# PI7C8150A

## 2-PORT PCI-to-PCI BRIDGE

REVISION 1.1

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

06-0057

## 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.

All other trademarks are of their respective companies.

# REVISION HISTORY

| Date | Revision Number | Description |
|---|---|---|
| 10/15/03 | 1.00 | First Release of Data Sheet |
| 05/20/05 | 1.01 | Corrected VDD and VSS pin assignments in Section 2.2.7. Removed pins 106 and 155 (R16 and B14) as these should be MS1 and MS0 respectively.<br><br>Added Pb-free parts in the Ordering Information |
| 04/20/06 | 1.1 | Removed 'Advance Information' from header<br><br>Removed 'solutions@pericom.com' email link<br><br>Changed logos<br><br>Updated compliance to *PCI Local Bus Specification*, revision 2.3 |

*This page intentionally left blank.*

# TABLE OF CONTENTS

**1 INTRODUCTION ..... 11**

**2 SIGNAL DEFINITIONS ..... 12**

2.1 SIGNAL TYPES ..... 12
2.2 SIGNALS ..... 12
***2.2.1 PRIMARY BUS INTERFACE SIGNALS ..... 12***
***2.2.3 CLOCK SIGNALS ..... 15***
***2.2.4 MISCELLANEOUS SIGNALS ..... 16***
***2.2.5 GENERAL PURPOSE I/O INTERFACE SIGNALS ..... 17***
***2.2.6 JTAG BOUNDARY SCAN SIGNALS ..... 17***
***2.2.7 POWER AND GROUND ..... 17***
2.3 PIN LIST – 208-PIN FQFP ..... 18
2.4 PIN LIST – 256-BALL PBGA ..... 20

**3 PCI BUS OPERATION ..... 21**

3.1 TYPES OF TRANSACTIONS ..... 22
3.2 SINGLE ADDRESS PHASE ..... 22
3.3 DEVICE SELECT (DEVSEL_L) GENERATION ..... 23
3.4 DATA PHASE ..... 23
3.5 WRITE TRANSACTIONS ..... 23
***3.5.1 MEMORY WRITE TRANSACTIONS ..... 23***
***3.5.2 MEMORY WRITE AND INVALIDATE ..... 24***
***3.5.3 DELAYED WRITE TRANSACTIONS ..... 25***
***3.5.4 WRITE TRANSACTION ADDRESS BOUNDARIES ..... 26***
***3.5.5 BUFFERING MULTIPLE WRITE TRANSACTIONS ..... 26***
***3.5.6 FAST BACK-TO-BACK TRANSACTIONS ..... 26***
3.6 READ TRANSACTIONS ..... 27
***3.6.1 PREFETCHABLE READ TRANSACTIONS ..... 27***
***3.6.2 NON-PREFETCHABLE READ TRANSACTIONS ..... 27***
***3.6.3 READ PREFETCH ADDRESS BOUNDARIES ..... 28***
***3.6.4 DELAYED READ REQUESTS ..... 28***
***3.6.5 DELAYED READ COMPLETION WITH TARGET ..... 29***
***3.6.6 DELAYED READ COMPLETION ON INITIATOR BUS ..... 29***
***3.6.7 FAST BACK-TO-BACK READ TRANSACTION ..... 30***
3.7 CONFIGURATION TRANSACTIONS ..... 30
***3.7.1 TYPE 0 ACCESS TO PI7C8150A ..... 31***
***3.7.2 TYPE 1 TO TYPE 0 CONVERSION ..... 31***
***3.7.3 TYPE 1 TO TYPE 1 FORWARDING ..... 33***
***3.7.4 SPECIAL CYCLES ..... 34***
3.8 TRANSACTION TERMINATION ..... 34
***3.8.1 MASTER TERMINATION INITIATED BY PI7C8150A ..... 35***
***3.8.2 MASTER ABORT RECEIVED BY PI7C8150A ..... 36***
***3.8.3 TARGET TERMINATION RECEIVED BY PI7C8150A ..... 36***
3.8.3.1 DELAYED WRITE TARGET TERMINATION RESPONSE ..... 37
3.8.3.2 POSTED WRITE TARGET TERMINATION RESPONSE ..... 37
3.8.3.3 DELAYED READ TARGET TERMINATION RESPONSE ..... 38
***3.8.4 TARGET TERMINATION INITIATED BY PI7C8150A ..... 39***
3.8.4.1 TARGET RETRY ..... 39

3.8.4.2 TARGET DISCONNECT ..... 40
3.8.4.3 TARGET ABORT ..... 40

**4 ADDRESS DECODING ..... 41**

4.1 ADDRESS RANGES ..... 41
4.2 I/O ADDRESS DECODING ..... 41
***4.2.1 I/O BASE AND LIMIT ADDRESS REGISTER ..... 42***
***4.2.2 ISA MODE ..... 43***
4.3 MEMORY ADDRESS DECODING ..... 43
***4.3.1 MEMORY-MAPPED I/O BASE AND LIMIT ADDRESS REGISTERS ..... 44***
***4.3.2 PREFETCHABLE MEMORY BASE AND LIMIT ADDRESS REGISTERS ..... 45***
4.4 VGA SUPPORT ..... 46
***4.4.1 VGA MODE ..... 46***
***4.4.2 VGA SNOOP MODE ..... 46***

**5 TRANSACTION ORDERING ..... 47**

5.1 TRANSACTIONS GOVERNED BY ORDERING RULES ..... 47
5.2 GENERAL ORDERING GUIDELINES ..... 48
5.3 ORDERING RULES ..... 48
5.4 DATA SYNCHRONIZATION ..... 51

**6 ERROR HANDLING ..... 52**

6.1 ADDRESS PARITY ERRORS ..... 52
6.2 DATA PARITY ERRORS ..... 53
***6.2.1 CONFIGURATION WRITE TRANSACTIONS TO CONFIGURATION SPACE ..... 53***
***6.2.2 READ TRANSACTIONS ..... 53***
***6.2.3 DELAYED WRITE TRANSACTIONS ..... 54***
***6.2.4 POSTED WRITE TRANSACTIONS ..... 57***
6.3 DATA PARITY ERROR REPORTING SUMMARY ..... 58
6.4 SYSTEM ERROR (SERR_L) REPORTING ..... 62

**7 EXCLUSIVE ACCESS ..... 63**

7.1 CONCURRENT LOCKS ..... 63
7.2 ACQUIRING EXCLUSIVE ACCESS ACROSS PI7C8150A ..... 63
***7.2.1 LOCKED TRANSACTIONS IN DOWNSTREAM DIRECTION ..... 63***
***7.2.2 LOCKED TRANSACTION IN UPSTREAM DIRECTION ..... 65***
7.3 ENDING EXCLUSIVE ACCESS ..... 65

**8 PCI BUS ARBITRATION ..... 66**

8.1 PRIMARY PCI BUS ARBITRATION ..... 66
8.2 SECONDARY PCI BUS ARBITRATION ..... 66
***8.2.1 SECONDARY BUS ARBITRATION USING THE INTERNAL ARBITER ..... 66***
***8.2.2 PREEMPTION ..... 68***
***8.2.3 SECONDARY BUS ARBITRATION USING AN EXTERNAL ARBITER ..... 68***
***8.2.4 BUS PARKING ..... 68***

**9 CLOCKS ..... 69**

9.1 PRIMARY CLOCK INPUTS ..... 69
9.2 SECONDARY CLOCK OUTPUTS ..... 69

**10 GENERAL PURPOSE I/O INTERFACE ..... 69**

10.1 GPIO CONTROL REGISTERS ..... 70
10.2 SECONDARY CLOCK CONTROL ..... 70
10.3 LIVE INSERTION ..... 72

**11 PCI POWER MANAGEMENT ..... 72**

**12 RESET ..... 73**

12.1 PRIMARY INTERFACE RESET ..... 73
12.2 SECONDARY INTERFACE RESET ..... 74
12.3 CHIP RESET ..... 74

**13 SUPPORTED COMMANDS ..... 74**

13.1 PRIMARY INTERFACE ..... 74
13.2 SECONDARY INTERFACE ..... 76

**14 CONFIGURATION REGISTERS ..... 77**

14.1 CONFIGURATION REGISTER ..... 77
***14.1.1 VENDOR ID REGISTER – OFFSET 00h ..... 78***
***14.1.2 DEVICE ID REGISTER – OFFSET 00h ..... 78***
***14.1.3 COMMAND REGISTER – OFFSET 04h ..... 78***
***14.1.4 STATUS REGISTER – OFFSET 04h ..... 79***
***14.1.5 REVISION ID REGISTER – OFFSET 08h ..... 80***
***14.1.6 CLASS CODE REGISTER – OFFSET 08h ..... 80***
***14.1.7 CACHE LINE SIZE REGISTER – OFFSET 0Ch ..... 80***
***14.1.8 PRIMARY LATENCY TIMER REGISTER – OFFSET 0Ch ..... 80***
***14.1.9 HEADER TYPE REGISTER – OFFSET 0Ch ..... 81***
***14.1.10 PRIMARY BUS NUMBER REGISTSER – OFFSET 18h ..... 81***
***14.1.11 SECONDARY BUS NUMBER REGISTER – OFFSET 18h ..... 81***
***14.1.12 SUBORDINATE BUS NUMBER REGISTER – OFFSET 18h ..... 81***
***14.1.13 SECONDARY LATENCY TIMER REGISTER – OFFSET 18h ..... 81***
***14.1.14 I/O BASE REGISTER – OFFSET 1Ch ..... 81***
***14.1.15 I/O LIMIT REGISTER – OFFSET 1Ch ..... 82***
***14.1.16 SECONDARY STATUS REGISTER – OFFSET 1Ch ..... 82***
***14.1.17 MEMORY BASE REGISTER – OFFSET 20h ..... 83***
***14.1.18 MEMORY LIMIT REGISTER – OFFSET 20h ..... 83***
***14.1.19 PEFETCHABLE MEMORY BASE REGISTER – OFFSET 24h ..... 83***
***14.1.20 PREFETCHABLE MEMORY LIMIT REGISTER – OFFSET 24h ..... 84***
***14.1.21 PREFETCHABLE MEMORY BASE ADDRESS UPPER 32-BITS REGISTER – OFFSET 28h ..... 84***
***14.1.22 PREFETCHABLE MEMORY LIMIT ADDRESS UPPER 32-BITS REGISTER – OFFSET 2Ch ..... 84***
***14.1.23 I/O BASE ADDRESS UPPER 16-BITS REGISTER – OFFSET 30h ..... 84***
***14.1.24 I/O LIMIT ADDRESS UPPER 16-BITS REGISTER – OFFSET 30h ..... 85***
***14.1.25 ECP POINTER REGISTER – OFFSET 34h ..... 85***
***14.1.26 INTERRUPT LINE REGISTER – OFFSET 3Ch ..... 85***
***14.1.27 INTERRUPT PIN REGISTER – OFFSET 3Ch ..... 85***
***14.1.28 BRIDGE CONTROL REGISTER – OFFSET 3Ch ..... 85***
***14.1.29 DIAGNOSTIC / CHIP CONTROL REGISTER – OFFSET 40h ..... 87***
***14.1.30 ARBITER CONTROL REGISTER – OFFSET 40h ..... 88***
***14.1.31 EXTENDED CHIP CONTROL REGISTER – OFFSET 48h ..... 88***
***14.1.32 UPSTREAM MEMORY CONTROL REGISTER – OFFSET 48h ..... 89***

*14.1.33 SECONDARY BUS ARBITER PREEMPTION CONTROL REGISTER – OFFSET 4Ch* ..... *89*
*14.1.34 UPSTREAM (S TO P) MEMORY BASE REGISTER – OFFSET 50h* ..... *89*
*14.1.35 UPSTREAM (S TO P) MEMORY LIMIT REGISTER – OFFSET 50h* ..... *90*
*14.1.36 UPSTREAM (S TO P) MEMORY BASE UPPER 32-BITS REGISTER – OFFSET 54h* ..... *90*
*14.1.37 UPSTREAM (S TO P) MEMORY LIMIT UPPER 32-BITS REGISTER – OFFSET 58h* ..... *90*
*14.1.38 P_SERR_L EVENT DISABLE REGISTER – OFFSET 64h* ..... *90*
*14.1.39 GPIO DATA AND CONTROL REGISTER – OFFSET 64h* ..... *91*
*14.1.40 SECONDARY CLOCK CONTROL REGISTER – OFFSET 68h* ..... *92*
*14.1.41 P_SERR_L STATUS REGISTER – OFFSET 68h* ..... *92*
*14.1.42 PORT OPTION REGISTER – OFFSET 74h* ..... *93*
*14.1.43 RETRY COUNTER REGISTER – OFFSET 78h* ..... *95*
*14.1.44 PRIMARY MASTER TIMEOUT COUNTER – OFFSET 80h* ..... *95*
*14.1.45 SECONDARY MASTER TIMEOUT COUNTER – OFFSET 80h* ..... *95*
*14.1.46 CAPABILITY ID REGISTER – OFFSET B0h* ..... *95*
*14.1.47 NEXT POINTER REGISTER – OFFSET B0h* ..... *95*
*14.1.48 SLOT NUMBER REGISTER – OFFSET B0h* ..... *96*
*14.1.49 CHASSIS NUMBER REGISTER – OFFSET B0h* ..... *96*
*14.1.50 CAPABILITY ID REGISTER – OFFSET DCh* ..... *96*
*14.1.51 NEXT ITEM POINTER REGISTER – OFFSET DCh* ..... *96*
*14.1.52 POWER MANAGEMENT CAPABILITIES REGISTER – OFFSET DCh* ..... *96*
*14.1.53 POWER MANAGEMENT DATA REGISTER – OFFSET E0h* ..... *97*
*14.1.54 CAPABILITY ID REGISTER – OFFSET E4h* ..... *97*
*14.1.55 NEXT POINTER REGISTER – OFFSET E4h* ..... *97*

**15 BRIDGE BEHAVIOR ..... 97**

15.1 BRIDGE ACTIONS FOR VARIOUS CYCLE TYPES ..... 98
15.2 ABNORMAL TERMINATION (INITIATED BY BRIDGE MASTER) ..... 98
*15.2.1 MASTER ABORT* ..... *98*
*15.2.2 PARITY AND ERROR REPORTING* ..... *98*
*15.2.3 REPORTING PARITY ERRORS* ..... *99*
*15.2.4 SECONDARY IDSEL MAPPING* ..... *99*

**16 IEEE 1149.1 COMPATIBLE JTAG CONTROLLER ..... 99**

16.1 BOUNDARY SCAN ARCHITECTURE ..... 99
*16.1.1 TAP PINS* ..... *100*
*16.1.2 INSTRUCTION REGISTER* ..... *100*
16.2 BOUNDARY SCAN INSTRUCTION SET ..... 101
16.3 TAP TEST DATA REGISTERS ..... 102
16.4 BYPASS REGISTER ..... 102
16.5 BOUNDARY-SCAN REGISTER ..... 102
16.6 TAP CONTROLLER ..... 102

**17 ELECTRICAL AND TIMING SPECIFICATIONS ..... 106**

17.1 MAXIMUM RATINGS ..... 106
17.2 DC SPECIFICATIONS ..... 106
17.3 AC SPECIFICATIONS ..... 107
17.4 66MHZ TIMING ..... 108
17.5 33MHZ TIMING ..... 108
17.6 POWER CONSUMPTION ..... 108

**18 PACKAGE INFORMATION ..... 109**

18.1 208-PIN FQFP PACKAGE DIAGRAM ..... 109
18.2 256-BALL PBGA PACKAGE DIAGRAM ..... 110
18.3 PART NUMBER ORDERING INFORMATION ..... 110

# LIST OF TABLES

TABLE 2-1. PIN LIST – 208-PIN FQFP ..... 18
TABLE 2-2. PIN LIST – 256-PIN PBGA ..... 20
TABLE 3-1. PCI TRANSACTIONS ..... 22
TABLE 3-2. WRITE TRANSACTION FORWARDING ..... 23
TABLE 3-3. WRITE TRANSACTION DISCONNECT ADDRESS BOUNDARIES ..... 26
TABLE 3-4. READ PREFETCH ADDRESS BOUNDARIES ..... 28
TABLE 3-5. READ TRANSACTION PREFETCHING ..... 28
TABLE 3-6. DEVICE NUMBER TO IDSEL S_AD PIN MAPPING ..... 32
TABLE 3-7. DELAYED WRITE TARGET TERMINATION RESPONSE ..... 37
TABLE 3-8. RESPONSE TO POSTED WRITE TARGET TERMINATION ..... 38
TABLE 3-9. RESPONSE TO DELAYED READ TARGET TERMINATION ..... 38
TABLE 5-1. SUMMARY OF TRANSACTION ORDERING ..... 50
TABLE 6-1. SETTING THE PRIMARY INTERFACE DETECTED PARITY ERROR BIT ..... 58
TABLE 6-2. SETTING SECONDARY INTERFACE DETECTED PARITY ERROR BIT ..... 59
TABLE 6-3. SETTING PRIMARY INTERFACE MASTER DATA PARITY ERROR DETECTED BIT ..... 59
TABLE 6-4. SETTING SECONDARY INTERFACE MASTER DATA PARITY ERROR DETECTED BIT ..... 60
TABLE 6-5. ASSERTION OF P_PERR_L ..... 60
TABLE 6-6. ASSERTION OF S_PERR_L ..... 61
TABLE 6-7. ASSERTION OF P_SERR_L FOR DATA PARITY ERRORS ..... 61
TABLE 10-1. GPIO OPERATION ..... 71
TABLE 10-2. GPIO SERIAL DATA FORMAT ..... 71
TABLE 11-1. POWER MANAGEMENT TRANSITIONS ..... 73
TABLE 16-1. TAP PINS ..... 101
TABLE 16-2. JTAG BOUNDARY REGISTER ORDER ..... 103

# LIST OF FIGURES

FIGURE 8-1 SECONDARY ARBITER EXAMPLE ..... 67
FIGURE 16-1 TEST ACCESS PORT BLOCK DIAGRAM ..... 100
FIGURE 17-1 PCI SIGNAL TIMING MEASUREMENT CONDITIONS ..... 107
FIGURE 18-1 208-PIN FQFP PACKAGE OUTLINE ..... 109
FIGURE 18-2 256-PIN PBGA PACKAGE OUTLINE ..... 110

*This page intentionally left blank.*

# 1 INTRODUCTION

## Product Description

The PI7C8150A is an enhanced PCI-to-PCI Bridge is designed to be fully compliant with the *PCI Local Bus Specification* Revision 2.3. Both the primary and secondary interfaces are specified to run at 32-bits and up to 66MHz (33MHz for PI7C8150A-33).

## Product Features

- 32-bit Primary and Secondary Ports run up to 66MHz (33MHz for PI7C8150A-33)
- Compliant with the *PCI Local Bus Specification, Revision 2.3*
- 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 *Advanced Configuration Power Interface (ACPI) Specification.*
- Compliant with the *PCI Power Management Specification, Revision 1.0.*
- Synchronous operation support
- Provides internal arbitration for one set of nine secondary bus masters
  - Programmable 2-level priority arbiter
  - Disable control for use of external arbiter
- Supports posted write buffers in all directions
- Four 128 byte FIFO's for delay transactions
- Two 128 byte FIFO's for posted memory transactions
- Enhanced address decoding
- Extended Commercial temperature range 0°C to 85°C
- IEEE 1149.1 JTAG interface support
- 3.3V core; 3.3V and 5V signaling
- 208-pin FQFP and 256-pin PBGA
  - Pb-free & Green available

# 2 SIGNAL DEFINITIONS

## 2.1 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 |

## 2.2 Signals

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

### 2.2.1 PRIMARY BUS INTERFACE SIGNALS

| Name | Pin # | Pin # | Type | Description |
|---|---|---|---|---|
| P_AD[31:0] | 49, 50, 55, 57, 58, 60, 61, 63, 67, 68, 70, 71, 73, 74, 76, 77, 93, 95, 96, 98, 99, 101, 107, 109, 112, 113, 115, 116, 118, 119, 121, 122 | N3, T2, T4, N5, P5, T5, N6, R5, T6, P7, T7, R7, T8, P8, R8, T9, R12, P12, T14, R13, N12, T15, P16, N15, M14, M13, M15, L13, M16, L14, L15, L16 | TS | **Primary Address / Data:** Multiplexed address and data bus. Address is indicated by P_FRAME_L assertion. Write data is stable and valid when P_IRDY_L is asserted and read data is stable and valid when P_TRDY_L is asserted. Data is transferred on rising clock edges when both P_IRDY_L and P_TRDY_L are asserted. During bus idle, PI7C8150A drives P_AD to a valid logic level when P_GNT_L is asserted. |
| P_CBE[3:0] | 64, 79, 92, 110 | R6, R9, T13, N16 | 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, PI7C8150A drives P_CBE[3:0] to a valid logic level when P_GNT_L is asserted. |
| P_PAR | 90 | N11 | TS | **Primary Parity.** Parity is even across P_AD[31:0], P_CBE[3:0], and P_PAR (i.e. an even number of 1's). P_PAR is an input and is valid and stable one cycle after the address phase (indicated by assertion of P_FRAME_L) for address parity. For write data phases, P_PAR is an input and is valid one clock after P_IRDY_L is asserted. For read data phase, P_PAR is an output and is valid one clock after P_TRDY_L is asserted. Signal P_PAR is tri-stated one cycle after the P_AD lines are tri-stated. During bus idle, PI7C8150A drives P_PAR to a valid logic level when P_GNT_L is asserted. |

| Name | Pin # | Pin # | Type | Description |
|---|---|---|---|---|
| P_FRAME_L | 80 | P9 | 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_L indicates the final data phase requested by the initiator. Before being tri-stated, it is driven to a de-asserted state for one cycle. |
| P_IRDY_L | 82 | T10 | 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_L | 83 | R10 | 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_L | 84 | P10 | STS | **Primary Device Select (Active LOW).** Asserted by the target indicating that the device is accepting the transaction. As a master, PI7C8150A waits for the assertion of this signal within 5 cycles of P_FRAME_L assertion; otherwise, terminate with master abort. Before tri-stated, it is driven to a de-asserted state for one cycle. |
| P_STOP_L | 85 | T11 | 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_L | 87 | R11 | STS | **Primary LOCK (Active LOW).** Asserted by the master for multiple transactions to complete. |
| P_IDSEL | 65 | P6 | I | **Primary ID Select.** Used as a chip select line for Type 0 configuration access to PI7C8150A configuration space. |
| P_PERR_L | 88 | T12 | 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_L | 89 | P11 | OD | **Primary System Error (Active LOW).** Can be driven LOW by any device to indicate a system error condition. PI7C8150A drives this pin on:<br>▪ Address parity error<br>▪ Posted write data parity error on target bus<br>▪ Secondary S_SERR_L asserted<br>▪ Master abort during posted write transaction<br>▪ Target abort during posted write transaction<br>▪ Posted write transaction discarded<br>▪ Delayed write request discarded<br>▪ Delayed read request discarded<br>▪ Delayed transaction master timeout<br>This signal requires an external pull-up resistor for proper operation. |
| P_REQ_L | 47 | P2 | TS | **Primary Request (Active LOW):** This is asserted by PI7C8150A to indicate that it wants to start a transaction on the primary bus. PI7C8150A de-asserts this pin for at least 2 PCI clock cycles before asserting it again. |
| P_GNT_L | 46 | R1 | I | **Primary Grant (Active LOW):** When asserted, PI7C8150A can access the primary bus. During idle and P_GNT_L asserted, PI7C8150A will drive P_AD, P_CBE, and P_PAR to valid logic levels. |
| P_RESET_L | 43 | P1 | I | **Primary RESET (Active LOW):** When P_RESET_L is active, all PCI signals should be asynchronously tri-stated. |

| Name | Pin # | Pin # | Type | Description |
|---|---|---|---|---|
| P_M66EN | 102 | R14 | I | **Primary Interface 66MHz Operation.** This input is used to specify if PI7C8150A 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. |

## 2.2.2 SECONDARY BUS INTERFACE SIGNALS

| Name | Pin # | Pin # | Type | Description |
|---|---|---|---|---|
| S_AD[31:0] | 206, 204, 203, 201, 200, 198, 197, 195, 192, 191, 189, 188, 186, 185, 183, 182, 165, 164, 162, 161, 159, 154, 152, 150, 147, 146, 144, 143, 141, 140, 138, 137 | A4, D5, C5, A5, B5, D6, A6, C6, C7, A7, B7, C8, A8, B8, A9, C9, C12, D12, A14, B13, A15, B16, E13, C16, E14, D16, F13, E16, F14, F15, F16, G16 | TS | **Secondary Address/Data:** Multiplexed address and data bus. Address is indicated by S_FRAME_L assertion. Write data is stable and valid when S_IRDY_L is asserted and read data is stable and valid when S_IRDY_L is asserted. Data is transferred on rising clock edges when both S_IRDY_L and S_TRDY_L are asserted. During bus idle, PI7C8150A drives S_AD to a valid logic level when S_GNT_L is asserted respectively. |
| S_CBE[3:0] | 194, 180, 167, 149 | B6, B9, B12, E15 | 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, PI7C8150A drives S_CBE[3:0] to a valid logic level when the internal grant is asserted. |
| S_PAR | 168 | A13 | TS | **Secondary Parity:** Parity is even across S_AD[31:0], S_CBE[3:0], and S_PAR (i.e. an even number of 1's). S_PAR is an input and is valid and stable one cycle after the address phase (indicated by assertion of S_FRAME_L) for address parity. For write data phases, S_PAR is an input and is valid one clock after S_IRDY_L is asserted. For read data phase, S_PAR is an output and is valid one clock after S_TRDY_L is asserted. Signal S_PAR is tri-stated one cycle after the S_AD lines are tri-stated. During bus idle, PI7C8150A drives S_PAR to a valid logic level when the internal grant is asserted. |
| S_FRAME_L | 179 | A10 | 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_L 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_L | 177 | B10 | 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. |
| S_TRDY_L | 176 | C10 | 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. |

| Name | Pin # | Pin # | Type | Description |
|---|---|---|---|---|
| S_DEVSEL_L | 175 | A11 | STS | **Secondary Device Select (Active LOW):** Asserted by the target indicating that the device is accepting the transaction. As a master, PI7C8150A waits for the assertion of this signal within 5 cycles of S_FRAME_L assertion; otherwise, terminate with master abort. Before tri-stated, it is driven to a de-asserted state for one cycle. |
| S_STOP_L | 173 | B11 | 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_L | 172 | C11 | STS | **Secondary LOCK (Active LOW):** Asserted by the master for multiple transactions to complete. |
| S_PERR_L | 171 | A12 | 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_L | 169 | D11 | I | **Secondary System Error (Active LOW):** Can be driven LOW by any device to indicate a system error condition. |
| S_REQ_L[8:0] | 9, 8, 7, 6, 5, 4, 3, 2, 207 | E4, E3, D2, C1, C2, D3, A2,B3, B4 | 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_L[8:0] | 19, 18, 17, 16, 15, 14, 13, 11, 10 | G1, F1, F2, G3, F4, E1, E2,F3, D1 | TS | **Secondary Grant (Active LOW):** PI7C8150A asserts this pin to access the secondary bus. PI7C8150A de-asserts this pin for at least 2 PCI clock cycles before asserting it again. During idle and S_GNT_L asserted, PI7C8150A will drive S_AD, S_CBE, and S_PAR. |
| S_RESET_L | 22 | H1 | O | **Secondary RESET (Active LOW):** Asserted when any of the following conditions are met:<br>1. Signal P_RESET_L is asserted.<br>2. Secondary reset bit in bridge control register in configuration space is set.<br>When asserted, all control signals are tri-stated and zeroes are driven on S_AD, S_CBE, and S_PAR. |
| S_M66EN | 153 | D15 | I/OD | **Secondary Interface 66MHz Operation:**<br>This input is used to specify if PI7C8150A is 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.<br><br>If P_M66EN is pulled LOW, the S_M66EN is also driven LOW. |
| S_CFN_L | 23 | H2 | 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_L[0] is reconfigured to be the secondary bus grant input, and S_GNT_L[0] is reconfigured to be the secondary bus request output. S_CFN_L has a weak internal pull-down resistor. |

## 2.2.3 CLOCK SIGNALS

| Name | Pin # | Pin # | Type | Description |
|---|---|---|---|---|
| P_CLK | 45 | M4 | I | **Primary Clock Input:** Provides timing for all transactions on the primary interface. |
| S_CLKIN | 21 | H3 | I | **Secondary Clock Input:** Provides timing for all transactions on the secondary interface. |

| Name | Pin # | Pin # | Type | Description |
|---|---|---|---|---|
| S_CLKOUT[9:0] | 42, 41, 39, 38, 36, 35, 33, 32, 30, 29 | M3, M2, N1, L4, L3, M1, L2, L1, K3, K2 | O | **Secondary Clock Output:** Provides secondary clocks phase synchronous with the P_CLK in synchronous mode.<br><br>When these clocks are used, one of the clock outputs must be fed back to S_CLKIN. Unused outputs may be disabled by:<br>1. Writing the secondary clock disable bits in the configuration space<br>2. Using the serial disable mask using the GPIO pins and MSK_IN<br>3. Terminating them electrically. |

## 2.2.4 MISCELLANEOUS SIGNALS

| Name | Pin # | Pin # | Type | Description |
|---|---|---|---|---|
| MSK_IN | 126 | K15 | I | **MSK_IN - Secondary Clock Disable Serial Input (synchronous mode):** This pin is used by PI7C8150A to disable secondary clock outputs. The serial stream is received by MSK_IN, starting when P_RESET is detected deasserted and S_RESET_L 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 | 124 | K14 | 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 | 135 | G14 | 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 | 44 | N2 | I | **Bus/Power Clock Control Management Pin:** When this pin is tied HIGH and the PI7C8150A is placed in the $D3_{HOT}$ power state, it enables the PI7C8150A 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 PI7C8150A enters the $D3_{HOT}$ power state. |
| CFG66 / SCAN_EN_H | 125 | K16 | I | This is a multiplexed pin that has 2 functions.<br><br>**CFG66 - 66MHz Configuration:** This pin is used to designate 66MHz operation. Tie HIGH to enable 66MHz operation or tie LOW to designate 33MHz operation.<br><br>**SCAN_EN_H - Full-Scan Enable Control (synchronous mode):** When SCAN_EN_H is LOW, full-scan is in shift operation. When SCAN_EN_H is HIGH, full-scan is in parallel operation. *Note: Valid only in test mode. Pin is CFG66 in normal operation.* |

| | | | | |
|---|---|---|---|---|
| MS0, MS1 | 155, 106 | B14, R16 | I | **Mode Selection:** Reserved for future features.<br><br>MS0 / MS1 / Description<br>0 / 0 / RESERVED<br>0 / 1 / RESERVED<br>1 / 0 / Normal operation<br>1 / 1 / RESERVED<br><br>MS0 should be set to 1 and MS1 should be set to 0 for normal operation. |

## 2.2.5 GENERAL PURPOSE I/O INTERFACE SIGNALS

| Name | Pin # | Pin # | Type | Description |
|---|---|---|---|---|
| GPIO[3:0] | 24, 25, 27, 28 | J3, J2, J1, K1 | 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. |

## 2.2.6 JTAG BOUNDARY SCAN SIGNALS

| Name | Pin # | Pin # | Type | Description |
|---|---|---|---|---|
| TCK | 133 | H15 | I | **Test Clock.** Used to clock state information and data into and out of the PI7C8150A during boundary scan. |
| TMS | 132 | H14 | I | **Test Mode Select.** Used to control the state of the Test Access Port controller. |
| TDO | 130 | H16 | O | **Test Data Output.** When SCAN_EN_H is HIGH, it is used (in conjunction with TCK) to shift data out of the Test Access Port (TAP) in a serial bit stream. |
| TDI | 129 | J15 | I | **Test Data Input.** When SCAN_EN_H is HIGH, it is used (in conjunction with TCK) to shift data and instructions into the Test Access Port (TAP) in a serial bit stream. |
| TRST_L | 134 | G15 | I | **Test Reset.** Active LOW signal to reset the Test Access Port (TAP) controller into an initialized state. |

## 2.2.7 POWER AND GROUND

| Name | Pin # | Pin # | Type | Description |
|---|---|---|---|---|
| VDD | 1, 26, 34, 40, 51, 53, 56, 62, 69, 75, 81, 91, 97, 103, 105, 108, 114, 120, 131, 139, 145, 151, 157, 163, 170, 178, 184, 190, 196, 202, 208 | A3, C4, C15, D7, D8, D9, D10, E6, E7, E8, E9, E10, E11, F5, F12, G4, G5, G12, G13, H4, H5, H12, H13, J4, J5, J12, J13, K4, K5, K12, K13, L5, L12, M6, M7, M8, M9, M10, M11, N7, N8, N9, N10, P13, P15, R3, T3 | P | **Power:** +3.3V Digital power. |
| VSS | 12, 20, 31, 37, 48, 52, 54, 59, 66, 72, | A1, A16, B1, B2, B15, C3, | P | **Ground:** Digital ground. |

| Name | Pin # | Pin # | Type | Description |
|---|---|---|---|---|
| | 78, 86, 94, 100, 104, 111, 117, 123, 136, 142, 148, 156, 158, 160, 166, 174, 181, 187, 193, 199, 205 | C13, C14, D4, D13, D14, E5, E12, F6, F7, F8, F9, F10, F11, G2, G6, G7, G8, G9, G10, G11, H6, H7, H8, H9, H10, H11, J6, J7, J8, J9, J10, J11, K6, K7, K8, K9, K10, K11, L6, L7, L8, L9, L10, L11, M5, M12, N4, N13, N14, P3, P4, P14, R2, R4, R15, T1, T16 | | |

## 2.3 PIN LIST – 208-PIN FQFP

**Table 2-1. Pin List – 208-pin FQFP**

| Pin Number | Name | Type | Pin Number | Name | Type |
|---|---|---|---|---|---|
| 1 | VDD | P | 2 | S_REQ_L[1] | I |
| 3 | S_REQ_L[2] | I | 4 | S_REQ_L[3] | I |
| 5 | S_REQ_L[4] | I | 6 | S_REQ_L[5] | I |
| 7 | S_REQ_L[6] | I | 8 | S_REQ_L[7] | I |
| 9 | S_REQ_L[8] | I | 10 | S_GNT_L[0] | TS |
| 11 | S_GNT_L[1] | TS | 12 | VSS | P |
| 13 | S_GNT_L[2] | TS | 14 | S_GNT_L[3] | TS |
| 15 | S_GNT_L[4] | TS | 16 | S_GNT_L[5] | TS |
| 17 | S_GNT_L[6] | TS | 18 | S_GNT_L[7] | TS |
| 19 | S_GNT_L[8] | TS | 20 | VSS | P |
| 21 | S_CLKIN | I | 22 | S_RESET_L | O |
| 23 | S_CFN_L | I | 24 | GPIO[3] | TS |
| 25 | GPIO[2] | TS | 26 | VDD | P |
| 27 | GPIO[1] | TS | 28 | GPIO[0] | TS |
| 29 | S_CLKOUT[0] | O | 30 | S_CLKOUT[1] | O |
| 31 | VSS | P | 32 | S_CLKOUT[2] | O |
| 33 | S_CLKOUT[3] | O | 34 | VDD | P |
| 35 | S_CLKOUT[4] | O | 36 | S_CLKOUT[5] | O |
| 37 | VSS | P | 38 | S_CLKOUT[6] | O |
| 39 | S_CLKOUT[7] | O | 40 | VDD | P |
| 41 | S_CLKOUT[8] | O | 42 | S_CLKOUT[9] | O |
| 43 | P_RESET_L | I | 44 | BPCCE | I |
| 45 | P_CLK | I | 46 | P_GNT_L | I |
| 47 | P_REQ_L | TS | 48 | VSS | P |
| 49 | P_AD[31] | TS | 50 | P_AD[30] | TS |
| 51 | VDD | P | 52 | VSS | P |
| 53 | VDD | P | 54 | VSS | P |
| 55 | P_AD[29] | TS | 56 | VDD | P |
| 57 | P_AD[28] | TS | 58 | P_AD[27] | TS |
| 59 | VSS | P | 60 | P_AD[26] | TS |
| 61 | P_AD[25] | TS | 62 | VDD | P |
| 63 | P_AD[24] | TS | 64 | P_CBE[3] | TS |
| 65 | P_IDSEL | I | 66 | VSS | P |

| Pin Number | Name | Type | Pin Number | Name | Type |
|---|---|---|---|---|---|
| 67 | P_AD[23] | TS | 68 | P_AD[22] | TS |
| 69 | VDD | P | 70 | P_AD[21] | TS |
| 71 | P_AD[20] | TS | 72 | VSS | P |
| 73 | P_AD[19] | TS | 74 | P_AD[18] | TS |
| 75 | VDD | P | 76 | P_AD[17] | TS |
| 77 | P_AD[16] | TS | 78 | VSS | P |
| 79 | P_CBE[2] | TS | 80 | P_FRAME_L | STS |
| 81 | VDD | P | 82 | P_IRDY_L | STS |
| 83 | P_TRDY_L | STS | 84 | P_DEVSEL_L | STS |
| 85 | P_STOP_L | STS | 86 | VSS | P |
| 87 | P_LOCK_L | STS | 88 | P_PERR_L | STS |
| 89 | P_SERR_L | STS | 90 | P_PAR | STS |
| 91 | VDD | P | 92 | P_CBE[1] | TS |
| 93 | P_AD[15] | TS | 94 | VSS | P |
| 95 | P_AD[14] | TS | 96 | P_AD[13] | TS |
| 97 | VDD | P | 98 | P_AD[12] | TS |
| 99 | P_AD[11] | TS | 100 | VSS | P |
| 101 | P_AD[10] | TS | 102 | P_M66EN | I |
| 103 | VDD | P | 104 | VSS | P |
| 105 | VDD | P | 106 | MS1 | I |
| 107 | P_AD[9] | TS | 108 | VDD | P |
| 109 | P_AD[8] | TS | 110 | P_CBE[0] | TS |
| 111 | VSS | P | 112 | P_AD[7] | TS |
| 113 | P_AD[6] | TS | 114 | VDD | P |
| 115 | P_AD[5] | TS | 116 | P_AD[4] | TS |
| 117 | VSS | P | 118 | P_AD[3] | TS |
| 119 | P_AD[2] | TS | 120 | VDD | P |
| 121 | P_AD[1] | TS | 122 | P_AD[0] | TS |
| 123 | VSS | P | 124 | P_VIO | I |
| 125 | CFG66 / SCAN_EN_H | I | 126 | MSK_IN | I |
| 127 | RESERVED | - | 128 | RESERVED | - |
| 129 | TDI | I | 130 | TDO | O |
| 131 | VDD | P | 132 | TMS | I |
| 133 | TCK | I | 134 | TRST_L | I |
| 135 | S_VIO | I | 136 | VSS | P |
| 137 | S_AD[0] | TS | 138 | S_AD[1] | TS |
| 139 | VDD | P | 140 | S_AD[2] | TS |
| 141 | S_AD[3] | TS | 142 | VSS | P |
| 143 | S_ADD[4] | TS | 144 | S_AD[5] | TS |
| 145 | VDD | P | 146 | S_AD[6] | TS |
| 147 | S_AD[7] | TS | 148 | VSS | P |
| 149 | S_CBE[0] | TS | 150 | S_AD[8] | TS |
| 151 | VDD | P | 152 | S_AD[9] | TS |
| 153 | S_M66EN | I/OD | 154 | S_AD[10] | TS |
| 155 | MS0 | I | 156 | VSS | P |
| 157 | VDD | P | 158 | VSS | P |
| 159 | S_AD[11] | TS | 160 | VSS | P |
| 161 | S_AD[12] | TS | 162 | S_AD[13] | TS |
| 163 | VDD | P | 164 | S_AD[14] | TS |
| 165 | S_AD[15] | TS | 166 | VSS | P |
| 167 | S_CBE[1] | TS | 168 | S_PAR | TS |
| 169 | S_SERR_L | I | 170 | VDD | P |
| 171 | S_PERR_L | STS | 172 | S_LOCK_L | STS |
| 173 | S_STOP_L | STS | 174 | VSS | P |
| 175 | S_DEVSEL_L | STS | 176 | S_TRDY_L | STS |
| 177 | S_IRDY_L | STS | 178 | VDD | P |
| 179 | S_FRAME_L | STS | 180 | S_CBE[2] | TS |
| 181 | VSS | P | 182 | S_AD[16] | TS |
| 183 | S_AD[17] | TS | 184 | VDD | P |
| 185 | S_AD[18] | TS | 186 | S_AD[19] | TS |
| 187 | VSS | P | 188 | S_AD[20] | TS |
| 189 | S_AD[21] | TS | 190 | VDD | P |

| Pin Number | Name | Type | Pin Number | Name | Type |
|---|---|---|---|---|---|
| 191 | S_AD[22] | TS | 192 | S_AD[23] | TS |
| 193 | VSS | P | 194 | S_CBE[3] | TS |
| 195 | S_AD[24] | TS | 196 | VDD | P |
| 197 | S_AD[25] | TS | 198 | S_AD[26] | TS |
| 199 | VSS | P | 200 | S_AD[27] | TS |
| 201 | S_AD[28] | TS | 202 | VDD | P |
| 203 | S_AD[29] | TS | 204 | S_AD[30] | TS |
| 205 | VSS | P | 206 | S_AD[31] | TS |
| 207 | S_REQ_L[0] | I | 208 | VDD | P |

## 2.4 PIN LIST – 256-BALL PBGA

**Table 2-2. Pin List – 256-pin PBGA**

| Pin Number | Name | Type | Pin Number | Name | Type | Pin Number | Name | Type |
|---|---|---|---|---|---|---|---|---|
| A1 | VSS | P | A2 | S_REQ_L[2] | I | A3 | VDD | P |
| A4 | S_AD[31] | TS | A5 | S_AD[28] | TS | A6 | S_AD[25] | TS |
| A7 | S_AD[22] | TS | A8 | S_AD[19] | TS | A9 | S_AD[17] | TS |
| A10 | S_FRAME_L | STS | A11 | S_DEVSEL_L | STS | A12 | S_PERR_L | STS |
| A13 | S_PAR | TS | A14 | S_AD[13] | TS | A15 | S_AD[11] | TS |
| A16 | VSS | P | B1 | VSS | P | B2 | VSS | P |
| B3 | S_REQ_L[1] | I | B4 | S_REQ_L[0] | I | B5 | S_AD[27] | TS |
| B6 | S_CBE_L[3] | TS | B7 | S_AD[21] | TS | B8 | S_AD[18] | TS |
| B9 | S_CBE_L[2] | TS | B10 | S_IRDY_L | STS | B11 | S_STOP_L | STS |
| B12 | S_CBE_L[1] | TS | B13 | S_AD[12] | TS | B14 | MS0 | P |
| B15 | VSS | P | B16 | S_AD[10] | TS | C1 | S_REQ_L[5] | I |
| C2 | S_REQ_L[4] | I | C3 | VSS | P | C4 | VDD | P |
| C5 | S_AD[29] | TS | C6 | S_AD[24] | TS | C7 | S_AD[23] | TS |
| C8 | S_AD[20] | TS | C9 | S_AD[16] | TS | C10 | S_TRDY_L | STS |
| C11 | S_LOCK_L | STS | C12 | S_AD[15] | TS | C13 | VSS | P |
| C14 | VSS | P | C15 | VDD | P | C16 | S_AD[8] | TS |
| D1 | S_GNT_L[0] | TS | D2 | S_REQ_L[6] | I | D3 | S_REQ_L[3] | I |
| D4 | VSS | P | D5 | S_AD[30] | TS | D6 | S_AD[26] | TS |
| D7 | VDD | P | D8 | VDD | P | D9 | VDD | P |
| D10 | VDD | P | D11 | S_SERR_L | I | D12 | S_AD[14] | TS |
| D13 | VSS | P | D14 | VSS | P | D15 | S_M66EN | I/OD |
| D16 | S_AD[6] | TS | E1 | S_GNT_L[3] | TS | E2 | S_GNT_L[2] | TS |
| E3 | S_REQ_L[7] | I | E4 | S_REQ_L[8] | I | E5 | VSS | P |
| E6 | VDD | P | E7 | VDD | P | E8 | VDD | P |
| E9 | VDD | P | E10 | VDD | P | E11 | VDD | P |
| E12 | VSS | P | E13 | S_AD[9] | TS | E14 | S_AD[7] | TS |
| E15 | S_CBE_L[0] | TS | E16 | S_AD[4] | TS | F1 | S_GNT_L[7] | TS |
| F2 | S_GNT_L[6] | TS | F3 | S_GNT_L[1] | TS | F4 | S_GNT_L[4] | TS |
| F5 | VDD | P | F6 | VSS | P | F7 | VSS | P |
| F8 | VSS | P | F9 | VSS | P | F10 | VSS | P |
| F11 | VSS | P | F12 | VDD | P | F13 | S_AD[5] | TS |
| F14 | S_AD[3] | TS | F15 | S_AD[2] | TS | F16 | S_AD[1] | TS |
| G1 | S_GNT_L[8] | TS | G2 | VSS | P | G3 | S_GNT_L[5] | TS |
| G4 | VDD | P | G5 | VDD | P | G6 | VSS | P |
| G7 | VSS | P | G8 | VSS | P | G9 | VSS | P |
| G10 | VSS | P | G11 | VSS | P | G12 | VDD | P |
| G13 | VDD | P | G14 | S_VIO | I | G15 | TRST_L | I |
| G16 | S_AD[0] | TS | H1 | S_RESET_L | O | H2 | S_CFN_L | I |
| H3 | S_CLKIN | I | H4 | VDD | P | H5 | VDD | P |
| H6 | VSS | P | H7 | VSS | P | H8 | VSS | P |
| H9 | VSS | P | H10 | VSS | P | H11 | VSS | P |
| H12 | VDD | P | H13 | VDD | P | H14 | TMS | I |
| H15 | TCK | I | H16 | TDO | O | J1 | GPIO[1] | TS |

| Pin Number | Name | Type | Pin Number | Name | Type | Pin Number | Name | Type |
|---|---|---|---|---|---|---|---|---|
| J2 | GPIO[2] | TS | J3 | GPIO[3] | TS | J4 | VDD | P |
| J5 | VDD | P | J6 | VSS | P | J7 | VSS | P |
| J8 | VSS | P | J9 | VSS | P | J10 | VSS | P |
| J11 | VSS | P | J12 | VDD | P | J13 | VDD | P |
| J14 | RESERVED | - | J15 | TDI | I | J16 | RESERVED | - |
| K1 | GPIO[0] | TS | K2 | S_CLKOUT[0] | O | K3 | S_CLKOUT[1] | O |
| K4 | VDD | P | K5 | VDD | P | K6 | VSS | P |
| K7 | VSS | P | K8 | VSS | P | K9 | VSS | P |
| K10 | VSS | P | K11 | VSS | P | K12 | VDD | P |
| K13 | VDD | P | K14 | P_VIO | I | K15 | MSK_IN | I |
| K16 | CFG66 SCAN_EN_H | I | L1 | S_CLKOUT[2] | O | L2 | S_CLKOUT[3] | O |
| L3 | S_CLKOUT[5] | O | L4 | S_CLKOUT[6] | O | L5 | VDD | P |
| L6 | VSS | P | L7 | VSS | P | L8 | VSS | P |
| L9 | VSS | P | | | | | | |
| L10 | VSS | P | L11 | VSS | P | L12 | VDD | P |
| L13 | P_AD[4] | TS | L14 | P_AD[2] | TS | L15 | P_AD[1] | TS |
| L16 | P_AD[0] | TS | M1 | S_CLKOUT[4] | O | M2 | S_CLKOUT[8] | O |
| M3 | S_CLKOUT[9] | O | M4 | P_CLK | I | M5 | VSS | P |
| M6 | VDD | P | M7 | VDD | P | M8 | VDD | P |
| M9 | VDD | P | M10 | VDD | P | M11 | VDD | P |
| M12 | VSS | P | M13 | P_AD[6] | TS | M14 | P_AD[7] | TS |
| M15 | P_AD[5] | TS | M16 | P_AD[3] | TS | N1 | S_CLKOUT[7] | O |
| N2 | BPCCE | I | N3 | P_AD[31] | TS | N4 | VSS | P |
| N5 | P_AD[28] | TS | N6 | P_AD[25] | TS | N7 | VDD | P |
| N8 | VDD | P | N9 | VDD | P | N10 | VDD | P |
| N11 | P_PAR | TS | N12 | P_AD[11] | TS | N13 | VSS | P |
| N14 | VSS | P | N15 | P_AD[8] | TS | N16 | P_CBE_L[0] | TS |
| P1 | P_RESET_L | I | P2 | P_REQ_L | TS | P3 | VSS | P |
| P4 | VSS | P | P5 | P_AD[27] | TS | P6 | P_IDSEL | I |
| P7 | P_AD[22] | TS | P8 | P_AD[18] | TS | P9 | P_FRAME_L | STS |
| P10 | P_DEVSEL_L | STS | P11 | P_SERR_L | OD | P12 | P_AD[14] | TS |
| P13 | VDD | P | P14 | VSS | P | P15 | VDD | P |
| P16 | P_AD[9] | TS | R1 | P_GNT_L | I | R2 | VSS | P |
| R3 | VDD | P | R4 | VSS | P | R5 | P_AD[24] | TS |
| R6 | P_CBE_L[3] | TS | R7 | P_AD[20] | TS | R8 | P_AD[17] | TS |
| R9 | P_CBE_L[2] | TS | R10 | P_TRDY_L | STS | R11 | P_LOCK_L | STS |
| R12 | P_AD[15] | TS | R13 | P_AD[12] | TS | R14 | P_M66EN | I |
| R15 | VSS | P | R16 | MS1 | P | T1 | VSS | P |
| T2 | P_AD[30] | TS | T3 | VDD | P | T4 | P_AD[29] | TS |
| T5 | P_AD[26] | TS | T6 | P_AD[23] | TS | T7 | P_AD[21] | TS |
| T8 | P_AD[19] | TS | T9 | P_AD[16] | TS | T10 | P_IRDY_L | STS |
| T11 | P_STOP_L | STS | T12 | P_PERR_L | STS | T13 | P_CBE_L[1] | TS |
| T14 | P_AD[13] | TS | T15 | P_AD[10] | TS | T16 | VSS | P |

# 3 PCI BUS OPERATION

This Chapter offers information about PCI transactions, transaction forwarding across PI7C8150A, and transaction termination. The PI7C8150A has two 128-byte FIFO's for buffering of upstream and downstream transactions. These hold addresses, data, commands, and byte enables that are used for write transactions. The PI7C8150A also has an additional four 128-byte FIFO's that hold addresses, data, commands, and byte enables for read transactions.

## 3.1 TYPES OF TRANSACTIONS

This section provides a summary of PCI transactions performed by PI7C8150A. Table 3-1 lists the command code and name of each PCI transaction. The Master and Target columns indicate support for each transaction when PI7C8150A initiates transactions as a master, on the primary (P) and secondary (S) buses, and when PI7C8150A responds to transactions as a target, on the primary (P) and secondary (S) buses.

**Table 3-1. 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 3-1, the following PCI commands are not supported by PI7C8150A:

- PI7C8150A never initiates a PCI transaction with a reserved command code and, as a target, PI7C8150A ignores reserved command codes.
- PI7C8150A does not generate interrupt acknowledge transactions. PI7C8150A ignores interrupt acknowledge transactions as a target.
- PI7C8150A does not respond to special cycle transactions. PI7C8150A 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.
- PI7C8150A neither generates Type 0 configuration transactions on the primary PCI bus nor responds to Type 0 configuration transactions on the secondary PCI buses.

## 3.2 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]. PI7C8150A 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 non-zero, PI7C8150A automatically disconnects the transaction after the first data transfer.

## 3.3 DEVICE SELECT (DEVSEL_L) GENERATION

PI7C8150A always performs positive address decoding (medium decode) when accepting transactions on either the primary or secondary buses. PI7C8150A never does subtractive decode.

## 3.4 DATA PHASE

The address phase of a PCI transaction is followed by one or more data phases. A data phase is completed when IRDY_L and either TRDY_L or STOP_L are asserted. A transfer of data occurs only when both IRDY_L and TRDY_L are asserted during the same PCI clock cycle. The last data phase of a transaction is indicated when FRAME_L is de-asserted and both TRDY_L and IRDY_L are asserted, or when IRDY_L and STOP_L are asserted. See Section 3.8 for further discussion of transaction termination.

Depending on the command type, PI7C8150A can support multiple data phase PCI transactions. For detailed descriptions of how PI7C8150A imposes disconnect boundaries, see Section 3.5.4 for write address boundaries and Section 3.6.3 read address boundaries.

## 3.5 WRITE TRANSACTIONS

Write transactions are treated as either posted write or delayed write transactions. Table 3-2 shows the method of forwarding used for each type of write operation.

**Table 3-2. 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 |

### 3.5.1 MEMORY WRITE TRANSACTIONS

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

When PI7C8150A determines that a memory write transaction is to be forwarded across the bridge, PI7C8150A asserts DEVSEL_L with medium timing and TRDY_L 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, PI7C8150A accepts write data without obtaining access to the target bus.

The PI7C8150A 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 PI7C8150A 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 PI7C8150A 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, PI7C8150A asserts its request on the target bus. This can occur while PI7C8150A 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, PI7C8150A asserts FRAME_L and drives the stored write address out on the target bus. On the following cycle, PI7C8150A 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, PI7C8150A can drive one DWORD of write data each PCI clock cycle; that is, no master wait states are inserted. If write data is flowing through PI7C8150A and the initiator stalls, PI7C8150A will signal the last data phase for the current transaction at the target bus if the queue empties. PI7C8150A will restart the follow-on transactions if the queue has new data.

PI7C8150A 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 (PI7C8150A starts another transaction to deliver the rest of the write data).
- The target returns a target abort (PI7C8150A discards remaining write data).
- The master latency timer expires, and PI7C8150A no longer has the target bus grant (PI7C8150A starts another transaction to deliver remaining write data).

Section 3.8.3.2 provides detailed information about how PI7C8150A responds to target termination during posted write transactions.

### 3.5.2 MEMORY WRITE AND INVALIDATE

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

If offset 74h bits [8:7] = 11, the PI7C8150A 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 offset 74h bits [8:7] = 00, the PI7C8150A converts Memory Write and Invalidate transactions to Memory Write transactions at the destination.

If the value in the cache line size register does meet the memory write and invalidate conditions, the PI7C8150A returns a target disconnect to the initiator on a cache line boundary.

### 3.5.3 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 PI7C8150A forwards it as a delayed transaction, PI7C8150A claims the access by asserting DEVSEL_L and returns a target retry to the initiator. During the address phase, PI7C8150A samples the bus command, address, and address parity one cycle later. After IRDY_L is asserted, PI7C8150A 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 PI7C8150A initiates the transaction on the target bus. PI7C8150A transfers the write data to the target. If PI7C8150A 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 PI7C8150A is unable to deliver write data after $2^{24}$ (default) or $2^{32}$ (maximum) attempts, PI7C8150A will report a system error. PI7C8150A also asserts P_SERR_L if the primary SERR_L enable bit is set in the command register. See Section 6.4 for information on the assertion of P_SERR_L. 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 PI7C8150A claims the access by asserting DEVSEL_L and returns TRDY_L to the initiator, to indicate that the write data was transferred. If the initiator requests multiple DWORD, PI7C8150A also asserts STOP_L in conjunction with TRDY_L 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, PI7C8150A returns a target retry to the initiator. PI7C8150A 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, PI7C8150A does not make a new entry into the delayed transaction queue. Section 3.8.3.1 provides detailed information about how PI7C8150A responds to target termination during delayed write transactions.

PI7C8150A 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, PI7C8150A discards the delayed write completion from the delayed transaction completion queue. PI7C8150A also conditionally asserts P_SERR_L (see Section 6.4).

### 3.5.4 WRITE TRANSACTION ADDRESS BOUNDARIES

PI7C8150A imposes internal address boundaries when accepting write data. The aligned address boundaries are used to prevent PI7C8150A from continuing a transaction over a device address boundary and to provide an upper limit on maximum latency. PI7C78150A returns a target disconnect to the initiator when it reaches the aligned address boundaries under conditions shown in Table 3-3.

**Table 3-3. 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, 16 | Cache line boundary if posted memory write data FIFO does not have enough space for the cache line |

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

### 3.5.5 BUFFERING MULTIPLE WRITE TRANSACTIONS

PI7C8150A 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, PI7C8150A returns a target disconnect to the initiator.

Delayed write transactions are posted 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 6 for information about how multiple posted and delayed write transactions are ordered.

### 3.5.6 FAST BACK-TO-BACK TRANSACTIONS

PI7C8150A can recognize and post fast back-to-back write transactions. When PI7C8150A 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.

## 3.6 READ TRANSACTIONS

Delayed read forwarding is used for all read transactions crossing PI7C8150A. Delayed read transactions are treated as either prefetchable or non-prefetchable. Table 3-5 shows the read behavior, prefetchable or non-prefetchable, for each type of read operation.

### 3.6.1 PREFETCHABLE READ TRANSACTIONS

A prefetchable read transaction is a read transaction where PI7C8150A 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, PI7C8150A forces all byte enable bits to be turned 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 pre-fetched depends on the type of transaction. The amount of pre-fetching may also be affected by the amount of free buffer space available in PI7C8150A, and by any read address boundaries encountered.

Pre-fetching 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.

### 3.6.2 NON-PREFETCHABLE READ TRANSACTIONS

A non-prefetchable read transaction is a read transaction where PI7C8150A 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, PI7C8150A 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.

### 3.6.3 READ PREFETCH ADDRESS BOUNDARIES

PI7C8150A imposes internal read address boundaries on read pre-fetched data. When a read transaction reaches one of these aligned address boundaries, the PI7C8150A stops pre-fetched data, unless the target signals a target disconnect before the read pre-fetched boundary is reached. When PI7C8150A 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_L. Section 3.6.6 describes flow-through mode during read operations.

Table 3-4 shows the read pre-fetch address boundaries for read transactions during non-flow-through mode.

**Table 3-4. 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, 16 | Cache line address boundary |
| Memory Read Line | - | CLS = 0 or 16 | 16-DWORD aligned address boundary |
| Memory Read Line | - | CLS = 1, 2, 4, 8, 16 | Cache line boundary |
| Memory Read Multiple | - | CLS = 0 or 16 | 32-DWORD aligned address boundary |
| Memory Read Multiple | - | CLS = 1, 2, 4, 8, 16 | 2X of cache line boundary |

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

**Table 3-5. 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 4.3 for detailed information about prefetchable and non-prefetchable address spaces.

### 3.6.4 DELAYED READ REQUESTS

PI7C8150A treats all read transactions as delayed read transactions, which means that the read request from the initiator is posted into a delayed transaction queue. Read data from the target is placed in the read data queue directed toward the initiator bus interface and is transferred to the initiator when the initiator repeats the read transaction.

When PI7C8150A accepts a delayed read request, it first samples the read address, read bus command, and address parity. When IRDY_L is asserted, PI7C8150A then samples the byte enable bits for the first data phase. This information is entered into the delayed transaction queue. PI7C8150A 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.

### 3.6.5 DELAYED READ COMPLETION WITH TARGET

When delayed read request reaches the head of the delayed transaction queue, PI7C8150A arbitrates for the target bus and initiates the read transaction only if all previously queued posted write transactions have been delivered. PI7C8150A 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, PI7C8150A 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 PI7C8150A 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, PI7C8150A does not initiate any further attempts to read more data.

If PI7C8150A is unable to obtain read data from the target after $2^{24}$ (default) or $2^{32}$ (maximum) attempts, PI7C8150A will report system error. The number of attempts is programmable. PI7C8150A also asserts P_SERR_L if the primary SERR_L enable bit is set in the command register. See Section 6.4 for information on the assertion of P_SERR_L.

Once PI7C8150A receives DEVSEL_L and TRDY_L 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 PI7C8150A 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 depends on the conditions given in Table 3-4 (assuming no disconnect is received from the target).

### 3.6.6 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 PI7C8150A transfers the data to the initiator when the initiator repeats the transaction. For memory read transactions, PI7C8150A aliases the memory read, memory read line, and memory read multiple bus commands when matching the bus command of the transaction to the bus command in the delayed transaction queue. PI7C8150A returns a target disconnect along with the transfer of the last DWORD of read data to the initiator. If PI7C8150A 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, PI7C8150A 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, PI7C8150A 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.

PI7C8150A 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 register. If the initiator does not repeat the read transaction and before the master timeout timer expires ($2^{15}$ default), PI7C8150A discards the read transaction and read data from its queues. PI7C8150A also conditionally asserts P_SERR_L (see Section 6.4).

PI7C8150A 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 5 for a discussion of how delayed read transactions are ordered when crossing PI7C8150A.

### 3.6.7 FAST BACK-TO-BACK READ TRANSACTION

PI7C8150A can recognize fast back-to-back read transactions.

## 3.7 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 PI7C8150A 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.

### 3.7.1 TYPE 0 ACCESS TO PI7C8150A

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 PI7C8150A responds to a Type 0 configuration transaction by asserting P_DEVSEL_L 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.

PI7C8150A 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 PI7C8150A ignores all Type 0 transactions initiated on the secondary interface.

### 3.7.2 TYPE 1 TO TYPE 0 CONVERSION

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.

PI7C8150A 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. PI7C8150A 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, PI7C8150A generates a Type 0 transaction only on the secondary bus, and never on the primary bus.

PI7C8150A 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 PI7C8150A 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].
- Decodes the device number and drives the bit pattern specified in Table 3-6 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.

PI7C8150A 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]. presents the mapping that PI7C8150A uses.

**Table 3-6. 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)<br>0000 0000 0000 0000 (P_AD[7:2] = 00h) | - |

PI7C8150A can assert up to 9 unique address lines to be used as IDSEL signals for up to 9 devices on the secondary bus, for device numbers ranging from 0 through 8. Because of electrical loading constraints of the PCI bus, more than 9 IDSEL signals should not be necessary. However, if device numbers greater than 9 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.

PI7C8150A 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.

### 3.7.3 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 PI7C8150A detects a Type 1 configuration transaction intended for a PCI bus downstream from the secondary bus, PI7C8150A 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.

PI7C8150A 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 PI7C8150A forwards Type 1 to Type 1 configuration write transactions as delayed transactions. Type 1 to Type 1 configuration write transactions are limited to a single data transfer.

### 3.7.4 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.

PI7C8150A 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_L is a configuration write command.

When PI7C8150A 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, PI7C8150A responds with TRDY_L to the next attempt of the con-figuration transaction from the initiator. If more than one data transfer is requested, PI7C8150A responds with a target disconnect operation during the first data phase.

## 3.8 TRANSACTION TERMINATION

This section describes how PI7C8150A 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_L at the beginning of the last data phase, and de-asserts IRDY# at the end of the last data phase in conjunction with either TRDY_L or STOP_L assertion from the target.

- **Master abort**

A master abort occurs when no target response is detected. When the initiator does not detect a DEVSEL_L from the target within five clock cycles after asserting FRAME_L, the initiator terminates the transaction with a master abort. If FRAME_L is still asserted, the initiator de-asserts FRAME_L on the next cycle, and then de-asserts IRDY_L on the following cycle. IRDY_L must be asserted in the same cycle in which FRAME_L de-asserts. If FRAME_L is already de-asserted, IRDY_L can be de-asserted 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_L and DEVSEL_L asserted in conjunction with FRAME_L de-asserted and IRDY_L asserted.

- **Target retry**

STOP_L and DEVSEL_L asserted with TRDY_L de-asserted during the first data phase. No data transfers occur during the transaction. This transaction must be repeated.

- **Target disconnect with data transfer**

STOP_L, DEVSEL_L and TRDY_L asserted. It signals that this is the last data transfer of the transaction.

- **Target disconnect without data transfer**

STOP_L and DEVSEL_L asserted with TRDY_L de-asserted after previous data transfers have been made. Indicates that no more data transfers will be made during this transaction.

- **Target abort**

STOP_L asserted with DEVSEL_L and TRDY_L de-asserted. Indicates that target will never be able to complete this transaction. DEVSEL_L must be asserted for at least one cycle during the transaction before the target abort is signaled.

### 3.8.1 MASTER TERMINATION INITIATED BY PI7C8150A

PI7C8150A, as an initiator, uses normal termination if DEVSEL_L is returned by target within five clock cycles of PI7C8150A's assertion of FRAME_L on the target bus. As an initiator, PI7C8150A 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 PI7C8150A's bus grant is de-asserted.
- The target terminates the transaction with a retry, disconnect, or target abort.

If PI7C8150A 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 PI7C8150A 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.

### 3.8.2 MASTER ABORT RECEIVED BY PI7C8150A

If the initiator initiates a transaction on the target bus and does not detect DEVSEL_L returned by the target within five clock cycles of the assertion of FRAME_L, PI7C8150A 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, PI7C8150A is able to reflect the master abort condition back to the initiator. When PI7C8150A detects a master abort in response to a delayed transaction, and when the initiator repeats the transaction, PI7C8150A does not respond to the transaction with DEVSEL_L, 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, PI7C8150A discards the posted write data and makes no more attempts to deliver the data. PI7C8150A 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_L enable bit (bit 8 of command register for secondary bus) are set, PI7C8150A asserts P_SERR_L if the master-abort-on-posted-write is not set. The master-abort-on-posted-write bit is bit 4 of the P_SERR_L event disable register (offset 64h).

**Note:** When PI7C8150A 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.

### 3.8.3 TARGET TERMINATION RECEIVED BY PI7C8150A

When PI7C8150A initiates a transaction on the target bus and the target responds with DEVSEL_L, the target can end the transaction with one of the following types of termination:

- Normal termination (upon de-assertion of FRAME_L)
- Target retry

- Target disconnect
- Target abort

PI7C8150A handles these terminations in different ways, depending on the type of transaction being performed.

#### 3.8.3.1 DELAYED WRITE TARGET TERMINATION RESPONSE

When PI7C8150A initiates a delayed write transaction, the type of target termination received from the target can be passed back to the initiator. Table 3-7 shows the response to each type of target termination that occurs during a delayed write transaction.

PI7C8150A repeats a delayed write transaction until one of the following conditions is met:

- PI7C8150A completes at least one data transfer.
- PI7C8150A receives a master abort.
- PI7C8150A receives a target abort.

PI7C8150A makes $2^{24}$ (default) or $2^{32}$ (maximum) write attempts resulting in a response of target retry.

**Table 3-7. 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 PI7C8150A makes $2^{24}$ (default) attempts of the same delayed write trans-action on the target bus, PI7C8150A asserts P_SERR_L if the SERR_L 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_L event disable register (offset 64h). PI7C8150A will report system error. See Section 6.4 for a description of system error conditions.

#### 3.8.3.2 POSTED WRITE TARGET TERMINATION RESPONSE

When PI7C8150A initiates a posted write transaction, the target termination cannot be passed back to the initiator. Table 3-8 shows the response to each type of target termination that occurs during a posted write transaction.

**Table 3-8. 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, PI7C8150A 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, PI7C8150A 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 PI7C8150A makes $2^{24}$ (default) write transaction attempts and fails to deliver all posted write data associated with that transaction, PI7C8150A asserts P_SERR_L if the primary SERR_L 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_L event disable register (offset 64h). PI7C8150A will report system error. See Section 6.4 for a discussion of system error conditions.

### 3.8.3.3 DELAYED READ TARGET TERMINATION RESPONSE

When PI7C8150A 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 3-9 shows the response to each type of target termination that occurs during a delayed read transaction.

PI7C8150A repeats a delayed read transaction until one of the following conditions is met:

- PI7C8150A completes at least one data transfer.
- PI7C8150A receives a master abort.
- PI7C8150A receives a target abort.

PI7C8150A makes $2^{24}$ (default) read attempts resulting in a response of target retry.

**Table 3-9. 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 Termination | Response |
|---|---|
| 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 PI7C8150A makes $2^{24}$(default) attempts of the same delayed read transaction on the target bus, PI7C8150A asserts P_SERR_L if the primary SERR_L 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_L event disable register (offset 64h). PI7C8150A will report system error. See Section 6.4 for a description of system error conditions.

## 3.8.4 TARGET TERMINATION INITIATED BY PI7C8150A

PI7C8150A 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.

### 3.8.4.1 TARGET RETRY

PI7C8150A returns a target retry to the initiator when it cannot accept write data or return read data as a result of internal conditions. PI7C8150A 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 PI7C8150A, 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 PI7C8150A, and the read transaction is not a locked transaction.
- PI7C78150B 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 PI7C8150A, 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.

#### 3.8.4.2 TARGET DISCONNECT

PI7C8150A returns a target disconnect to an initiator when one of the following conditions is met:

- PI7C8150A hits an internal address boundary.
- PI7C8150A cannot accept any more write data.
- PI7C8150A has no more read data to deliver.

See Section 3.5.4 for a description of write address boundaries, and Section 3.6.3 for a description of read address boundaries.

#### 3.8.4.3 TARGET ABORT

PI7C8150A returns a target abort to an initiator when one of the following conditions is met:

- PI7C8150A is returning a target abort from the intended target.

- When PI7C8150A returns a target abort to the initiator, it sets the signaled target abort bit in the status register corresponding to the initiator interface.

# 4 ADDRESS DECODING

PI7C8150A 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.

## 4.1 ADDRESS RANGES

PI7C8150A 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 PI7C8150A. The addresses that are not marked for downstream are always forwarded upstream.

## 4.2 I/O ADDRESS DECODING

PI7C8150A 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 4.4 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, PI7C8150A 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, PI7C8150A 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.*

### 4.2.1 I/O BASE AND LIMIT ADDRESS REGISTER

PI7C8150A 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. PI7C8150A supports 32-bit I/O addressing, which allows I/O addresses downstream of PI7C8150A 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 PI7C8150A 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.

### 4.2.2 ISA MODE

PI7C8150A supports ISA mode by providing an ISA enable bit in the bridge control register in configuration space. ISA mode modifies the response of PI7C8150A 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 PI7C8150A 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, PI7C8150A 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, PI7C8150A 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 PI7C8150A 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.

## 4.3 MEMORY ADDRESS DECODING

PI7C8150A 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 4.4.1 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.*

### 4.3.1 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. PI7C8150A 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 PI7C8150A uses to determine when to forward memory commands. PI7C8150A forwards a memory transaction from the primary to the secondary interface if the transaction address falls within the memory-mapped I/O address range. PI7C8150A 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.

### 4.3.2 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. PI7C8150A 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 PI7C8150A uses to determine when to forward memory commands. PI7C8150A forwards a memory transaction from the primary to the secondary interface if the transaction address falls within the prefetchable memory address range. PI7C8150A ignores memory transactions initiated on the secondary interface that fall into this address range. PI7C8150A 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.

## 4.4 VGA SUPPORT

PI7C8150A provides two modes for VGA support:

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

### 4.4.1 VGA MODE

When a VGA-compatible device exists downstream from PI7C8150A, set the VGA mode bit in the bridge control register in configuration space to enable VGA mode. When PI7C8150A 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. PI7C8150A 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. PI7C8150A 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 <15:10> 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.

### 4.4.2 VGA SNOOP MODE

PI7C8150A provides VGA snoop mode, allowing for VGA palette write transactions to be forwarded downstream. This mode is used when a graphics device downstream from PI7C8150A 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 PI7C8150A claims VGA palette write transactions by asserting DEVSEL_L in VGA snoop mode.

When VGA snoop bit is set, PI7C8150A 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, PI7C8150A behaves in the same way as if only the VGA mode bit were set.

# 5 TRANSACTION ORDERING

To maintain data coherency and consistency, PI7C8150A complies with the ordering rules set forth in the PCI Local Bus Specification, Revision 2.3, for transactions crossing the bridge. This chapter describes the ordering rules that control transaction forwarding across PI7C8150A.

## 5.1 TRANSACTIONS GOVERNED BY ORDERING RULES

Ordering relationships are established for the following classes of transactions crossing PI7C8150A:

**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.

PI7C8150A does not combine or merge write transactions:

- PI7C8150A does not combine separate write transactions into a single write transaction—this optimization is best implemented in the originating master.

- PI7C8150A does not merge bytes on separate masked write transactions to the same DWORD address—this optimization is also best implemented in the originating master.

- PI7C8150A 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.

## 5.2 GENERAL ORDERING GUIDELINES

Independent transactions on primary and secondary buses have a relationship only when those transactions cross PI7C8150A.

The following general ordering guidelines govern transactions crossing PI7C8150A:

- 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. PI7C8150A 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 PI7C8150A and must also be true for other bus agents. Otherwise, a deadlock can occur.

- PI7C8150A accepts posted write transactions, regardless of the state of completion of any delayed transactions being forwarded across PI7C8150A.

## 5.3 ORDERING RULES

Table 5-1 shows the ordering relationships of all the transactions and refers by number to the ordering rules that follow.

**Table 5-1. 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] | Yes | Yes | Yes | Yes |
| Delayed Write Request | No[4] | Yes | Yes | Yes | Yes |
| Delayed Read Completion | No[3] | Yes | Yes | Yes | Yes |
| Delayed Write Completion | Yes | Yes | Yes | Yes | Yes |

**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 PI7C8150A'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 5-1. These ordering rules apply to posted write transactions, delayed write and read requests, and delayed write and read completion transactions crossing PI7C8150A in the same direction. Note that delayed completion transactions cross PI7C8150A 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 PI7C8150A 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.

## 5.4 DATA SYNCHRONIZATION

Data synchronization refers to the relationship between interrupt signaling and data delivery. The PCI Local Bus Specification, Revision 2.3, 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.

PI7C8150A 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.

# 6 ERROR HANDLING

PI7C8150A checks, forwards, and generates parity on both the primary and secondary interfaces. To maintain transparency, PI7C8150A always tries to forward the existing parity condition on one bus to the other bus, along with address and data. PI7C8150A 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, PI7C8150A implements the following:

- PERR_L and SERR_L signals on both the primary and secondary interfaces
- Primary status and secondary status registers
- The device-specific P_SERR_L event disable register

This chapter provides detailed information about how PI7C8150A handles errors.
It also describes error status reporting and error operation disabling.

## 6.1 ADDRESS PARITY ERRORS

PI7C8150A checks address parity for all transactions on both buses, for all address and all bus commands. When PI7C8150A 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, PI7C8150A does not claim the transaction with P_DEVSEL_L; this may allow the transaction to terminate in a master abort. If parity error response bit is not set, PI7C8150A proceeds normally and accepts the transaction if it is directed to or across PI7C8150A.
- PI7C8150A sets the detected parity error bit in the status register.
- PI7C8150A asserts P_SERR_L and sets signaled system error bit in the status register, if both the following conditions are met:
    - The SERR_L enable bit is set in the command register.
    - The parity error response bit is set in the command register.

When PI7C8150A 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, PI7C8150A does not claim the transaction with S_DEVSEL_L; this may allow the transaction to terminate in a master abort. If parity error response bit is not set, PI7C8150A proceeds normally and accepts transaction if it is directed to or across PI7C8150A.
- PI7C8150A sets the detected parity error bit in the secondary status register.
- PI7C8150A asserts P_SERR_L and sets signaled system error bit in status register, if both of the following conditions are met:
    - The SERR_L enable bit is set in the command register.
    - The parity error response bit is set in the bridge control register.

## 6.2 DATA PARITY ERRORS

When forwarding transactions, PI7C8150A 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 PI7C8150A.

### 6.2.1 CONFIGURATION WRITE TRANSACTIONS TO CONFIGURATION SPACE

When PI7C8150A detects a data parity error during a Type 0 configuration write transaction to PI7C8150A configuration space, the following events occur:

If the parity error response bit is set in the command register, PI7C8150A asserts P_TRDY_L and writes the data to the configuration register. PI7C8150A also asserts P_PERR_L. If the parity error response bit is not set, PI7C8150A does not assert P_PERR_L.

PI7C8150A sets the detected parity error bit in the status register, regardless of the state of the parity error response bit.

### 6.2.2 READ TRANSACTIONS

When PI7C8150A detects a parity error during a read transaction, the target drives data and data parity, and the initiator checks parity and conditionally asserts PERR_L. For downstream transactions, when PI7C8150A detects a read data parity error on the secondary bus, the following events occur:

- PI7C8150A asserts S_PERR_L two cycles following the data transfer, if the secondary interface parity error response bit is set in the bridge control register.

- PI7C8150A sets the detected parity error bit in the secondary status register.
- PI7C8150A 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.
- PI7C8150A 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.
- PI7C8150A completes the transaction normally.

For upstream transactions, when PI7C8150A detects a read data parity error on the primary bus, the following events occur:

- PI7C8150A asserts P_PERR_L two cycles following the data transfer, if the primary interface parity error response bit is set in the command register.
- PI7C8150A sets the detected parity error bit in the primary status register.
- PI7C8150A 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.
- PI7C8150A 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.
- PI7C8150A completes the transaction normally.

PI7C8150A 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_L two cycles after the data transfer occurs. It is assumed that the initiator takes responsibility for handling a parity error condition; therefore, when PI7C8150A detects PERR_L asserted while returning read data to the initiator, PI7C8150A does not take any further action and completes the transaction normally.

### 6.2.3 DELAYED WRITE TRANSACTIONS

When PI7C8150A 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_L.

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 PI7C8150A 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 PI7C8150A 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, PI7C8150A asserts TRDY_L to the initiator and the transaction is not queued. If multiple data phases are requested, STOP_L is also asserted to cause a target disconnect. Two cycles after the data transfer, PI7C8150A also asserts PERR_L.

- If the parity-error-response bit is not set, PI7C8150A returns a target retry. It queues the transaction as usual. PI7C8150A does not assert PERR_L. In this case, the initiator repeats the transaction.

- PI7C8150A 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_L assertion).

For downstream transactions, when PI7C8150A is delivering data to the target on the secondary bus and S_PERR_L is asserted by the target, the following events occur:

- PI7C8150A 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.

- PI7C8150A captures the parity error condition to forward it back to the initiator on the primary bus.

Similarly, for upstream transactions, when PI7C8150A is delivering data to the target on the primary bus and P_PERR_L is asserted by the target, the following events occur:

- PI7C8150A 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.

- PI7C8150A 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 PI7C8150A has write status to return, the following events occur:

- PI7C8150A first asserts P_TRDY_L and then asserts P_PERR_L two cycles later, if the primary interface parity-error-response bit is set in the command register.
- PI7C8150A 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 PI7C8150A has write status to return, the following events occur:

- PI7C8150A first asserts S_TRDY_L and then asserts S_PERR_L two cycles later, if the secondary interface parity-error-response bit is set in the bridge control register (offset 3Ch).
- PI7C8150A 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:

- PI7C8150A asserts P_PERR_L 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.
- PI7C8150A 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:

- PI7C8150A asserts S_PERR_L 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.

- PI7C8150A completes the transaction normally.

### 6.2.4 POSTED WRITE TRANSACTIONS

During downstream posted write transactions, when PI7C8150A responds as a target, it detects a data parity error on the initiator (primary) bus and the following events occur:

- PI7C8150A asserts P_PERR_L two cycles after the data transfer, if the parity error response bit is set in the command register of primary interface.
- PI7C8150A sets the parity error detected bit in the status register of the primary interface.
- PI7C8150A captures and forwards the bad parity condition to the secondary bus.
- PI7C8150A completes the transaction normally.

Similarly, during upstream posted write transactions, when PI7C8150A responds as a target, it detects a data parity error on the initiator (secondary) bus, the following events occur:

- PI7C8150A asserts S_PERR_L two cycles after the data transfer, if the parity error response bit is set in the bridge control register of the secondary interface.
- PI7C8150A sets the parity error detected bit in the status register of the secondary interface.
- PI7C8150A captures and forwards the bad parity condition to the primary bus.
- PI7C8150A 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_L, the following events occur:

- PI7C8150A 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.
- PI7C8150A asserts P_SERR_L and sets the signaled system error bit in the status register, if all the following conditions are met:
    - The SERR_L enable bit is set in the command register.
    - The posted write parity error bit of P_SERR_L 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.
    - PI7C8150A 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_L, the following events occur:

- PI7C8150A 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.
- PI7C8150A asserts P_SERR_L and sets the signaled system error bit in the status register, if all the following conditions are met:
    - The SERR_L 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.
    - PI7C8150A 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_L 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_L will not be asserted.

## 6.3 DATA PARITY ERROR REPORTING SUMMARY

In the previous sections, the responses of PI7C8150A to data parity errors are presented according to the type of transaction in progress. This section organizes the responses of PI7C8150A to data parity errors according to the status bits that PI7C8150A sets and the signals that it asserts.

Table 6-1 shows setting the detected parity error bit in the status register, corresponding to the primary interface. This bit is set when PI7C8150A detects a parity error on the primary interface.

**Table 6-1. Setting the Primary Interface Detected Parity Error Bit**

| 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 |

**X = don't care**

Table 6-2 shows setting the detected parity error bit in the secondary status register, corresponding to the secondary interface. This bit is set when PI7C8150A detects a parity error on the secondary interface.

**Table 6-2. Setting 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 |
| 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 |

**X = don't care**

Table 6-3 shows setting data parity detected bit in the primary interface's status register. This bit is set under the following conditions:

- PI7C8150A 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_L signal is detected asserted or a parity error is detected on the primary bus.

**Table 6-3. Setting Primary Interface Master Data Parity Error Detected Bit**

| 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 |

**X = don't care**

Table 6-4 shows setting the data parity detected bit in the status register of secondary interface. This bit is set under the following conditions:

- The PI7C8150A 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_L signal is detected asserted or a parity error is detected on the secondary bus.

**Table 6-4. Setting Secondary Interface Master Data Parity Error 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 |
| 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 |

**X= don't care**

Table 6-5 shows assertion of P_PERR_L. This signal is set under the following conditions:

- PI7C8150A 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.
- PI7C8150A detects a data parity error on the primary bus or detects S_PERR_L asserted during the completion phase of a downstream delayed write transaction on the target (secondary) bus.

**Table 6-5. Assertion of P_PERR_L**

| P_PERR_L | 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 |

**X = don't care**
**[2]The parity error was detected on the target (secondary) bus but not on the initiator (primary) bus.**

Table 6-6 shows assertion of S_PERR_L that is set under the following conditions:

- PI7C8150A 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.
- PI7C8150A detects a data parity error on the secondary bus or detects P_PERR_L asserted during the completion phase of an upstream delayed write transaction on the target (primary) bus.

**Table 6-6. Assertion of S_PERR_L**

| S_PERR_L | 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 |
| 1 | Delayed Write | Downstream | Secondary | x / x |
| 0[2] | Delayed Write | Upstream | Primary | 1 / 1 |
| 0 | Delayed Write | Upstream | Secondary | x / 1 |

**X = don't care**
**[2]The parity error was detected on the target (secondary) bus but not on the initiator (primary) bus.**

Table 6-7 shows assertion of P_SERR_L. This signal is set under the following conditions:

- PI7C8150A has detected P_PERR_L asserted on an upstream posted write transaction or S_PERR_L asserted on a downstream posted write transaction.
- PI7C8150A 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_L enable bit must be set in the command register.

**Table 6-7. Assertion of P_SERR_L for Data Parity Errors**

| P_SERR_L | 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 |

| P_SERR_L | Transaction Type | Direction | Bus Where Error Was Detected | Primary / Secondary Parity Error Response Bits |
|---|---|---|---|---|
| 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 |

**X = don't care**

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

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

## 6.4 SYSTEM ERROR (SERR_L) REPORTING

PI7C8150A uses the P_SERR_L signal to report conditionally a number of system error conditions in addition to the special case parity error conditions described in Section 6.2.3.

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

- For PI7C8150A to assert P_SERR_L for any reason, the SERR_L enable bit must be set in the command register.
- Whenever PI7C8150A asserts P_SERR_L, PI7C8150A must also set the signaled system error bit in the status register.

In compliance with the PCI-to-PCI Bridge Architecture Specification, PI7C8150A asserts P_SERR_L when it detects the secondary SERR_L input, S_SERR_L, asserted and the SERR_L forward enable bit is set in the bridge control register. In addition, PI7C8150A also sets the received system error bit in the secondary status register.

PI7C8150A also conditionally asserts P_SERR_L 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_L status register reports the reason for the assertion of P_SERR_L. Most of these events have additional device-specific disable bits in the P_SERR_L event disable register that make it possible to mask out P_SERR_L assertion for specific events. The master timeout condition has a SERR_L enable bit for that event in the bridge control register and therefore does not have a device-specific disable bit.

# 7 EXCLUSIVE ACCESS

This chapter describes the use of the LOCK_L signal to implement exclusive access to a target for transactions that cross PI7C8150A.

## 7.1 CONCURRENT LOCKS

The primary and secondary bus lock mechanisms operate concurrently except when a locked transaction crosses PI7C8150A. 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.

## 7.2 ACQUIRING EXCLUSIVE ACCESS ACROSS PI7C8150A

For any PCI bus, before acquiring access to the LOCK_L 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_L signal must be de-asserted.

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

### 7.2.1 LOCKED TRANSACTIONS IN DOWNSTREAM DIRECTION

Locked transactions can cross PI7C8150A 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 PI7C8150A detects on the primary bus, an initial locked transaction intended for a target on the secondary bus, PI7C8150A samples the address, transaction type, byte enable bits, and parity, as described in Section 3.5.4. 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, PI7C8150A does not queue any more transactions until the locked sequence is finished. PI7C8150A signals a target retry to all transactions initiated subsequent to the locked read transaction that are intended for targets on the other side of PI7C8150A. PI7C8150A 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, PI7C8150A initiates the transaction as a locked read transaction by de-asserting LOCK_L on the target bus during the first address phase, and by asserting LOCK_L one cycle later. If LOCK_L is already asserted (used by another initiator), PI7C8150A waits to request access to the secondary bus until LOCK_L is de-asserted when the target bus is idle. Note that the existing lock on the target bus could not have crossed PI7C8150A. Otherwise, the pending queued locked transaction would not have been queued. When PI7C8150A 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, PI7C8150A transfers the read data back to the initiator, and the lock is then also established on the primary bus.

For PI7C8150A 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_L during address phase, and assert LOCK_L one cycle later). If the LOCK_L sequence is not used in subsequent attempts, a master timeout condition may result. When a master timeout condition occurs, SERR_L is conditionally asserted (see Section 6.4), the read data and queued read transaction are discarded, and the LOCK_L 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 PI7C8150A are driven as locked transactions on the target bus.

The first transaction to establish LOCK_L 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 PI7C8150A 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. PI7C8150A resumes forwarding unlocked transactions in both directions.

### 7.2.2 LOCKED TRANSACTION IN UPSTREAM DIRECTION

PI7C8150A ignores upstream lock and transactions. PI7C8150A will pass these transactions as normal transactions without lock established.

## 7.3 ENDING EXCLUSIVE ACCESS

After the lock has been acquired on both initiator and target buses, PI7C8150A 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. PI7C8150A does not know whether the current transaction is the last one in a sequence of locked transactions until the initiator de-asserts the LOCK_L signal at
end of the transaction.

When the last locked transaction is a delayed transaction, PI7C8150A has already completed the transaction on the target bus. In this example, as soon as PI7C8150A detects that the initiator has relinquished the LOCK_L signal by sampling it in the de-asserted state while FRAME_L is de-asserted, PI7C8150A de-asserts the LOCK_L signal on the target bus as soon as possible. Because of this behavior, LOCK_L may not be de-asserted until several cycles after the last locked transaction has been completed on the target bus. As soon as PI7C8150A has de-asserted LOCK_L 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, PI7C8150A de-asserts LOCK_L 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 PI7C8150A receives a target abort or a master abort in response to a locked delayed transaction, PI7C8150A returns a target abort or a master abort when the initiator repeats the locked transaction. The initiator must then de-assert LOCK_L at the end of the transaction. PI7C8150A sets the appropriate status bits, flagging the abnormal target termination condition (see Section 3.8). Normal forwarding of unlocked posted and delayed transactions is resumed.

When PI7C8150A receives a target abort or a master abort in response to a locked posted write transaction, PI7C8150A cannot pass back that status to the initiator. PI7C8150A asserts SERR_L on the initiator bus when a target abort or a master abort is received during

a locked posted write transaction, if the SERR_L enable bit is set in the command register. Signal SERR_L is asserted for the master abort condition if the master abort mode bit is set in the bridge control register (see Section 6.4).

# 8 PCI BUS ARBITRATION

PI7C8150A 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 PI7C8150A, typically on the motherboard. For the secondary PCI bus, PI7C8150A 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.

## 8.1 PRIMARY PCI BUS ARBITRATION

PI7C8150A implements a request output pin, P_REQ_L, and a grant input pin, P_GNT_L, for primary PCI bus arbitration. PI7C8150A asserts P_REQ_L 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, PI7C8150A keeps P_REQ_L asserted. However, if a target retry, target disconnect, or a target abort is received in response to a transaction initiated by PI7C8150A on the primary PCI bus, PI7C8150A de-asserts P_REQ_L for two PCI clock cycles.

For all cycles through the bridge, P_REQ_L is not asserted until the transaction request has been completely queued. When P_GNT_L is asserted LOW by the primary bus arbiter after PI7C8150A has asserted P_REQ_L, PI7C8150A initiates a transaction on the primary bus during the next PCI clock cycle. When P_GNT_L is asserted to PI7C8150A when P_REQ_L is not asserted, PI7C8150A parks P_AD, P_CBE, and P_PAR by driving them to valid logic levels. When the primary bus is parked at PI7C8150A and PI7C8150A has a transaction to initiate on the primary bus, PI7C8150A starts the transaction if P_GNT_L was asserted during the previous cycle.

## 8.2 SECONDARY PCI BUS ARBITRATION

PI7C8150A implements an internal secondary PCI bus arbiter. This arbiter supports eight external masters on the secondary bus in addition to PI7C8150A. The internal arbiter can be disabled, and an external arbiter can be used instead for secondary bus arbitration.

### 8.2.1 SECONDARY BUS ARBITRATION USING THE INTERNAL ARBITER

To use the internal arbiter, the secondary bus arbiter enable pin, S_CFN_L, must be tied LOW. PI7C8150A has nine secondary bus request input pins, S_REQ_L[8:0], and has nine secondary bus output grant pins, S_GNT_L[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_L is HIGH.

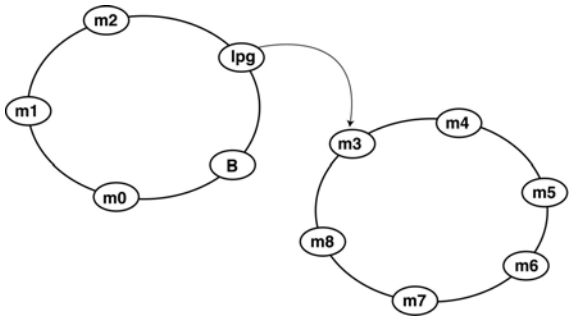


**Figure 8-1 Secondary Arbiter Example**

The secondary arbiter supports a 2-sets programmable 2-level rotating algorithm with each set taking care of 9 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 8-1 shows an example of an internal arbiter where four masters, including PI7C8150A, 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**, *B, m0, m1, m2,* **m7** and so on.

Each bus master, including PI7C8150A, 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 PI7C8150A is assigned to the high priority group. PI7C8150A 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_L 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 PI7C8150A detects that an initiator has failed to assert S_FRAME_L 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_L or S_IRDY_L is asserted, the arbiter can be de-asserted one grant and asserted another grant during the same PCI clock cycle.

### 8.2.2 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.

### 8.2.3 SECONDARY BUS ARBITRATION USING AN EXTERNAL ARBITER

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

When S_CFN_L is tied HIGH, PI7C8150A, reconfigures two pins to be external request and grant pins. The S_GNT_L[0] pin is reconfigured to be the external request pin because it's an output. The S_REQ_L[0] pin is reconfigured to be the external grant pin because it's an input. When an external arbiter is used, PI7C8150A uses the S_GNT_L[0] pin to request the secondary bus. When the reconfigured S_REQ_L[0] pin is asserted LOW after PI7C8150A has asserted S_GNT_L[0], PI7C8150A initiates a transaction on the secondary bus one cycle later. If grant is asserted and PI7C8150A has not asserted the request, PI7C8150A parks AD, CBE and PAR pins by driving them to valid logic levels.

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

### 8.2.4 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 and CBE signals should be driven first, with the PAR signal driven one cycle later.

PI7C8150A parks the primary bus only when P_GNT_L is asserted, P_REQ_L is de-asserted, and the primary PCI bus is idle. When P_GNT_L is de-asserted, PI7C8150A 3-states the P_AD, P_CBE, and P_PAR signals on the next PCI clock cycle. If PI7C8150A is parking the primary PCI bus and wants to initiate a transaction on that bus, then

PI7C8150A can start the transaction on the next PCI clock cycle by asserting P_FRAME_L if P_GNT_L 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, PI7C8150A keeps the secondary bus grant asserted to a particular master until a new secondary bus request comes along. After reset, PI7C8150A 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 PI7C8150A. By default, offset 48h, bit 1, is set to 0. If the internal arbiter is disabled, PI7C8150A parks the secondary bus only when the reconfigured grant signal, S_REQ_L[0], is asserted and the secondary bus is idle.

# 9 CLOCKS

This chapter provides information about the clocks.

## 9.1 PRIMARY CLOCK INPUTS

PI7C8150A 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. The secondary clock is derived internally from the primary clock, P_CLK. PI7C8150A operates at a maximum frequency of 66 MHz (33MHz for PI7C8150A-33).

## 9.2 SECONDARY CLOCK OUTPUTS

PI7C8150A 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. The secondary clock edges are delayed from P_CLK edges by a minimum of 0ns. This is the rule for using secondary clocks:

Each secondary clock output is limited to no more than one load.

# 10 GENERAL PURPOSE I/O INTERFACE

The PI7C8150A 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 PI7C8150A to a halt through hardware, permitting live insertion of option cards behind the PI7C8150A.

## 10.1 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 8-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.

## 10.2 SECONDARY CLOCK CONTROL

The PI7C8150A 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_RST_L is detected deasserted and the secondary reset signal, S_RST_L, is detected asserted. The deassertion of S_RST_L is delayed until the PI7C8150A 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 10-1 shows the operation of the GPIO pins.

**Table 10-1. GPIO Operation**

| GPIO Pin | Operation |
|---|---|
| GPIO[0] | Shift register clock output at 33MHz max frequency |
| GPIO[1] | Not used |
| GPIO[2] | Shift register control<br>0: Load<br>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 PI7C8150A. 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 10-2 shows the format of the serial stream.

**Table 10-2. 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] | PI7C8150A 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 PI7C8150A's S_CLKIN input.

If desired, the assignment of S_CLKOUT outputs to slots, devices, and PI7C8150A'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 PI7C8150A S_CLKIN input. When the secondary reset signal, S_RST_L, is detected asserted and the primary reset signal, P_RST_L, is detected deasserted, PI7C8150A drives GPIO[2] LOW for one cycle to load the clock mask inputs into the shift register. On the next cycle, PI7C8150A 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, PI7C8150A tri-states the GPIO signals and can deassert S_RST_L if the secondary reset bit is clear. PI7C8150A then ignores MSK_IN. Control of the GPIO signal now reverts to PI7C8150A 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.

## 10.3 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, PI7C8150A 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, PI7C8150A 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 PI7C8150A is not responding to any I/O or memory transactions during this time. PI7C8150A continues to accept configuration transactions in live insertion mode. Once live insertion mode brings PI7C8150A to a halt and queued transactions are completed, the secondary reset bit in the bridge control register can be used to assert S_RST_L, if desired, to reset and tri-state secondary bus devices, and to enable any live insertion hardware.

# 11 PCI POWER MANAGEMENT

PI7C8150A 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 11-1 shows the states and related actions that PI7C8150A performs during power management transitions. (No other transactions are permitted.)

**Table 11-1. Power Management Transitions**

| Current Status | Next State | Action |
|---|---|---|
| D0 | D3cold | Power has been removed from PI7C8150A. A power-up reset must be performed to bring PI7C8150A to D0. |
| D0 | D3hot | If enabled to do so by the BPCCE pin, PI7C8150A will disable the secondary clocks and drive them LOW. |
| D0 | D2 | Unimplemented power state. PI7C8150A will ignore the write to the power state bits (power state remains at D0). |
| D0 | D1 | Unimplemented power state. PI7C8150A will ignore the write to the power state bits (power state remains at D0). |
| D3hot | D0 | PI7C8150A enables secondary clock outputs and performs an internal chip reset. Signal S_RST_L will not be asserted. All registers will be returned to the reset values and buffers will be cleared. |
| D3hot | D3cold | Power has been removed from PI7C8150A. A power-up reset must be performed to bring PI7C8150A to D0. |
| D3cold | D0 | Power-up reset. PI7C8150A performs the standard power-up reset functions as described in Section 12. |

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

# 12 RESET

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

## 12.1 PRIMARY INTERFACE RESET

PI7C8150A has a reset input, P_RESET_L. When P_RESET_L is asserted, the following events occur:

- PI7C8150A immediately tri-states all primary and secondary PCI interface signals.
- PI7C8150A performs a chip reset.
- Registers that have default values are reset.

P_RESET_L asserting and de-asserting edges can be asynchronous to P_CLK and S_CLKOUT. PI7C8150A is not accessible during P_RESET_L. After P_RESET_L is de-asserted, PI7C8150A remains inaccessible for 16 PCI clocks before the first configuration transaction can be accepted.

## 12.2 SECONDARY INTERFACE RESET

PI7C8150A is responsible for driving the secondary bus reset signals, S_RESET_L. PI7C8150A asserts S_RESET_L when any of the following conditions are met:

**Signal P_RESET_L is asserted.** Signal S_RESET_L remains asserted as long as P_RESET_L is asserted and does not de-assert until P_RESET_L is de-asserted.

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

**S_RESET_L pin is asserted.** When S_RESET_L is asserted, PI7C8150A immediately 3-states all the secondary PCI interface signals associated with the secondary port. The S_RESET_L in asserting and de-asserting edges can be asynchronous to P_CLK.

When S_RESET_L is asserted, all secondary PCI interface control signals, including the secondary grant outputs, are immediately 3-stated. Signals S1_AD, S1_CBE[3:0]#, S_PAR are driven low for the duration of S_RESET_L assertion. 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_L is asserted by means of the secondary reset bit, PI7C8150A remains accessible during secondary interface reset and continues to respond to accesses to its configuration space from the primary interface.

## 12.3 CHIP RESET

The chip reset bit in the diagnostic control register can be used to reset the PI7C8150A and the secondary bus.

When the chip reset bit is set, all registers and chip state are reset and all signals are tristated. S_RESET_L is asserted and the secondary reset bit is automatically set. S_RESET_L remains asserted until a configuration write operation clears the secondary reset bit and the serial clock mask has been shifted in. Within 20 PCI clock cycles after completion of the configuration write operation, PI7C8150A's reset bit automatically clears and PI7C8150A is ready for configuration.

During reset, PI7C8150A is inaccessible.

# 13 SUPPORTED COMMANDS

The PCI command set is given below for the primary and secondary interfaces.

## 13.1 PRIMARY INTERFACE

| P_CBE [3:0] | Command | Action |
|---|---|---|
| 0000 | Interrupt Acknowledge | Ignore |

| P_CBE [3:0] | Command | Action |
|---|---|---|
| 0001 | Special Cycle | Do not claim. Ignore. |
| 0010 | I/O Read | 1. If address is within pass through I/O range, claim and pass through.<br><br>2. Otherwise, do not pass through and do not claim for internal access. |
| 0011 | I/O Write | Same as I/O Read. |
| 0100 | Reserved | ----- |
| 0101 | Reserved | ----- |
| 0110 | Memory Read | 1. If address is within pass through memory range, claim and pass through.<br><br>2. If address is within pass through memory mapped I/O range, claim and pass through.<br><br>3. Otherwise, do not pass through and do not claim for internal access. |
| 0111 | Memory Write | Same as Memory Read. |
| 1000 | Reserved | ----- |
| 1001 | Reserved | ----- |
| 1010 | Configuration Read | **Type 0 Configuration Read:**<br>If the bridge's IDSEL line is asserted, perform function decode and claim if target function is implemented. Otherwise, ignore. If claimed, permit access to target function's configuration registers. Do not pass through under any circumstances.<br><br>**Type 1 Configuration Read:**<br>1. If the target bus is the bridge's secondary bus: claim and pass through as a Type 0 Configuration Read.<br><br>2. If the target bus is a subordinate bus that exists behind the bridge (but not equal to the secondary bus): claim and pass through as a Type 1 Configuration Read.<br><br>3. Otherwise, ignore. |
| 1011 | Configuration Write | **Type 0 Configuration Write: same as Configuration Read.**<br><br>**Type 1 Configuration Write (not special cycle request):**<br>1. If the target bus is the bridge's secondary bus: claim and pass through as a Type 0 Configuration Write<br><br>2. If the target bus is a subordinate bus that exists behind the bridge (but not equal to the secondary bus): claim and pass through unchanged as a Type 1 Configuration Write.<br><br>3. Otherwise, ignore.<br><br>**Configuration Write as Special Cycle Request (device = 1Fh, function = 7h)**<br>1. If the target bus is the bridges secondary bus: claim and pass through as a special cycle.<br><br>2. If the target bus is a subordinate bus that exists behind the bridge (but not equal to the secondary bus): claim and pass through unchanged as a type 1 Configuration Write.<br><br>3. Otherwise ignore |
| 1100 | Memory Read Multiple | Same as Memory Read |
| 1101 | Dual Address Cycle | Supported |
| 1110 | Memory Read Line | Same as Memory Read |

| P_CBE [3:0] | Command | Action |
|---|---|---|
| 1111 | Memory Write and Invalidate | Same as Memory Read |

## 13.2 SECONDARY INTERFACE

| S_CBE[3:0] | Command | Action |
|---|---|---|
| 0000 | Interrupt Acknowledge | Ignore |
| 0001 | Special Cycle | Do not claim. Ignore. |
| 0010 | I/O Read | Same as Primary Interface |
| 0011 | I/O Write | Same as I/O Read. |
| 0100 | Reserved | ----- |
| 0101 | Reserved | ----- |
| 0110 | Memory Read | Same as Primary Interface |
| 0111 | Memory Write | Same as Memory Read. |
| 1000 | Reserved | ----- |
| 1001 | Reserved | ----- |
| 1010 | Configuration Read | Ignore |
| 1011 | Configuration Write | **I. Type 0 Configuration Write: Ignore**<br><br>**II. Type 1 Configuration Write (not special cycle request):Ignore**<br><br>**III. Configuration Write as Special Cycle Request (device = 1Fh, function = 7h):**<br>1. If the target bus is the bridge's primary bus: claim and pass through as a Special Cycle<br><br>2. If the target bus is neither the primary bus nor is it in range of buses defined by the bridge's secondary and subordinate bus registers: claim and pass through unchanged as a Type 1 Configuration Write.<br><br>3. If the target bus is not the bridge's primary bus, but is in range of buses defined by the bridge's secondary and subordinate bus registers: ignore. |
| 1100 | Memory Read Multiple | Same as Memory Read |
| 1101 | Dual Address Cycle | Supported |
| 1110 | Memory Read Line | Same as Memory Read |
| 1111 | Memory Write and Invalidate | Same as Memory Read |

# 14 CONFIGURATION REGISTERS

PCI configuration defines a 64-byte space (configuration header) to define various attributes of PI7C8150A as shown below.

## 14.1 CONFIGURATION REGISTER

| 31-24 | 23-16 | 15-8 | 7-0 | 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 | I/O Base | 1Ch |
| Memory Limit | | Memory Base | | 20h |
| Prefetchable Memory Limit | | Prefetchable Memory Base | | 24h |
| Prefetchable Base Upper 32-bit | | | | 28h |
| Prefetchable Limit Upper 32-bit | | | | 2Ch |
| I/O Limit Upper 16-bit | | I/O Base Upper 16-bit | | 30h |
| Reserved | | | Capability Pointer to DCh | 34h |
| Reserved | | | | 38h |
| Bridge Control | | Reserved | Interrupt Line | 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 |
| Upstream (S to P) Memory Limit | | Upstream (S to P) Memory Base | | 50h |
| Upstream (S to P) Memory Base Upper 32-bit | | | | 54h |
| Upstream (S to P) Memory Limit Upper 32-bit | | | | 58h |
| Reserved | | | | 5Ch |
| Reserved | | | | 60h |
| GPIO Data and Control | | | P_SERR# Event Disable | 64h |
| Reserved | P_SERR_L Status | Secondary Clock Control | | 68h |
| Reserved | | | | 6Ch |
| Reserved | | | | 70h |
| Reserved | | Port Option | | 74h |
| Retry Counter | | | | 78h |
| Reserved | | | | 7Ch |
| Secondary Master Timeout Counter | | Primary Master Timeout Counter | | 80h |
| Reserved | | | | 84h-AFh |
| Chassis Number | Slot Number | Next Pointer | Capability ID | B0h |
| Reserved | | | | B4h-D8h |
| Power Management Capabilities | | Next Item Pointer | Capability ID | DCh |
| Reserved | PPB Support Extensions | Power Management Data | | E0h |
| Reserved | | Next Pointer | Capability ID | E4h |
| Reserved | | | | E8h-FFh |

### 14.1.1 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. |

### 14.1.2 DEVICE ID REGISTER – OFFSET 00h

| Bit | Function | Type | Description |
|---|---|---|---|
| 31:16 | Device ID | R/O | Identifies this device as the PI7C8150A. Hardwired as 8150h. |

### 14.1.3 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<br><br>0: ignore I/O transactions on the primary interface<br><br>1: enable response to I/O transactions on the primary interface<br><br>Reset to 0 |
| 1 | Memory Space Enable | R/W | Controls response to memory accesses on the primary interface<br><br>0: ignore memory transactions on the primary interface<br><br>1: enable response to memory transactions on the primary interface<br><br>Reset to 0 |
| 2 | Bus Master Enable | R/W | Controls ability to operate as a bus master on the primary interface<br><br>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<br><br>1: enables PI7C8150A to operate as a master on the primary interfaces for memory and I/O transactions forwarded from the secondary interface<br><br>Reset to 0 |
| 3 | Special Cycle Enable | R/O | No special cycles defined.<br>Bit is defined as read only and returns 0 when read |
| 4 | Memory Write And Invalidate Enable | R/O | Memory write and invalidate not supported.<br>Bit is implemented as read only and returns 0 when read (unless forwarding a transaction for another master) |
| 5 | VGA Palette Snoop Enable | R/W | Controls response to VGA compatible palette accesses<br><br>0: ignore VGA palette accesses on the primary<br><br>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) |

| Bit | Function | Type | Description |
|---|---|---|---|
| 6 | Parity Error Response | R/W | Controls response to parity errors<br><br>0: PI7C8150A may ignore any parity errors that it detects and continue normal operation<br><br>1: PI7C8150A must take its normal action when a parity error is detected<br><br>Reset to 0 |
| 7 | Wait Cycle Control | R/O | Controls the ability to perform address / data stepping<br><br>0: disable address/data stepping (affects primary and secondary)<br><br>1: enable address/data stepping (affects primary and secondary)<br><br>Reset to 0 |
| 8 | P_SERR_L enable | R/W | Controls the enable for the P_SERR_L pin<br><br>0: disable the P_SERR_L driver<br><br>1: enable the P_SERR_L driver<br><br>Reset to 0 |
| 9 | Fast Back-to-Back Enable | R/W | Controls PI7C8150A's ability to generate fast back-to-back transactions to different devices on the primary interface.<br><br>0: no fast back-to-back transactions<br><br>1: enable fast back-to-back transactions<br><br>Reset to 0 |
| 15:10 | Reserved | R/O | Returns 000000 when read |

### 14.1.4 STATUS REGISTER – OFFSET 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)<br><br>Reset to 1 |
| 21 | 66MHz Capable | R/O | Set to 1 to enable 66MHz operation on the primary interface<br><br>Reset to 1 |
| 22 | Reserved | R/O | Reset to 0 |
| 23 | Fast Back-to-Back Capable | R/O | Set to 1 to enable decoding of fast back-to-back transactions on the primary interface to different targets<br><br>Reset to 1 |
| 24 | Data Parity Error Detected | R/WC | Set to 1 when P_PERR_L is asserted and bit 6 of command register is set<br><br>Reset to 0 |
| 26:25 | DEVSEL_L timing | R/O | DEVSEL_L timing (medium decoding)<br><br>00: fast DEVSEL_L decoding<br>01: medium DEVSEL_L decoding<br>10: slow DEVSEL_L decoding<br>11: reserved<br><br>Reset to 01 |

| Bit | Function | Type | Description |
|---|---|---|---|
| 27 | Signaled Target Abort | R/WC | Set to 1 (by a target device) whenever a target abort cycle occurs<br><br>Reset to 0 |
| 28 | Received Target Abort | R/WC | Set to 1 (by a master device) whenever transactions are terminated with target aborts<br><br>Reset to 0 |
| 29 | Received Master Abort | R/WC | Set to 1 (by a master) when transactions are terminated with Master Abort<br><br>Reset to 0 |
| 30 | Signaled System Error | R/WC | Set to 1 when P_SERR_L is asserted<br><br>Reset to 0 |
| 31 | Detected Parity Error | R/WC | Set to 1 when address or data parity error is detected on the primary interface<br><br>Reset to 0 |

### 14.1.5 REVISION ID REGISTER – OFFSET 08h

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

### 14.1.6 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 |

### 14.1.7 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.<br>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).<br><br>Reset to 0 |

### 14.1.8 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_L.<br><br>Reset to 0 |

### 14.1.9 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. |

### 14.1.10 PRIMARY BUS NUMBER REGISTSER – 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.<br><br>Reset to 0 |

### 14.1.11 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.<br><br>Reset to 0 |

### 14.1.12 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.<br><br>Reset to 0 |

### 14.1.13 SECONDARY LATENCY TIMER REGISTER – 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.<br><br>Reset to 0 |

### 14.1.14 I/O BASE REGISTER – OFFSET 1Ch

| Bit | Function | Type | Description |
|---|---|---|---|
| 3:0 | 32-bit Indicator | R/O | Read as 01h to indicate 32-bit I/O addressing |

| 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<br><br>Reset to 0 |

## 14.1.15 I/O LIMIT REGISTER – OFFSET 1Ch

| Bit | Function | Type | Description |
|---|---|---|---|
| 11:8 | 32-bit Indicator | R/O | Read as 01h to indicate 32-bit I/O addressing |
| 15:12 | I/O Base 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 base address upper 16 bits address register<br><br>Reset to 0 |

## 14.1.16 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<br><br>Reset to 1 |
| 22 | Reserved | R/O | Reset to 0 |
| 23 | Fast Back-to-Back Capable | R/O | Set to 1 to enable decoding of fast back-to-back transactions on the secondary interface to different targets<br><br>Reset to 0 |
| 24 | Master Data Parity Error Detected | R/WC | Set to 1 when S_PERR_L is asserted and bit 6 of command register is set<br><br>Reset to 0 |
| 26:25 | DEVSEL_L timing | R/O | DEVSEL# timing (medium decoding)<br><br>00: fast DEVSEL_L decoding<br>01: medium DEVSEL_L decoding<br>10: slow DEVSEL_L decoding<br>11: reserved<br><br>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<br><br>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<br><br>Reset to 0 |

| Bit | Function | Type | Description |
|---|---|---|---|
| 29 | Received Master Abort | R/WC | Set to 1 (by a master) when transactions on its secondary interface are terminated with Master Abort<br><br>Reset to 0 |
| 30 | Received System Error | R/WC | Set to 1 when S_SERR_L is asserted<br><br>Reset to 0 |
| 31 | Detected Parity Error | R/WC | Set to 1 when address or data parity error is detected on the secondary interface<br><br>Reset to 0 |

### 14.1.17 MEMORY BASE REGISTER – OFFSET 20h

| Bit | Function | Type | Description |
|---|---|---|---|
| 3:0 | | R/O | Lower four bits of register are read only and return 0.<br><br>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.<br><br>Reset to 0 |

### 14.1.18 MEMORY LIMIT REGISTER – OFFSET 20h

| Bit | Function | Type | Description |
|---|---|---|---|
| 19:16 | | R/O | Lower four bits of register are read only and return 0.<br><br>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. |

### 14.1.19 PEFETCHABLE MEMORY BASE REGISTER – OFFSET 24h

| Bit | Function | Type | Description |
|---|---|---|---|
| 3:0 | 64-bit addressing | R/O | Indicates 64-bit addressing<br><br>0000: 32-bit addressing<br><br>0001: 64-bit addressing<br><br>Reset to 1 |
| 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. |

### 14.1.20 PREFETCHABLE MEMORY LIMIT REGISTER – OFFSET 24h

| Bit | Function | Type | Description |
|---|---|---|---|
| 19:16 | 64-bit addressing | R/O | Indicates 64-bit addressing<br><br>0000: 32-bit addressing<br><br>0001: 64-bit addressing<br><br>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. |

### 14.1.21 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.<br><br>Reset to 0 |

### 14.1.22 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.<br><br>Reset to 0 |

### 14.1.23 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.<br><br>Reset to 0 |

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

| Bit | Function | Type | Description |
|---|---|---|---|
| 31:0 | 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.<br><br>Reset to 0 |

### 14.1.25 ECP 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. |

### 14.1.26 INTERRUPT LINE REGISTER – OFFSET 3Ch

| Bit | Function | Type | Description |
|---|---|---|---|
| 7:0 | Interrupt Line | R/W | For POST to program to FFh, indicating that the PI7C8150A does not implement an interrupt pin. |

### 14.1.27 INTERRUPT PIN REGISTER – OFFSET 3Ch

| Bit | Function | Type | Description |
|---|---|---|---|
| 15:8 | Interrupt Pin | R/O | Interrupt pin not supported on the PI7C8150A |

### 14.1.28 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.<br><br>0: ignore address and data parity errors on the secondary interface<br><br>1: enable parity error reporting and detection on the secondary interface<br><br>Reset to 0 |
| 17 | S_SERR_L enable | R/W | Controls the forwarding of S_SERR_L to the primary interface.<br><br>0: disable the forwarding of S_SERR_L to primary interface<br><br>1: enable the forwarding of S_SERR_L to primary interface<br><br>Reset to 0 |

| Bit | Function | Type | Description |
|---|---|---|---|
| 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 or PCI I/O space.<br><br>0: forward all I/O addresses in the range defined by the I/O base and I/O limit registers<br><br>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<br><br>Reset to 0 |
| 19 | VGA enable | R/W | Controls the bridge's response to VGA compatible addresses.<br><br>0: does not forward VGA compatible memory and I/O addresses from primary to secondary<br><br>1: forward VGA compatible memory and I/O addresses from primary to secondary regardless of other settings<br><br>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.<br><br>0: does not report master aborts (returns FFFF_FFFFh on reads and discards data on writes)<br><br>1: reports master aborts by signaling target abort if possible by the assertion of P_SERR_L if enabled<br><br>Reset to 0 |
| 22 | Secondary Interface Reset | R/W | Controls the assertion of S_RESET_L signal pin on the secondary interface<br><br>0: does not force the assertion of S_RESET_L pin<br><br>1: forces the assertion of S_RESET_L<br><br>Reset to 0 |
| 23 | Fast Back-to-Back Enable | R/W | Controls bridge's ability to generate fast back-to-back transactions to different devices on the secondary interface.<br><br>0: does not allow fast back-to-back transactions<br><br>1: enables fast back-to-back transactions<br><br>Reset to 0 |
| 24 | Primary Master Timeout | R/W | Set's the maximum number of PCI clocks the bridge will wait for an initiator on the primary to repeat a delayed transaction request. The counter starts right after the delayed transaction is at the front of the queue. If the master has not repeated at least once before the counter expires, the bridge discards the transaction from the queue.<br><br>0: $2^{15}$ PCI clocks<br><br>1: $2^{10}$ PCI clocks<br><br>Reset to 0 |

| Bit | Function | Type | Description |
|---|---|---|---|
| 25 | Secondary Master Timeout | R/W | Set's the maximum number of PCI clocks the bridge will wait for an initiator on the secondary to repeat a delayed transaction request. The counter starts right after the delayed transaction is at the front of the queue. If the master has not repeated at least once before the counter expires, the bridge discards the transaction from the queue.<br><br>0: $2^{15}$ PCI clocks<br><br>1: $2^{10}$ PCI clocks<br><br>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.<br><br>Reset to 0 |
| 27 | Discard Timer P_SERR_L enable | R/W | This bit is set to 1 and P_SERR_L is asserted when either the primary discard timer or the secondary discard timer expire.<br><br>Reset to 0 |
| 31-28 | Reserved | R/O | Reserved. Returns 0 when read. Reset to 0. |

## 14.1.29 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.<br><br>0: memory write disconnects at 4KB aligned address boundary<br><br>1: memory write disconnects at cache line aligned address boundary<br><br>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.<br><br>0: The bridge prefetches and does not forward byte enable bits during upstream memory reads.<br><br>1: The bridge requests only 1 DWORD from the target and forwards read byte enable bits during upstream memory reads.<br><br>Reset to 0 |
| 5 | Live Insertion Mode | R/W | Enables hardware control of transaction forwarding.<br><br>0: GPIO[3] has no effect on the I/O, memory, and master enable bits<br><br>1: If GPIO[3] is set to input mode, this bit enables GPIO[3] to mask I/O enable, memory enable and master enable bits to 0. PI7C8150A will stop accepting I/O and memory transactions as a result.<br><br>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.<br><br>0: PI7C8150A is ready for operation<br><br>1: Causes PI7C8150A to perform a chip reset |

| Bit | Function | Type | Description |
|---|---|---|---|
| 10:9 | Test Mode For All Counters at P and S1 | R/O | Controls the testability of the bridge's internal counters.<br>The bits are used for chip test only.<br><br>00: all bits are exercised<br><br>01: byte 1 is exercised<br><br>10: byte 2 is exercised<br><br>11: byte 3 is exercised<br><br>Reset to 0 |
| 15:11 | Reserved | R/O | Reserved. Returns 0 when read. Reset to 0. |

## 14.1.30 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.<br>Bits [24:16] correspond to request inputs S_REQ_L[8:0] respectively.<br>Bit 24 corresponds to S_REQ_L[8]<br>Bit 16 corresponds to S_REQ_L[0]<br><br>0: low priority<br><br>1: high priority<br><br>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.<br><br>0: low priority<br><br>1: high priority<br><br>Reset to 1 |
| 31:26 | Reserved | R/O | Reserved. Returns 0 when read. Reset to 0. |

## 14.1.31 EXTENDED CHIP CONTROL REGISTER – OFFSET 48h

| Bit | Function | Type | Description |
|---|---|---|---|
| 0 | Memory Read Flow Through Enable | R/W | Controls ability to do memory read flow through<br><br>0: Disable flow through during a memory read transaction<br><br>1: Enable flow through during a memory read transaction<br><br>Reset to 0 |
| 1 | Park | R/W | Controls bus arbiter's park function<br><br>0: Park to last master<br><br>1: Park to bridge<br><br>Reset to 0 |
| 15:2 | Reserved | R/O | Reserved. Returns 0 when read. Reset to 0 |

### 14.1.32 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 is the entire range except the down stream memory channel<br><br>1: Upstream memory is confined to upstream Memory Base and Limit (See offset 50th and 54th for upstream memory range)<br><br>Reset to 0 |
| 31:17 | Reserved | R/O | Reserved. Returns 0 when read. Reset to 0 |

### 14.1.33 SECONDARY BUS ARBITER PREEMPTION CONTROL REGISTER – OFFSET 4Ch

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

### 14.1.34 UPSTREAM (S TO P) MEMORY BASE REGISTER – OFFSET 50h

| Bit | Function | Type | Description |
|---|---|---|---|
| 3:0 | 64 bit addressing | R/O | 0: 32 bit addressing<br><br>1: 64 bit addressing<br><br>Reset to 1 |
| 15:4 | Upstream Memory Base Address | R/W | Controls upstream memory base address.<br><br>Reset to 00000000h |

### 14.1.35 UPSTREAM (S TO P) MEMORY LIMIT REGISTER – OFFSET 50h

| Bit | Function | Type | Description |
|---|---|---|---|
| 19:16 | 64 bit addressing | R/O | 0: 32 bit addressing<br><br>1: 64 bit addressing<br><br>Reset to 1 |
| 31:20 | Upstream Memory Limit Address | R/W | Controls upstream memory limit address.<br><br>Reset to 000FFFFFh |

### 14.1.36 UPSTREAM (S TO P) MEMORY BASE UPPER 32-BITS REGISTER – OFFSET 54h

| Bit | Function | Type | Description |
|---|---|---|---|
| 31:0 | Upstream Memory Base Address | R/W | Defines bits [63:32] of the upstream memory base<br><br>Reset to 0 |

### 14.1.37 UPSTREAM (S TO P) MEMORY LIMIT UPPER 32-BITS REGISTER – OFFSET 58h

| Bit | Function | Type | Description |
|---|---|---|---|
| 31:0 | Upstream Memory Limit Address | R/W | Defines bits [63:32] of the upstream memory limit<br><br>Reset to 0 |

### 14.1.38 P_SERR_L EVENT DISABLE REGISTER – OFFSET 64h

| Bit | Function | Type | Description |
|---|---|---|---|
| 0 | Reserved | R/O | Reserved. Returns 0 when read. Reset to 0 |
| 1 | Posted Write Parity Error | R/W | Controls PI7C8150A's ability to assert P_SERR_L when it is unable to transfer any read data from the target after $2^{24}$ attempts.<br><br>0: P_SERR_L is asserted if this event occurs and the SERR_L enable bit in the command register is set.<br><br>1: P_SERR_L is not assert if this event occurs.<br><br>Reset to 0 |
| 2 | Posted Write Non-Delivery | R/W | Controls PI7C8150A's ability to assert P_SERR_L when it is unable to transfer delayed write data after $2^{24}$ attempts.<br><br>0: P_SERR_L is asserted if this event occurs and the SERR_L enable bit in the command register is set<br><br>1: P_SERR_L is not asserted if this event occurs<br><br>Reset to 0 |

| Bit | Function | Type | Description |
|---|---|---|---|
| 3 | Target Abort During Posted Write | R/W | Controls PI7C8150A's ability to assert P_SERR_L when it receives a target abort when attempting to deliver posted write data.<br><br>0: P_SERR_L is asserted if this event occurs and the SERR_L enable bit in the command register is set<br><br>1: P_SERR_L is not asserted if this event occurs<br><br>Reset to 0 |
| 4 | Master Abort On Posted Write | R/W | Controls PI7C8150A's ability to assert P_SERR_L when it receives a master abort when attempting to deliver posted write data.<br><br>0: P_SERR# is asserted if this event occurs and the SERR# enable bit in the command register is set<br><br>1: P_SERR# is not asserted if this event occurs<br><br>Reset to 0 |
| 5 | Delayed Write Non-Delivery | R/W | Controls PI7C8150A's ability to assert P_SERR# when it is unable to transfer delayed write data after $2^{24}$ attempts.<br><br>0: P_SERR_L is asserted if this event occurs and the SERR_L enable bit in the command register is set<br><br>1: P_SERR_L is not asserted if this event occurs<br><br>Reset to 0 |
| 6 | Delayed Read – No Data From Target | R/W | Controls PI7C8150A's ability to assert P_SERR_L when it is unable to transfer any read data from the target after $2^{24}$ attempts.<br><br>0: P_SERR_L is asserted if this event occurs and the SERR_L enable bit in the command register is set<br><br>1: P_SERR_L is not asserted if this event occurs<br><br>Reset to 0 |
| 7 | Reserved | R/O | Reserved. Returns 0 when read. Reset to 0 |

## 14.1.39 GPIO DATA AND CONTROL REGISTER – OFFSET 64h

| Bit | Function | Type | Description |
|---|---|---|---|
| 11:8 | GPIO Output Write-1-to-Clear | R/WC | Writing 1 to any of these bits drives the corresponding bit LOW on the GPIO[3:0] bus if it is programmed as bidirectional. Data is driven on the PCI clock cycle following completion of the configuration write to this register. Bit positions corresponding to GPIO pins that are programmed as input only are not driven. Writing 0 has no effect and will show last the last value written when read.<br><br>Reset to 0. |
| 15:12 | GPIO Output Write-1-to-Set | R/WS | Writing 1 to any of these bits drives the corresponding bit HIGH on the GPIO[3:0] bus if it is programmed as bidirectional. Data is driven on the PCI clock cycle following completion of the configuration write to this register. Bit positions corresponding to GPIO pins that are programmed as input only are not driven. Writing 0 has no effect and will show last the last value written when read.<br><br>Reset to 0. |

| Bit | Function | Type | Description |
|---|---|---|---|
| 19:16 | GPIO Output Enable Write-1-to-Clear | R/WC | Writing 1 to and of these bits configures the corresponding GPIO[3:0] pin as an input only. The output driver is tristated. Writing 0 to this register has no effect and will reflect the last value written when read.<br><br>Reset to 0. |
| 23:20 | GPIO Output Enable Write-1-to-Set | R/WS | Writing 1 to and of these bits configures the corresponding GPIO[3:0] pin as bidirectional. The output driver is enabled and drives the value set in the output data register (65h). Writing 0 to this register has no effect and will reflect the last value written when read.<br><br>Reset to 0. |
| 27:24 | Reserved | R | Reserved. Returns 0 when read. Reset to 0. |
| 31:28 | GPIO Input Data Register | R/O | Reads the state of the GPIO[3:0] pins. The state is updated on the PCI clock following a change in the GPIO[3:0] pins. |

## 14.1.40 SECONDARY CLOCK CONTROL REGISTER – OFFSET 68h

| Bit | Function | Type | Description |
|---|---|---|---|
| 1:0 | Clock 0 disable | R/W | If either bit is 0, then S_CLKOUT [0] is enabled.<br>If both bits are 1, then S_CLKOUT [0] is disabled. |
| 3:2 | Clock 1 disable | R/W | If either bit is 0, then S_CLKOUT [1] is enabled.<br>If both bits are 1, then S_CLKOUT [1] is disabled. |
| 5:4 | Clock 2 disable | R/W | If either bit is 0, then S_CLKOUT [2] is enabled.<br>If both bits are 1, then S_CLKOUT [2] is disabled. |
| 7:6 | Clock 3 disable | R/W | If either bit is 0, then S_CLKOUT [3] is enabled.<br>If both bits are 1, then S_CLKOUT [3] is disabled. |
| 8 | Clock 4 disable | R/W | If bit is 0, then S_CLKOUT [4] is enabled.<br>If bit is 1, then S_CLKOUT [4] is disabled and driven low. |
| 9 | Clock 5 disable | R/W | If bit is 0, then S_CLKOUT [5] is enabled.<br>If bit is 1, then S_CLKOUT [5] is disabled and driven low. |
| 10 | Clock 6 disable | R/W | If bit is 0, then S_CLKOUT [6] is enabled.<br>If bit is 1, then S_CLKOUT [6] is disabled and driven low. |
| 11 | Clock 7 disable | R/W | If bit is 0, then S_CLKOUT [7] is enabled.<br>If bit is 1, then S_CLKOUT [7] is disabled and driven low. |
| 12 | Clock 8 disable | R/W | If bit is 0, then S_CLKOUT [8] is enabled.<br>If bit is 1, then S_CLKOUT [8] is disabled and driven low. |
| 13 | Clock 9 disable | R/W | If bit is 0, then S_CLKOUT [9] is enabled.<br>If bit is 1, then S_CLKOUT [9] is disabled and driven low. |
| 15:14 | Reserved | RO | Reserved. Returns 00 when read. |

## 14.1.41 P_SERR_L STATUS REGISTER – OFFSET 68h

| Bit | Function | Type | Description |
|---|---|---|---|
| 16 | Address Parity Error | R/WC | 1: Signal P_SERR_L was asserted because an address parity error was detected on P or S bus.<br><br>Reset to 0 |
| 17 | Posted Write Data Parity Error | R/WC | 1: Signal P_SERR_L was asserted because a posted write data parity error was detected on the target bus.<br><br>Reset to 0 |
| 18 | Posted Write Non-delivery | R/WC | 1: Signal P_SERR_L was asserted because the bridge was unable to deliver post memory write data to the target after $2^{24}$ attempts.<br><br>Reset to 0 |

| Bit | Function | Type | Description |
|---|---|---|---|
| 19 | Target Abort during Posted Write | R/WC | 1: Signal P_SERR_L was asserted because the bridge received a target abort when delivering post memory write data.<br><br>Reset to 0. |
| 20 | Master Abort during Posted Write | R/WC | 1: Signal P_SERR_L was asserted because the bridge received a master abort when attempting to deliver post memory write data<br><br>Reset to 0. |
| 21 | Delayed Write Non-delivery | R/WC | 1: Signal P_SERR_L was asserted because the bridge was unable to deliver delayed write data after $2^{24}$ attempts.<br><br>Reset to 0 |
| 22 | Delayed Read – No Data from Target | R/WC | 1: Signal P_SERR_L was asserted because the bridge was unable to read any data from the target after $2^{24}$ attempts.<br><br>Reset to 0. |
| 23 | Delayed Transaction Master Timeout | R/WC | 1: Signal P_SERR_L was asserted because a master did not repeat a read or write transaction before master timeout.<br><br>Reset to 0. |

## 14.1.42 PORT OPTION REGISTER – OFFSET 74h

| Bit | Function | Type | Description |
|---|---|---|---|
| 0 | Reserved | R/O | Reserved. Returns 0 when read. Reset to 0. |
| 1 | Primary MEMR Command Alias Enable | R/W | Controls PI7C8150A's detection mechanism for matching memory read retry cycles from the initiator on the primary interface<br><br>0: exact matching for non-posted memory write retry cycles from initiator on the primary interface<br><br>1: alias MEMRL or MEMRM to MEMR for memory read retry cycles from the initiator on the primary interface<br><br>Reset to 0 |
| 2 | Primary MEMW Command Alias Enable | R/W | Controls PI7C8150A's detection mechanism for matching non-posted memory write retry cycles from the initiator on the primary interface<br><br>0: exact matching for non-posted memory write retry cycles from initiator on the primary interface<br><br>1: alias MEMWI to MEMW for non-posted memory write retry cycles from initiator on the primary interface<br><br>Reset to 0 |
| 3 | Secondary MEMR Command Alias Enable | R/W | Controls PI7C8150A's detection mechanism for matching memory read retry cycles from the initiator on the secondary<br><br>0: exact matching for memory read retry cycles from initiator on the secondary interface<br><br>1: alias MEMRL or MEMRM to MEMR for memory read retry cycles from initiator on the secondary interface<br><br>Reset to 0 |

| Bit | Function | Type | Description |
|---|---|---|---|
| 4 | Secondary MEMW Command Alias Enable | R/W | Controls PI7C8150A's detection mechanism for matching non-posted memory write retry cycles from the initiator on the primary interface<br><br>0: exact matching for non-posted memory write retry cycles from initiator on the secondary interface<br><br>1: alias MEMWI to MEMW for non-posted memory write retry cycles from initiator on the secondary interface<br><br>Reset to 0 |
| 5:6 | Reserved | R/O | Reserved. Returns 0 when read. Reset to 0. |
| 7 | Primary MEMWI Command Alias Enable | R/W | Controls PI7C8150A's detection mechanism for matching non-posted memory write and invalidate cycles from the initiator on the primary interface<br><br>0: When accepting MEMWI command at the primary interface, PI7C8150A converts MEMWI to MEMW command on the secondary interface<br><br>1: Disconnects MEMWI command at aligned cache line boundaries |
| 8 | Secondary MEMWI Command Alias Enable | R/W | Controls PI7C8150A's detection mechanism for matching non-posted memory write and invalidate cycles from the initiator on the secondary interface<br><br>0: When accepting MEMWI command at the secondary interface, PI7C8150A converts MEMWI to MEMW command on the primary interface<br><br>1: Disconnects MEMWI command at aligned cache line boundaries |
| 9 | Enable Long Request | R/W | Controls PI7C8150A's ability to enable long requests for lock cycles<br><br>0: normal lock operation<br><br>1: enable long request for lock cycle<br><br>Reset to 0 |
| 10 | Enable Secondary To Hold Request Longer | R/W | Controls PI7C8150A's ability to enable the secondary bus to hold requests longer.<br><br>0: internal secondary master will release REQ_L after FRAME_L assertion<br><br>1: internal secondary master will hold REQ_L until there is no transactions pending in FIFO or until terminated by target<br><br>Reset to 1 |
| 11 | Enable Primary To Hold Request Longer | R/W | Controls PI7C8150A's ability to hold requests longer at the Primary Port.<br><br>0: internal Primary master will release REQ_L after FRAME_L assertion<br><br>1: internal Primary master will hold REQ_L until there is no transactions pending in FIFO or until terminated by target<br><br>Reset to 1 |
| 15:12 | Reserved | R/O | Reserved. Returns 0 when read. Reset to 0. |

### 14.1.43 RETRY COUNTER REGISTER – OFFSET 78h

| Bit | Function | Type | Description |
|---|---|---|---|
| 31:0 | Retry Counter | R/W | Holds the maximum number of attempts that PI7C8150A will try before reporting retry timeout. Retry count set at $2^{24}$ PCI clocks. Default is 0100 0000h. |

### 14.1.44 PRIMARY MASTER TIMEOUT COUNTER – OFFSET 80h

| Bit | Function | Type | Description |
|---|---|---|---|
| 15:0 | Primary Timeout | R/W | Primary timeout occurs after $2^{15}$ PCI clocks.<br><br>Reset to 8000h. |

### 14.1.45 SECONDARY MASTER TIMEOUT COUNTER – OFFSET 80h

| Bit | Function | Type | Description |
|---|---|---|---|
| 31:16 | Secondary Timeout | R/W | Secondary timeout occurs after $2^{15}$ PCI clocks.<br><br>Reset to 8000h. |

### 14.1.46 CAPABILITY ID REGISTER – OFFSET B0h

| Bit | Function | Type | Description |
|---|---|---|---|
| 7:0 | Capability ID | R/O | Capability ID for slot identification<br><br>00h: Reserved<br><br>01h: PCI Power Management (PCIPM)<br><br>02h: Accelerated Graphics Port (AGP)<br><br>03h: Vital Product Data (VPD)<br><br>04h: Slot Identification (SI)<br><br>05h: Message Signaled Interrupts (MSI)<br><br>06h: Compact PCI Hot Swap (CHS)<br><br>07h – 255h: Reserved<br><br>Reset to 04h |

### 14.1.47 NEXT POINTER REGISTER – OFFSET B0h

| Bit | Function | Type | Description |
|---|---|---|---|
| 15:8 | Next Pointer | R/O | Reset to 0000 0000: next pointer (00h if MS0=0 and MS1=1, or MS0=1) |

### 14.1.48 SLOT NUMBER REGISTER – OFFSET B0h

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

### 14.1.49 CHASSIS NUMBER REGISTER – OFFSET B0h

| Bit | Function | Type | Description |
|---|---|---|---|
| 31:24 | Chassis Number Register | R/W | Chassis number register.<br><br>Reset to 0 |

### 14.1.50 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. |

### 14.1.51 NEXT ITEM POINTER REGISTER – OFFSET DCh

| Bit | Function | Type | Description |
|---|---|---|---|
| 15:8 | Next Item Pointer | R/O | Points to slot number register (0Bh). |

### 14.1.52 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 PI7C8150A does not support the PME# pin. |
| 20 | Auxiliary Power | R/O | Read as 0 to indicate PI7C8150A does not support the PME# pin or an auxiliary power source. |
| 21 | Device Specific Initialization | R/O | Read as 0 to indicate PI7C8150A 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 PI7C8150A does not support the D1 power management state. |
| 26 | D2 Power State Support | R/O | Read as 0 to indicate PI7C8150A does not support the D2 power management state. |
| 31:27 | PME# Support | R/O | Read as 0 to indicate PI7C8150A does not support the PME# pin. |

### 14.1.53 POWER MANAGEMENT DATA REGISTER – OFFSET E0h

| Bit | Function | Type | Description |
|---|---|---|---|
| 1:0 | Power State | R/W | Indicates the current power state of PI7C8150A. If an unimplemented power state is written to this register, PI7C8150A 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_L<br><br>00: D0 state<br><br>01: not implemented<br><br>10: not implemented<br><br>11: D3 state<br><br>Reset to 0 |
| 7:2 | Reserved | R/O | Read as 0 |
| 8 | PME# Enable | R/O | Read as 0 as PI7C8150A does not support the PME# pin. |
| 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. |

### 14.1.54 CAPABILITY ID REGISTER – OFFSET E4h

| Bit | Function | Type | Description |
|---|---|---|---|
| 7:0 | Capability ID | R/O | 00h: Reserved.<br><br>01h: PCI Power Management (PCIPM)<br><br>02h: Accelerated Graphics Port (AGP)<br><br>03h: Vital Product Data (VPD)<br><br>04h: Slot Identification (SI)<br><br>05h: Message Signaled Interrupts (MSI)<br><br>06h: Compact PCI Hot Swap<br><br>07h-255h: Reserved |

### 14.1.55 NEXT POINTER REGISTER – OFFSET E4h

| Bit | Function | Type | Description |
|---|---|---|---|
| 15:8 | Next Pointer | R/O | End of pointer (00h) |

# 15 BRIDGE BEHAVIOR

A PCI cycle is initiated by asserting the FRAME_L 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 | PI7C8150A does not respond. It detects this situation by decoding the address as well as monitoring the P_DEVSEL_L for other fast and medium devices on the Primary Port. |
| Master on Primary | Target on Secondary | PI7C8150A asserts P_DEVSEL_L, 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 | PI7C8150A does not respond and the cycle will terminate as master abort. |
| Master on Secondary | Target on the same Secondary Port | PI7C8150A does not respond. |
| Master on Secondary | Target on Primary or the other Secondary Port | PI7C8150A asserts S_DEVSEL_L, 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 | PI7C8150A does not respond. |

## 15.2 ABNORMAL TERMINATION (INITIATED BY BRIDGE MASTER)

### 15.2.1 MASTER ABORT

Master abort indicates that when PI7C8150A acts as a master and receives no response (i.e., no target asserts DEVSEL_L or S_DEVSEL_L) from a target, the bridge de-asserts FRAME_L and then de-asserts IRDY_L.

### 15.2.2 PARITY AND ERROR REPORTING

Parity must be checked for all addresses and write data. Parity is defined on the P_PAR, and S_PAR 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_L signal by asserting P_SERR_L for one cycle and then 3-stating two cycles after the bad address. P_SERR_L 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_L 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_L 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 PI7C8150A 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 PI7C8150A.

# 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 PI7C8150A for board-level continuity test and diagnostics. The TAP pins assigned are TCK, TDI, TDO, TMS and TRST_L. 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.

PI7C8150A 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.

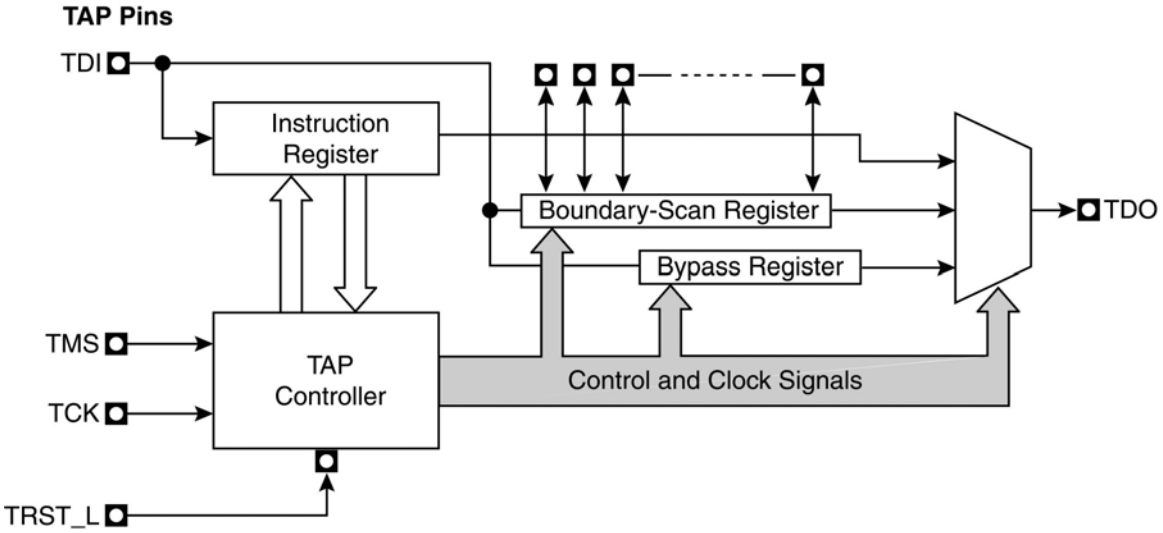


**Figure 16-1 Test Access Port Block Diagram**

## 16.1.1 TAP PINS

The PI7C8150A's TAP pins form a serial port composed of four input connections (TMS, TCK, TRST_L 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_L 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 PI7C8150A supports three mandatory boundary-scan instructions (BYPASS, SAMPLE and EXTEST). The table shown below lists the PI7C8150A's boundary-scan instruction codes.

**Table 16-1. TAP Pins**

| Instruction / Requisite | Opcode (binary) | Description |
|---|---|---|
| EXTEST<br>IEEE 1149.1<br>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<br>IEEE 1149.1<br>Required | 0001 | SAMPLE performs two functions:<br>▪ 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.<br>▪ 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 PI7C8150A 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 PI7C8150A.

## 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 16-1 shows the bit order of the PI7C8150A 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 PI7C8150A'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_L 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 | Pin Number | Type |
|---|---|---|---|
| 0 | S_AD[0] | 137 | BIDIR |
| 1 | S_AD[1] | 138 | BIDIR |
| 2 | S_AD[2] | 140 | BIDIR |
| 3 | S_AD[3] | 141 | BIDIR |
| 4 | S_AD[4] | 143 | BIDIR |
| 5 | S_AD[5] | 144 | BIDIR |
| 6 | S_AD[6] | 146 | BIDIR |
| 7 | S_AD[7] | 147 | BIDIR |
| 8 | S_CBE[0] | 149 | BIDIR |
| 9 | S_AD[8] | 150 | BIDIR |
| 10 | S_AD[9] | 152 | BIDIR |
| 11 | S_M66EN | 153 | OUTPUT |
| 12 | S_AD[10] | 154 | BIDIR |
| 13 | S_AD[11] | 159 | BIDIR |
| 14 | S_AD[12] | 161 | BIDIR |
| 15 | S_AD[13] | 162 | BIDIR |
| 16 | S_AD[14] | 164 | BIDIR |
| 17 | S_AD[15] | 165 | BIDIR |
| 18 | | | CONTROL |
| 19 | S_CBE[1] | 167 | BIDIR |
| 20 | S_PAR | 168 | BIDIR |
| 21 | S_SERR_L | 169 | INPUT |
| 22 | S_PERR_L | 171 | BIDIR |
| 23 | S_LOCK_L | 172 | BIDIR |
| 24 | S_STOP_L | 173 | BIDIR |
| 25 | | | CONTROL |
| 26 | S_DEVSEL_L | 175 | BIDIR |
| 27 | S_TRDY_L | 176 | BIDIR |
| 28 | S_IRDY_L | 177 | BIDIR |
| 29 | S_FRAME_L | 179 | BIDIR |
| 30 | S_CBE[2] | 180 | BIDIR |
| 31 | S_AD[16] | 182 | BIDIR |
| 32 | S_AD[17] | 183 | BIDIR |
| 33 | S_AD[18] | 185 | BIDIR |
| 34 | S_AD[19] | 186 | BIDIR |
| 35 | S_AD[20] | 188 | BIDIR |
| 36 | S_AD[21] | 189 | BIDIR |
| 37 | S_AD[22] | 191 | BIDIR |
| 38 | S_AD[23] | 192 | BIDIR |
| 39 | S_CBE[3] | 194 | BIDIR |
| 40 | S_AD[24] | 195 | BIDIR |
| 41 | S_AD[25] | 197 | BIDIR |
| 42 | S_AD[26] | 198 | BIDIR |
| 43 | S_AD[27] | 200 | BIDIR |
| 44 | S_AD[28] | 201 | BIDIR |
| 45 | S_AD[29] | 203 | BIDIR |
| 46 | S_AD[30] | 204 | BIDIR |
| 47 | | | CONTROL |

| Boundary-Scan Register Number | Pin Name | Pin Number | Type |
|---|---|---|---|
| 48 | S_AD[31] | 206 | BIDIR |
| 49 | S_REQ_L[0] | 207 | INPUT |
| 50 | S_REQ_L[1] | 2 | INPUT |
| 51 | S_REQ_L[2] | 3 | INPUT |
| 52 | S_REQ_L[3] | 4 | INPUT |
| 53 | S_REQ_L[4] | 5 | INPUT |
| 54 | S_REQ_L[5] | 6 | INPUT |
| 55 | S_REQ_L[6] | 7 | INPUT |
| 56 | S_REQ_L[7] | 8 | INPUT |
| 57 | S_REQ_L[8] | 9 | INPUT |
| 58 | S_GNT_L[0] | 10 | OUTPUT |
| 59 | S_GNT_L[1] | 11 | OUTPUT |
| 60 | | | CONTROL |
| 61 | S_GNT_L[2] | 13 | OUTPUT |
| 62 | S_GNT_L[3] | 14 | OUTPUT |
| 63 | S_GNT_L[4] | 15 | OUTPUT |
| 64 | S_GNT_L[5] | 16 | OUTPUT |
| 65 | S_GNT_L[6] | 17 | OUTPUT |
| 66 | S_GNT_L[7] | 18 | OUTPUT |
| 67 | S_GNT_L[8] | 19 | OUTPUT |
| 68 | S_CLKIN | 21 | INPUT |
| 69 | S_RESET_L | 22 | OUTPUT |
| 70 | S_CFN_L | 23 | INPUT |
| 71 | GPIO[3] | 24 | BIDIR |
| 72 | GPIO[2] | 25 | BIDIR |
| 73 | GPIO[1] | 27 | BIDIR |
| 74 | GPIO[0] | 28 | BIDIR |
| 75 | S_CLKOUT[0] | 29 | OUTPUT |
| 76 | S_CLKOUT[1] | 30 | OUTPUT |
| 77 | | | CONTROL |
| 78 | S_CLKOUT[2] | 32 | OUTPUT |
| 79 | S_CLKOUT[3] | 33 | OUTPUT |
| 80 | S_CLKOUT[4] | 35 | OUTPUT |
| 81 | S_CLKOUT[5] | 36 | OUTPUT |
| 82 | S_CLKOUT[6] | 38 | OUTPUT |
| 83 | S_CLKOUT[7] | 39 | OUTPUT |
| 84 | S_CLKOUT[8] | 41 | OUTPUT |
| 85 | S_CLKOUT[9] | 42 | OUTPUT |
| 86 | P_RESET_L | 43 | INPUT |
| 87 | BPCCE | 44 | INPUT |
| 88 | P_CLK | 45 | INPUT |
| 89 | P_GNT_L | 46 | INPUT |
| 90 | P_REQ_L | 47 | OUTPUT |
| 91 | | | CONTROL |
| 92 | P_AD[31] | 49 | BIDIR |
| 93 | P_AD[30] | 50 | BIDIR |
| 94 | P_AD[29] | 55 | BIDIR |
| 95 | P_AD[28] | 57 | BIDIR |
| 96 | P_AD[27] | 58 | BIDIR |
| 97 | P_AD[26] | 60 | BIDIR |
| 98 | P_AD[25] | 61 | BIDIR |
| 99 | P_AD[24] | 63 | BIDIR |
| 100 | P_CBE[3] | 64 | BIDIR |
| 101 | P_IDSEL | 65 | INPUT |
| 102 | P_AD[23] | 67 | BIDIR |
| 103 | P_AD[22] | 68 | BIDIR |
| 104 | P_AD[21] | 70 | BIDIR |
| 105 | P_AD[20] | 71 | BIDIR |
| 106 | P_AD[19] | 73 | BIDIR |
| 107 | P_AD[18] | 74 | BIDIR |
| 108 | P_AD[17] | 76 | BIDIR |

| Boundary-Scan Register Number | Pin Name | Pin Number | Type |
|---|---|---|---|
| 109 | P_AD[16] | 77 | BIDIR |
| 110 | | | CONTROL |
| 111 | P_CBE[2] | 79 | BIDIR |
| 112 | P_FRAME_L | 80 | BIDIR |
| 113 | P_IRDY_L | 82 | BIDIR |
| 114 | P_TRDY_L | 83 | BIDIR |
| 115 | P_DEVSEL_L | 84 | BIDIR |
| 116 | P_STOP_L | 85 | BIDIR |
| 117 | | | CONTROL |
| 118 | P_LOCK_L | 87 | INPUT |
| 119 | P_PERR_L | 88 | BIDIR |
| 120 | P_SERR_L | 89 | OUTPUT |
| 121 | P_PAR | 90 | BIDIR |
| 122 | P_CBE[1] | 92 | BIDIR |
| 123 | P_AD[15] | 93 | BIDIR |
| 124 | P_AD[14] | 95 | BIDIR |
| 125 | P_AD[13] | 96 | BIDIR |
| 126 | P_AD[12] | 98 | BIDIR |
| 127 | P_AD[11] | 99 | BIDIR |
| 128 | P_AD[10] | 101 | BIDIR |
| 129 | P_M66EN | 102 | INPUT |
| 130 | P_AD[9] | 107 | BIDIR |
| 131 | P_AD[8] | 109 | BIDIR |
| 132 | P_CBE[0] | 110 | BIDIR |
| 133 | P_AD[7] | 112 | BIDIR |
| 134 | P_AD[6] | 113 | BIDIR |
| 135 | P_AD[5] | 115 | BIDIR |
| 136 | P_AD[4] | 116 | BIDIR |
| 137 | P_AD[3] | 118 | BIDIR |
| 138 | P_AD[2] | 119 | BIDIR |
| 139 | P_AD[1] | 121 | BIDIR |
| 140 | P_AD[0] | 122 | BIDIR |
| 141 | | | CONTROL |
| 142 | CFG66 | 125 | INPUT |
| 143 | MSK_IN | 126 | 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 | 0°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 |

**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 |
|---|---|---|---|---|---|---|
| $V_{DD}$, $AV_{CC}$ | Supply Voltage | | 3 | 3.6 | V | |
| $V_{ih}$ | Input HIGH Voltage | | 0.5 $V_{DD}$ | $V_{DD}$ + 0.5 | V | 3, 4 |
| $V_{il}$ | Input LOW Voltage | | -0.5 | 0.3 $V_{DD}$ | V | 3, 4 |
| $V_{ih}$ | CMOS Input HIGH Voltage | | 0.7 $V_{DD}$ | $V_{DD}$ + 0.5 | V | 1, 4 |
| $V_{il}$ | CMOS Input LOW Voltage | | -0.5 | 0.3 $V_{DD}$ | V | 1, 4 |
| $V_{ipu}$ | Input Pull-up Voltage | | 0.7 $V_{DD}$ | | V | 3 |
| $I_{il}$ | Input Leakage Current | $0 < V_{in} < V_{DD}$ | | ±10 | µA | 3 |
| $V_{oh}$ | Output HIGH Voltage | $I_{out}$ = -500µA | 0.9$V_{DD}$ | | V | 3 |
| $V_{ol}$ | Output LOW Voltage | $I_{out}$ = 1500µA | | 0.1 $V_{DD}$ | V | 3 |
| $V_{oh}$ | CMOS Output HIGH Voltage | $I_{out}$ = -500µA | $V_{DD}$ – 0.5 | | V | 2 |
| $V_{ol}$ | CMOS Output LOW Voltage | $I_{out}$ = 1500µA | | 0.5 | V | 2 |
| $C_{in}$ | Input Pin Capacitance | | | 10 | pF | 3 |
| $C_{CLK}$ | CLK Pin Capacitance | | 5 | 12 | pF | 3 |
| $C_{IDSEL}$ | IDSEL Pin Capacitance | | | 8 | pF | 3 |
| $L_{pin}$ | Pin Inductance | | | 20 | nH | 3 |

**Notes:**
1. CMOS Input pins: S_CFN_L, TCK, TMS, TDI, TRST_L, SCAN_EN, SCAN_TM_L

2. CMOS Output pin: TDO

3. PCI pins: P_AD[31:0], P_CBE[3:0], P_PAR, P_FRAME_L, P_IRDY_L, P_TRDY_L, P_DEVSEL_L, P_STOP_L, P_LOCK_L, PIDSEL_L, P_PERR_L, P_SERR_L, P_REQ_L, P_GNT_L, P_RESET_L, S_AD[31:0], S_CBE[3:0], S_PAR, S_FRAME_L, S_IRDY_L, S_TRDY_L, S_DEVSEL_L, S_STOP_L, S_LOCK_L, S_PERR_L, S_SERR_L, S_REQ[7:0]_L, S_GNT[7:0]_L, S_RESET_L, S_EN, HSLED, HS_SW_L, HS_EN, ENUM_L.

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

## 17.3 AC SPECIFICATIONS

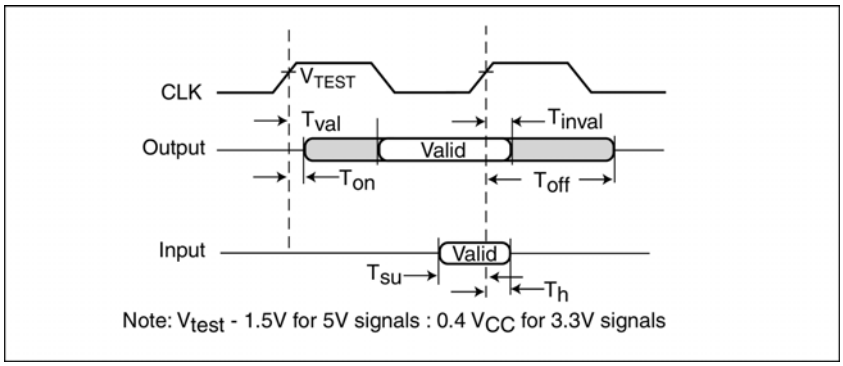


**Figure 17-1 PCI Signal Timing Measurement Conditions**

| | | 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_L, S_REQ_L[7:0], P_GNT_L, S_GNT_L[7:0], HSLED, HS_SW_L, HS_EN, and ENUM_L. Bused signals are P_AD, P_BDE_L, P_PAR, P_PERR_L, P_SERR_L, P_FRAME_L, P_IRDY_L, P_TRDY_L, P_LOCK_L, P_DEVSEL_L, P_STOP_L, P_IDSEL, S_AD, S_CBE_L, S_PAR, S_PERR_L, S_SERR_L, S_FRAME_L, S_IRDY_L, S_TRDY_L, S_LOCK_L, S_DEVSEL_L, and S_STOP_L.

4. REQ_L signals have a setup of 10 and GNT_L signals have a setup of 12.

## 17.4 66MHZ 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 | 3.14 | 5.07 | |
| $T_{CYCLE}$ | 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 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 | 3.14 | 5.07 | |
| $T_{CYCLE}$ | 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 POWER CONSUMPTION

| Parameter | Typical | Units |
|---|---|---|
| Power Consumption at 66MHz | 1.68 | W |
| Supply Current, $I_{CC}$ | 510 | mA |

# 18 PACKAGE INFORMATION

## 18.1 208-PIN FQFP PACKAGE DIAGRAM

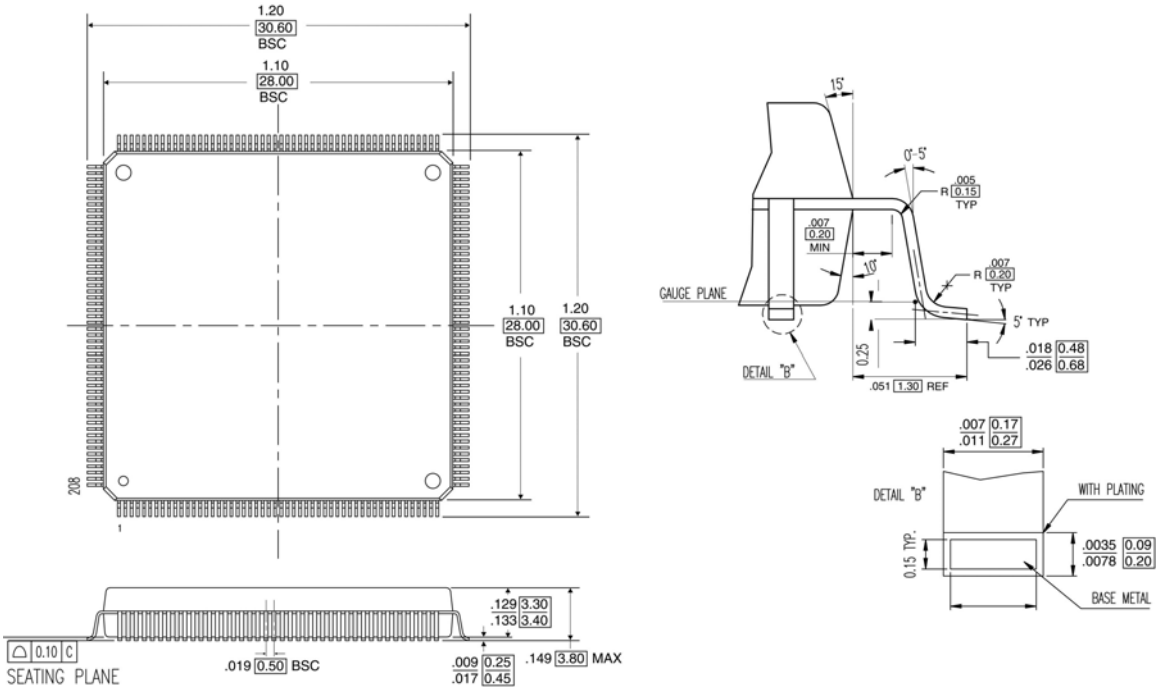


**Figure 18-1 208-pin FQFP Package Outline**

## 18.2 256-BALL PBGA PACKAGE DIAGRAM

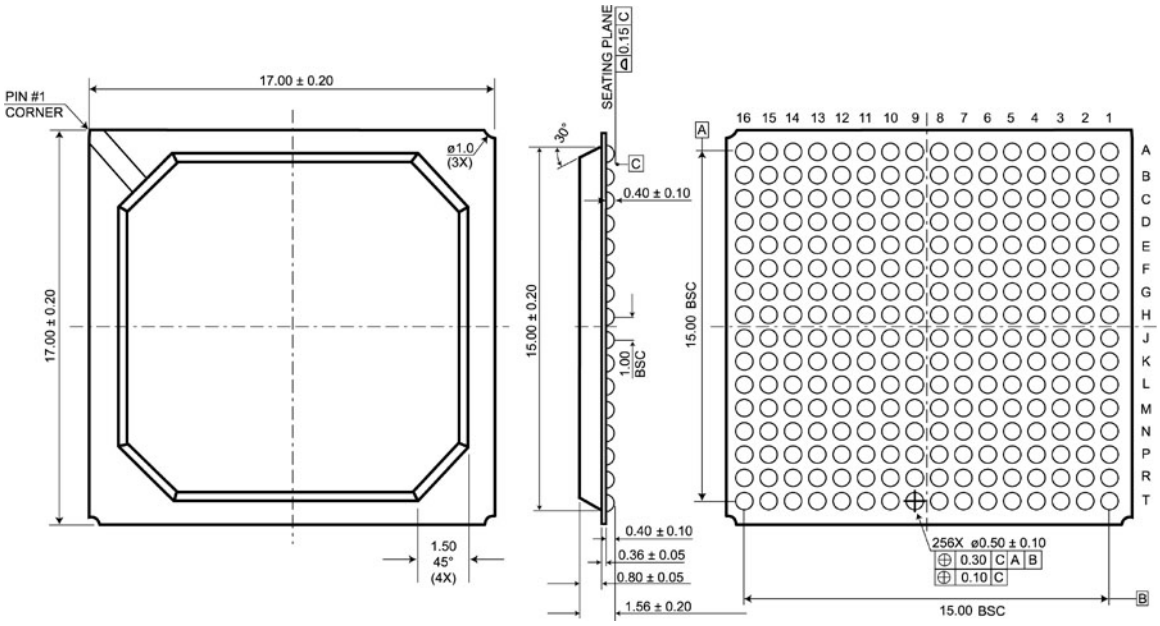


**Figure 18-2 256-pin PBGA Package Outline**

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

## 18.3 PART NUMBER ORDERING INFORMATION

| Part Number | Speed | Pin – Package | Temperature |
|---|---|---|---|
| PI7C8150AMA | 66MHz | 208 – FQFP | 0°C to 85°C |
| PI7C8150AND | 66MHz | 256 – PBGA | 0°C to 85°C |
| PI7C8150AMA-33 | 33MHz | 208 – FQFP | 0°C to 85°C |
| PI7C8150AND-33 | 33MHz | 256 – PBGA | 0°C to 85°C |
| PI7C8150AMAE | 66MHz | 208 – FQFP (Pb-free & Green) | 0°C to 85°C |
| PI7C8150ANDE | 66MHz | 256 – PBGA (Pb-free & Green) | 0°C to 85°C |
| PI7C8150AMAE-33 | 33MHz | 208 – FQFP (Pb-free & Green) | 0°C to 85°C |

*NOTES:*