independent power on reset circuit is provided to ensure the machine is in TEST_LOGIC_RESET state at power-up. The JTAG signal lines are not active when the PCI resource is operating PCI bus cycles.

PI7C8154A implements 3 basic instructions: BYPASS, SAMPLE/PRELOAD, and EXTEST.

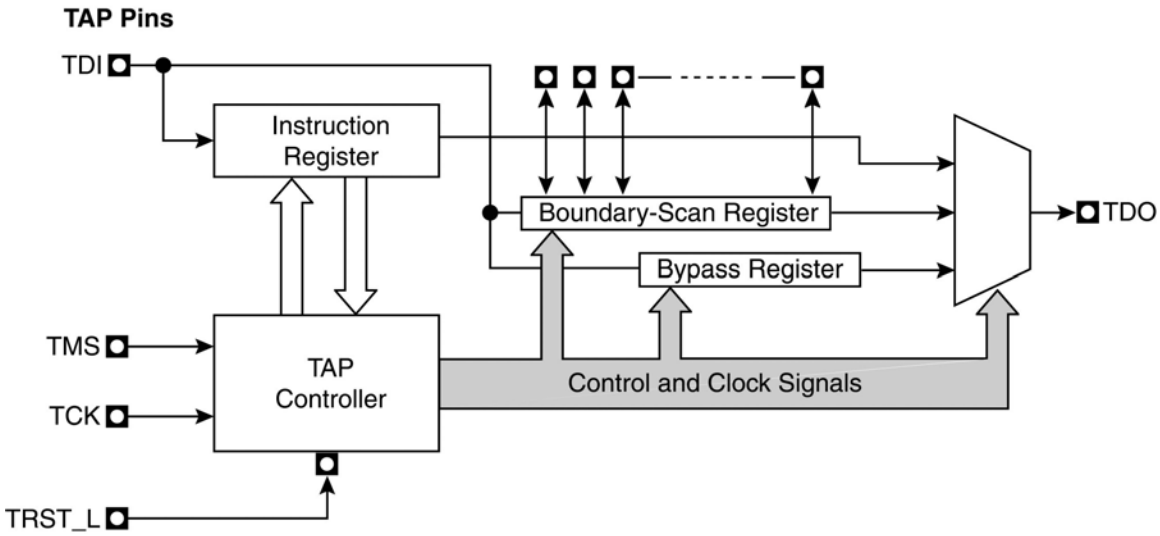
16.1 BOUNDARY SCAN ARCHITECTURE

Boundary-scan test logic consists of a boundary-scan register and support logic. These are accessed through a Test Access Port (TAP). The TAP provides a simple serial interface that allows all processor signal pins to be driven and/or sampled, thereby providing direct control and monitoring of processor pins at the system level.

This mode of operation is valuable for design debugging and fault diagnosis since it permits examination of connections not normally accessible to the test system. The following subsections

describe the boundary-scan test logic elements: TAP pins, instruction register, test data registers and TAP controller. Figure 16-1 illustrates how these pieces fit together to form the JTAG unit.

Figure 16-1 TEST ACCESS PORT DIAGRAM



16.1.1 TAP PINS

The PI7C8154A's TAP pins form a serial port composed of four input connections (TMS, TCK, TRST# and TDI) and one output connection (TDO). These pins are described in Table 16-1. The TAP pins provide access to the instruction register and the test data registers.

16.1.2 INSTRUCTION REGISTER

The Instruction Register (IR) holds instruction codes. These codes are shifted in through the Test Data Input (TDI) pin. The instruction codes are used to select the specific test operation to be performed and the test data register to be accessed.

The instruction register is a parallel-loadable, master/slave-configured 5-bit wide, serial-shift register with latched outputs. Data is shifted into and out of the IR serially through the TDI pin clocked by the rising edge of TCK. The shifted-in instruction becomes active upon latching from the master stage to the slave stage. At that time the IR outputs along with the TAP finite state machine outputs are decoded to select and control the test data register selected by that instruction. Upon latching, all actions caused by any previous instructions terminate.

The instruction determines the test to be performed, the test data register to be accessed, or both. The IR is two bits wide. When the IR is selected, the most significant bit is connected to TDI, and the least significant bit is connected to TDO. The value presented on the TDI pin is shifted into the IR on each rising edge of TCK. The TAP controller captures fixed parallel data (1101 binary). When a new instruction is shifted in through TDI, the value 1101(binary) is always shifted out through TDO, least significant bit first. This helps identify instructions in a long chain of serial data from several devices.

Upon activation of the TRST# reset pin, the latched instruction asynchronously changes to the id code instruction. When the TAP controller moves into the test state other than by reset activation, the opcode changes as TDI shifts, and becomes active on the falling edge of TCK.

16.2 BOUNDARY SCAN INSTRUCTION SET

The PI7C8154A supports three mandatory boundary-scan instructions (BYPASS, SAMPLE and EXTEST). Table 16-1 shown below lists the PI7C8154A's boundary-scan instruction codes.

Table 16-1 TAP PINS

Instruction / Requisite	Opcode (binary)	Description
EXTEST IEEE 1149.1 Required	00000	EXTEST initiates testing of external circuitry, typically board-level interconnects and off chip circuitry. EXTEST connects the boundary-scan register between TDI and TDO. When EXTEST is selected, all output signal pin values are driven by values shifted into the boundary-scan register and may change only of the falling edge of TCK. Also, when EXTEST is selected, all system input pin states must be loaded into the boundary-scan register on the rising-edge of TCK.
SAMPLE IEEE 1149.1 Required	0001	SAMPLE performs two functions: ▪ A snapshot of the sample instruction is captured on the rising edge of TCK without interfering with normal operation. The instruction causes boundary-scan register cells associated with outputs to sample the value being driven. ▪ On the falling edge of TCK, the data held in the boundary-scan cells is transferred to the slave register cells. Typically, the slave latched data is applied to the system outputs via the EXTEST instruction.
INTSCAN	00010	Enable internal SCAN test
CLAMP	00100	CLAMP instruction allows the state of the signals driven from component pins to be determined from the boundary-scan register while the bypass register is selected as the serial path between TDI and TDO. The signal driven from the component pins will not change while the CLAMP instruction is selected.
BYPASS	11111	BYPASS instruction selects the one-bit bypass register between TDI and TDO pins. 0 (binary) is the only instruction that accesses the bypass register. While this instruction is in effect, all other test data registers have no effect on system operation. Test data registers with both test and system functionality performs their system functions when this instruction is selected.

16.3 TAP TEST DATA REGISTERS

The PI7C8154A contains two test data registers (bypass and boundary-scan). Each test data register selected by the TAP controller is connected serially between TDI and TDO. TDI is connected to the test data register's most significant bit. TDO is connected to the least significant bit. Data is shifted one bit position within the register towards TDO on each rising edge of TCK. While any register is selected, data is transferred from TDI to TDO without inversion. The following sections describe each of the test data registers.

16.4 BYPASS REGISTER

The required bypass register, a one-bit shift register, provides the shortest path between TDI and TDO when a bypass instruction is in effect. This allows rapid movement of test data to and from other components on the board. This path can be selected when no test operation is being performed on the PI7C8154A.

16.5 BOUNDARY SCAN REGISTER

The boundary-scan register contains a cell for each pin as well as control cells for I/O and the high-impedance pin. Table 16-2shows the bit order of the PI7C8154A boundary-scan register. All table cells that contain "Control" select the direction of bi-directional pins or high-impedance output pins. When a "1" is loaded into the control cell, the associated pin(s) are high-impedance or selected as output.

The boundary-scan register is a required set of serial-shiftable register cells, configured in master/slave stages and connected between each of the PI7C8154A's pins and on-chip system logic. The VDD, GND, and JTAG pins are NOT in the boundary-scan chain.

The boundary-scan register cells are dedicated logic and do not have any system function. Data may be loaded into the boundary-scan register master cells from the device input pins and output pin-drivers in parallel by the mandatory SAMPLE and EXTEST instructions. Parallel loading takes place on the rising edge of TCK.

Data may be scanned into the boundary-scan register serially via the TDI serial input pin, clocked by the rising edge of TCK. When the required data has been loaded into the master-cell stages, it can be driven into the system logic at input pins or onto the output pins on the falling edge of TCK state. Data may also be shifted out of the boundary-scan register by means of the TDO serial output pin at the falling edge of TCK.

16.6 TAP CONTROLLER

The TAP (Test Access Port) controller is a 4-state synchronous finite state machine that controls the sequence of test logic operations. The TAP can be controlled via a bus master. The bus master can be either automatic test equipment or a component (i.e., PLD) that interfaces to the TAP. The TAP controller changes state only in response to a rising edge of TCK. The value of the test mode state (TMS) input signal at a rising edge of TCK controls the sequence of state changes. The TAP controller is initialized after power-up by applying a low to the TRST# pin. In addition, the TAP controller can be initialized by applying a high signal level on the TMS input for a minimum of five TCK periods.

For greater detail on the behavior of the TAP controller, test logic in each controller state and the state machine and public instructions, refer to the IEEE 1149.1 Standard Test Access Port and Boundary-Scan Architecture document (available from the IEEE).

Table 16-2 JTAG BOUNDARY REGISTER ORDER

Boundary-Scan Register Number	Pin Name	Ball Location	Type
0	S_PAR64	N21	BIDIR
1	S_AD[32]	M21	BIDIR
2	S_AD[33]	M23	BIDIR
3	S_AD[34]	M22	BIDIR
4	S_AD[35]	L22	BIDIR
5	S_AD[36]	L21	BIDIR
6	S_AD[37]	L23	BIDIR
7	S_AD[38]	K21	BIDIR
8	S_AD[39]	K22	BIDIR
9	S_AD[40]	K23	BIDIR
10	S_AD[41]	J22	BIDIR
11	S_AD[42]	J20	BIDIR
12	S_AD[43]	J23	BIDIR
13	S_AD[44]	H21	BIDIR
14	S_AD[45]	H22	BIDIR
15	S_AD[46]	H23	BIDIR
16	S_AD[47]	G21	BIDIR
17	S_AD[48]	G22	BIDIR
18	S_AD[49]	G20	BIDIR
19	S_AD[50]	F22	BIDIR
20	S_AD[51]	F23	BIDIR
21	S_AD[52]	F21	BIDIR
22	S_AD[53]	E23	BIDIR
23	S_AD[54]	E21	BIDIR
24	S_AD[55]	D22	BIDIR
25	S_AD[56]	E20	BIDIR
26	S_AD[57]	D21	BIDIR
27	S_AD[58]	C22	BIDIR
28	S_AD[59]	C23	BIDIR
29	S_AD[60]	C21	BIDIR
30	S_AD[61]	D20	BIDIR
31	S_AD[62]	A21	BIDIR
32	S_AD[63]	C20	BIDIR
33	S_CBE[4]	D19	BIDIR
34	S_CBE[5]	A20	BIDIR
35	S_CBE[6]	C19	BIDIR
36	S_CBE[7]	A19	BIDIR
37	S_REQ64#	B19	BIDIR
38	S_ACK64#	C18	BIDIR
39	S_AD[0]	A18	BIDIR
40	S_AD[1]	B18	BIDIR
41	S_AD[2]	A17	BIDIR
42	S_AD[3]	D17	BIDIR
43	S_AD[4]	B17	BIDIR
44	S_AD[5]	C17	BIDIR
45	S_AD[6]	B16	BIDIR
46	S_AD[7]	C16	BIDIR
47	S_CBE[0]	A15	BIDIR
48	S_AD[8]	B15	BIDIR
49	S_AD[9]	C15	BIDIR
50	S_M66EN	A14	BIDIR
51	S_AD[10]	B14	BIDIR
52	S_AD[11]	C14	BIDIR
53	S_AD[12]	D13	BIDIR
54	S_AD[13]	A13	BIDIR
55	S_AD[14]	B13	BIDIR
56	S_AD[15]	C13	BIDIR
57	S_CBE[1]	C12	BIDIR
58	*		CONTROL

Boundary-Scan Register Number	Pin Name	Ball Location	Type
59	S_PAR	B12	BIDIR
60	S_SERR#	B11	INPUT
61	S_PERR#	C11	BIDIR
62	S_LOCK#	A11	BIDIR
63	S_STOP#	C10	BIDIR
64	S_DEVSEL#	B10	BIDIR
65	S_TRDY#	A10	BIDIR
66	S_IRDY#	C9	BIDIR
67	*		CONTROL
68	S_FRAME#	B9	BIDIR
69	S_CBE[2]	D9	BIDIR
70	S_AD[16]	A9	BIDIR
71	S_AD[17]	C8	BIDIR
72	S_AD[18]	B8	BIDIR
73	S_AD[19]	A8	BIDIR
74	S_AD[20]	B7	BIDIR
75	S_AD[21]	D7	BIDIR
76	S_AD[22]	A7	BIDIR
77	S_AD[23]	A6	BIDIR
78	S_CBE[3]	C6	BIDIR
79	S_AD[24]	B5	BIDIR
80	S_AD[25]	C5	BIDIR
81	S_AD[26]	B4	BIDIR
82	S_AD[27]	A4	BIDIR
83	S_AD[28]	C4	BIDIR
84	S_AD[29]	B3	BIDIR
85	S_AD[30]	A3	BIDIR
86	*		CONTROL
87	S_AD[31]	C3	BIDIR
88	S_REQ#[0]	D4	INPUT
89	S_REQ#[1]	C1	INPUT
90	S_REQ#[2]	C2	INPUT
91	S_REQ#[3]	D3	INPUT
92	S_REQ#[4]	E4	INPUT
93	S_REQ#[5]	D1	INPUT
94	S_REQ#[6]	D2	INPUT
95	S_REQ#[7]	E3	INPUT
96	S_REQ#[8]	E1	INPUT
97	S_GNT#[0]	E2	BIDIR
98	S_GNT#[1]	F3	BIDIR
99	S_GNT#[2]	F1	BIDIR
100	S_GNT#[3]	F2	BIDIR
101	*		CONTROL
102	S_GNT#[4]	G1	BIDIR
103	S_GNT#[5]	G4	BIDIR
104	S_GNT#[6]	G2	BIDIR
105	S_GNT#[7]	G3	BIDIR
106	S_GNT#[8]	H1	BIDIR
107	S_RESET#	H2	BIDIR
108	S_CLKIN	J4	INPUT
109	S_CFN#	K1	INPUT
110	GPIO[3]	K2	BIDIR
111	GPIO[2]	K3	BIDIR
112	GPIO[1]	L4	BIDIR
113	GPIO[0]	L1	BIDIR
114	S_CLKOUT[0]	L2	OUTPUT
115	*		CONTROL
116	S_CLKOUT[1]	L3	OUTPUT
117	S_CLKOUT[2]	M3	OUTPUT
118	S_CLKOUT[3]	M1	OUTPUT
119	S_CLKOUT[4]	M2	OUTPUT
120	S_CLKOUT[5]	N3	OUTPUT

Boundary-Scan Register Number	Pin Name	Ball Location	Type
121	S_CLKOUT[6]	N1	OUTPUT
122	S_CLKOUT[7]	P3	OUTPUT
123	S_CLKOUT[8]	P2	OUTPUT
124	S_CLKOUT[9]	P1	OUTPUT
125	P_RESET#	R3	INPUT
126	P_GNT#	R2	INPUT
127	BPCCE	R4	INPUT
128	P_CLK	T3	INPUT
129	*		CONTROL
130	P_REQ#	U3	BIDIR
131	P_AD[31]	U2	BIDIR
132	P_AD[30]	U4	BIDIR
133	P_AD[29]	U1	BIDIR
134	P_AD[28]	V2	BIDIR
135	P_AD[27]	V1	BIDIR
136	P_AD[26]	V3	BIDIR
137	P_AD[25]	W2	BIDIR
138	P_AD[24]	W1	BIDIR
139	P_CBE[3]	Y2	BIDIR
140	P_IDSEL]	Y1	INPUT
141	P_AD[23]	W4	BIDIR
142	P_AD[22]	Y3	BIDIR
143	P_AD[21]	AA1	BIDIR
144	P_AD[20]	AA3	BIDIR
145	P_AD[19]	Y4	BIDIR
146	P_AD[18]	AB3	BIDIR
147	P_AD[17]	AA4	BIDIR
148	P_AD[16]	Y5	BIDIR
149	*		CONTROL
150	P_CBE[2]	AB4	BIDIR
151	P_FRAME#	AA5	BIDIR
152	P_IRDY#	AC5	BIDIR
153	P_TRDY#	AB5	BIDIR
154	P_DEVSEL#	AA6	BIDIR
155	P_STOP#	AC6	BIDIR
156	P_LOCK#	AB6	INPUT
157	*		CONTROL
158	P_PERR#	AC7	BIDIR
159	P_SERR#	Y7	OUTPUT
160	P_PAR	AB7	BIDIR
161	P_CBE[1]	AA7	BIDIR
162	P_AD[15]	AB8	BIDIR
163	P_AD[14]	AA8	BIDIR
164	P_AD[13]	AC9	BIDIR
165	P_AD[12]	AB9	BIDIR
166	P_AD[11]	AA9	BIDIR
167	P_AD[10]	AC10	BIDIR
168	P_M66EN	AB10	INPUT
169	P_AD[9]	AA10	BIDIR
170	P_AD[8]	Y11	BIDIR
171	P_CBE[0]	AC11	BIDIR
172	P_AD[7]	AB11	BIDIR
173	P_AD[6]	AA11	BIDIR
174	P_AD[5]	AA12	BIDIR
175	P_AD[4]	AB12	BIDIR
176	P_AD[3]	AB13	BIDIR
177	P_AD[2]	AA13	BIDIR
178	P_AD[1]	Y13	BIDIR
179	P_AD[0]	AA14	BIDIR
180	P_ACK64#	AB14	BIDIR
181	P_REQ64#	AC14	BIDIR
182	P_CBE[7]	AA15	BIDIR

Boundary-Scan Register Number	Pin Name	Ball Location	Type
183	P_CBE[6]	AB15	BIDIR
184	P_CBE[5]	Y15	BIDIR
185	P_CBE[4]	AC15	BIDIR
186	P_AD[63]	AA16	BIDIR
187	P_AD[62]	AB16	BIDIR
188	P_AD[61]	AA17	BIDIR
189	P_AD[60]	AB17	BIDIR
190	P_AD[59]	Y17	BIDIR
191	P_AD[58]	AB18	BIDIR
192	P_AD[57]	AC18	BIDIR
193	P_AD[56]	AA18	BIDIR
194	P_AD[55]	AC19	BIDIR
195	P_AD[54]	AA19	BIDIR
196	P_AD[53]	AB20	BIDIR
197	P_AD[52]	Y19	BIDIR
198	P_AD[51]	AA20	BIDIR
199	P_AD[50]	AB21	BIDIR
200	P_AD[49]	AC21	BIDIR
201	P_AD[48]	AA21	BIDIR
202	P_AD[47]	Y20	BIDIR
203	P_AD[46]	AA23	BIDIR
204	P_AD[45]	Y21	BIDIR
205	P_AD[44]	W20	BIDIR
206	P_AD[43]	Y23	BIDIR
207	P_AD[42]	W21	BIDIR
208	P_AD[41]	W23	BIDIR
209	P_AD[40]	W22	BIDIR
210	P_AD[39]	V21	BIDIR
211	P_AD[38]	V23	BIDIR
212	P_AD[37]	V22	BIDIR
213	P_AD[36]	U23	BIDIR
214	P_AD[35]	U20	BIDIR
215	P_AD[34]	U22	BIDIR
216	*		CONTROL
217	P_AD[33]	T23	BIDIR
218	P_AD[32]	T22	BIDIR
219	P_PAR64	T21	BIDIR
220	CONFIG66	R22	INPUT
221	MSK_IN	R21	INPUT

17 ELECTRICAL AND TIMING SPECIFICATIONS

17.1 MAXIMUM RATINGS

(Above which the useful life may be impaired. For user guidelines, not tested).

Storage Temperature	-65°C to 150°C
Ambient Temperature with Power Applied	-40°C to 85°C
Supply Voltage to Ground Potentials (AV_{CC} and V_{DD} only]	-0.3V to 3.6V
Voltage at Input Pins	-0.5V to 5.5V
Junction Temperature, T_J	125°C

Note:
Stresses greater than those listed under MAXIMUM RATINGS may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods of time may affect reliability.

17.2 DC SPECIFICATIONS

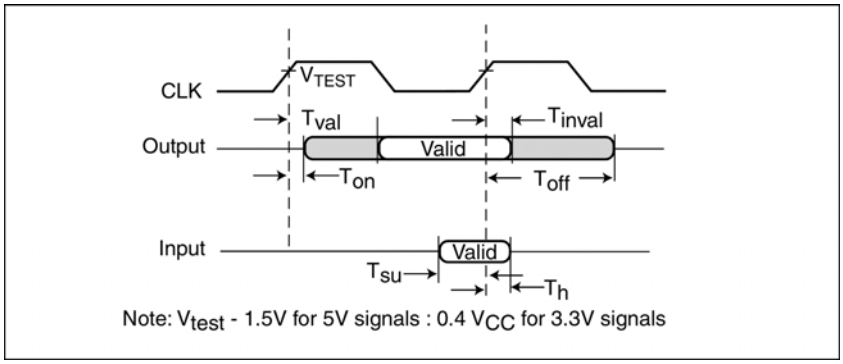
Symbol	Parameter	Condition	Min.	Max.	Units	Notes
V_{DD}	Supply Voltage		3	3.6	V	
V_{IH}	Input HIGH Voltage		$0.5V_{DD}$	V_{DD} + 0.5	V	1
V_{IL}	Input LOW Voltage		-0.5	0.3 V_{DD}	V	1
V_{OH}	Output HIGH Voltage	I_{out} = -500µA	$0.9V_{DD}$		V	
V_{OL}	Output LOW Voltage	I_{out} = 1500µA		0.1 V_{DD}	V	
V_{OH5V}	5V Signaling Output HIGH Voltage	I_{out} = -2 mA	2.4		V	
V_{OL5V}	5V Signaling Output LOW Voltage	I_{out} = 6 mA		0.5	V	
I_{IL}	Input Leakage Current	$0 < V_{in} < V_{DD}$		±10	µA	
C_{IN}	Input Pin Capacitance			10	pF	
C_{CLK}	CLK Pin Capacitance		5	12	pF	
C_{IDSEL}	IDSEL Pin Capacitance			8	pF	
L_{PIN}	Pin Inductance			20	nH	

Notes:

1. V_{DD} is in reference to the V_{DD} of the input device.

17.3 AC SPECIFICATIONS

Figure 17-1 PCI SIGNAL TIMING MEASUREMENT CONDITIONS



		66 MHz		33 MHz		
Symbol	**Parameter**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
Tsu	Input setup time to CLK – bused signals [1,2,3]	3	-	7	-	ns
Tsu(ptp)	Input setup time to CLK – point-to-point [1,2,3]	5	-	10, 12[4]	-	
Th	Input signal hold time from CLK [1,2]	0	-	0	-	
Tval	CLK to signal valid delay – bused signals [1,2,3]	2	6	2	11	
Tval(ptp)	CLK to signal valid delay – point-to-point [1,2,3]	2	6	2	12	
Ton	Float to active delay [1,2]	2	-	2	-	
Toff	Active to float delay [1,2]	-	14	-	28	

1. See Figure 17-1 PCI Signal Timing Measurement Conditions.
2. All primary interface signals are synchronized to P_CLK. All secondary interface signals are synchronized to S_CLKOUT.
3. Point-to-point signals are P_REQ#, S_REQ#[7:0], P_GNT#, S_GNT#[7:0], HSLED, HS_SW#, HS_EN, and ENUM#. Bused signals are P_AD, P_BDE#, P_PAR, P_PERR#, P_SERR#, P_FRAME#, P_IRDY#, P_TRDY#, P_LOCK#, P_DEVSEL#, P_STOP#, P_IDSEL, P_PAR64, P_REQ64#, P_ACK64#, S_AD, S_CBE#, S_PAR, S_PERR#, S_SERR#, S_FRAME#, S_IRDY#, S_TRDY#, S_LOCK#, S_DEVSEL#, S_STOP#, S_PA64, S_REQ64#, and S_ACK64#.
4. REQ# signals have a setup of 10ns and GNT# signals have a setup of 12ns.

17.4 66MHZ PCI SIGNALING TIMING

Symbol	**Parameter**	**Condition**	**Min.**	**Max.**	**Units**
T_{SKEW}	SKEW among S_CLKOUT[9:0]		0	0.250	ns
T_{DELAY}	DELAY between PCLK and S_CLKOUT[9:0]	20pF load	3.47	4.20	
T_{CYCLE}	P_CLK, S_CLKOUT[9:0] cycle time		15	30	
T_{HIGH}	P_CLK, S_CLKOUT[9:0] HIGH time		6		
T_{LOW}	P_CLK, S_CLKOUT[9:0] LOW time		6		

17.5 33MHZ PCI SIGNALING TIMING

Symbol	**Parameter**	**Condition**	**Min.**	**Max.**	**Units**
T_{SKEW}	SKEW among S_CLKOUT[9:0]		0	0.250	ns

T_{DELAY}	DELAY between PCLK and S_CLKOUT[9:0]	20pF load	3.47	4.20	
T_{CYCLE}	P_CLK, S_CLKOUT[9:0] cycle time		30		
T_{HIGH}	P_CLK, S_CLKOUT[9:0] HIGH time		11		
T_{LOW}	P_CLK, S_CLKOUT[9:0] LOW time		11		

17.6 RESET TIMING

Symbol	Parameter	Min.	Max.	Units
T_{RST}	P_RESET# active time after power stable	1	-	us
$T_{RST-CLK}$	P_RESET# active time after P_CLK stable	100	-	us
$T_{RST-OFF}$	P_RESET# active-to-output float delay	-	40	ns
T_{SRST}	S_RESET# active after P_RESET# assertion	-	40	ns
$T_{SRST-ON}$	S_RESET# active time after S_CLKIN stable	100	-	us
T_{DRST}	S_RESET# deassertion after P_RESET# deassertion	20	25	cycles

17.7 GPIO TIMING (66MHZ & 33MHZ)

Symbol	Parameter	Min.	Max.	Units
T_{VGPIO}	S_CLKIN to GPIO output valid	2	12	ns
T_{GON}	GPIO float to output valid	2	-	ns
T_{GOF}F	GPIO active to float delay	-	28	ns
T_{GSU}	GPIO-to-S_CLKIN setup time	7	-	ns
T_{GH}	GPIO hold time after S_CLKIN	0	-	nx
T_{GCVAL}	S_CLKIN-to-GPIO shift clock output valid	-	13.5	ns
T_{GCYC}	GPIO[0] cycle time	30	∞	ns
T_{GSVAL}	GPIO[0] to GPIO[2] shift control output valid	-	8	ns
T_{MSU}	MSK_IN setup time to GPIO[0]	15	-	ns
T_{MH}	MSK_IN hold time after GPIO[0]	0	-	ns

17.8 JTAG TIMING

Symbol	Parameter	Min.	Max.	Units
T_{IF}	TCK frequency	0	10	MHz
T_{JP}	TCK period	100	∞	ns
T_{JHT}	TCK HIGH time	45	-	ns
T_{JLT}	TCK LOW time	45	-	ns
T_{JRT}	TCK rise time[1]	-	10	ns
T_{JFT}	TCK fall time[2]	-	10	ns
T_{JE}	TDI, TMS setup time to TCK rising edge	10	-	ns
T_{JH}	TDI, TMS hold time from TCK rising edge	25	-	ns
T_{JD}	TDO valid delay from TCK falling edge[3]	-	30	ns
T_{JFD}	TDO float delay from TCK falling edge	-	30	ns

1. Measured between 0.8V to 2.0V
2. Measured between 2.0V to 0.8V
3. C1=50pF

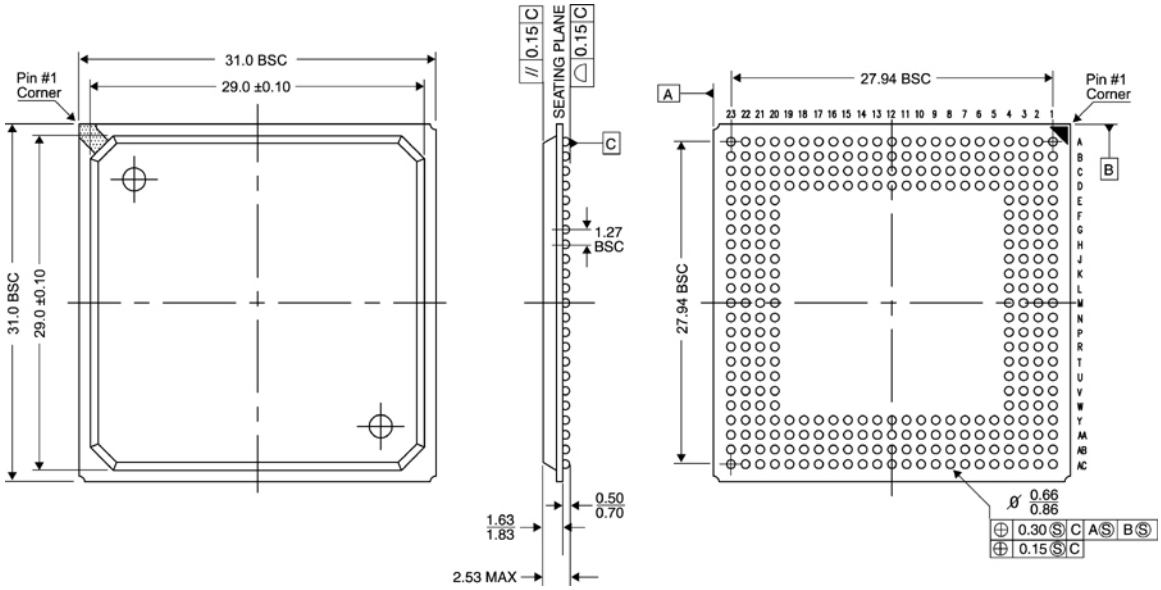
17.9 POWER CONSUMPTION

Parameter	Typical	Units
Power Consumption at 66MHz	1.38	W
Supply Current, I_{CC}	417	mA

18 PACKAGE INFORMATION

18.1 304-BALL PBGA PACKAGE DIAGRAM

Figure 18-1 304-BALL PBGA PACKAGE OUTLINE



Thermal characteristics can be found on the web:
http://www.pericom.com/packaging/mechanicals.php

18.2 ORDERING INFORMATION

Part Number	Speed	Pin – Package	Temperature
PI7C8154ANAE	66MHz	304 – PBGA (Pb-free & Green)	-40°C to 85°C

19 APPENDIX

19.1 PI7C8154/A/B vs. Intel 21154, PBGA-304

Pericom PI7C8154/A/B provides direct replacement to Intel 21154.

Following table is PI7C8154/A/B pin comparison to Intel 21154

Pin Number	PI7C8154/A/B	Intel 21154
A22	VDD / * EEDATA	VDD
A23	VSS / * EECLK	VSS
B6	VDD / *NC	VDD
D11	PMEENA_L	VDD
V20	Reserved	VSS
Y18	Reserved	VDD
AA22	VSS / * NC	VSS
AB1	VDD / *ASYNC_SEL#	VDD
AB2	VSS / * ASYNC_CLKIN	VSS
AC22	VDD / * EE_EN#	VDD

- Pin A22: VDD / *EEDATA – For 8154A/B, this pin is used as VDD and serial data interface for EEPROM. For Intel 21154, this pin is defined as VDD.
- Pin A23: VSS / *EECLK – For 8154A/B, this pin is used as VSS and serial clock interface for EEPROM. For Intel 21154, this pin is defined as VSS.
- Pin B6: VDD / *NC – For 8154/A, this pin is defined as power pin. For 8154B, this pin can be no connected. For Intel 21154 is defined as VDD
- Pin D11: PMEENA_L is used to indicate the secondary devices are capable of asserting PME_L or not. For Intel 21154, this pin is defined as VDD.
- Pin V20: For pin V20, it must be tied to ground. For Intel 21154, this pin is defined as VSS.
- Pin Y18: For pin Y18, it must be tied to VDD. For Intel 21154, this pin is defined as VDD.
- Pin AA22: VSS / *NC – For 8154/A, this pin is defined as ground pin. For 8154B, this pin can be no connected. For Intel 21154, this pin is defined as VSS.
- Pin AB1: VDD / *ASYNC_SEL# - For 8154/A, this pin is defined as power pin. For 8154B, this pin is used as enables asynchronous mode for the bridge. For Intel 21154, this pin is defined as VDD.
  - 0: Secondary bus clock outputs (S_CLKOUT [9:0]) will use the clock signal from ASYNC_CLKIN input instead of the P_CLK.
  - 1: Secondary bus clock outputs (S_CLKOUT [9:0]) will use the P_CLK input for synchronous operation.
- Pin AB2: VSS / * ASYNC_CLKIN – For 8154/A, this pin is defined as ground pin. For 8154B, this pin is used as an external clock input in order to generate the secondary clock outputs (S_CLKOUT [9:0]) when enabled by ASYNC_SEL#. For Intel 21154, this pin is defined as VSS.
- Pin AC22: VDD / * EE_EN# – For 8154, this pin is defined as power pin. For 8154A/B, this pin is used as enable EEPROM interface when it is tied low. For Intel 21154, this pin is defined as VDD.