PI7C8152A & PI7C8152B

2-Port PCI-to-PCI Bridge

REVISION 1.11

2380 Bering Drive, San Jose, CA 95131
Telephone: 1-877-PERICOM, (1-877-737-4266)
Fax: 408-435-1100
Email: solutions@pericom.com
Internet: http://www.pericom.com

14-0055

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
08/13/03	0.01	First draft of datasheet
08/14/03	0.02	Revised bit[4] offset 48h
09/19/03	1.00	Datasheet release to the web Revised revision ID register bit[7:0] offset 08h from 0h to 1h
09/25/03	1.10	Added descriptions for 8152A synchronous device Revised T_{DELAY} in sections 14.4 and 14.5 Revised Power Consumption in section 14.7
10/16/03	1.11	Revised Product Description in section 1. Revised pin description for S_CLKIN in section 2.2.3. Corrected Primary Clock Input description in section 9.1.

This page intentionally left blank.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

This page intentionally left blank.

This page intentionally left blank.

1 INTRODUCTION

Product Description

The PI7C8152A and PI7C8152B (PI7C8152x) are Pericom Semiconductor's PCI-to-PCI Bridge that are designed to be fully compliant with the 32-bit, 66MHz implementation of the *PCI Local Bus Specification, Revision 2.2*. The PI7C8152B supports both synchronous and asynchronous bus transactions between devices on the Primary Bus and the Secondary Buses operating up to 66MHz. The PI7C8152A supports synchronous transactions only. In synchronous mode, both buses must operate at the same frequency. The Primary and Secondary Bus can also operate in concurrent mode, resulting in added increase in system performance.

Product Features

- 32-bit Primary and Secondary Ports run up to 66MHz
- Compliant with the *PCI Local Bus Specification, Revision 2.2*
- Compliant with PCI-to-PCI Bridge Architecture Specification, Revision 1.1.
  - All I/O and memory commands
  - Type 1 to Type 0 configuration conversion
  - Type 1 to Type 1 configuration forwarding
  - Type 1 configuration write to special cycle conversion
- Compliant with the *Advanced Configuration Power Interface (ACPI) Specification.*
- Compliant with the *PCI Power Management Specification, Revision 1.1.*
- Synchronous and Asynchronous operation support

  - Supported modes of Asynchronous operation **(PI7C8152B ONLY)**

Primary	Secondary
25MHz to 66MHz	25MHz to 66MHz

  - Supported modes of Synchronous operation

Primary	Secondary
66MHz	66MHz
50MHz	50MHz
33MHz	33MHz
25MHz	25MHz

- Provides internal arbitration for four 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
- 32-bit I/O address range
- 32-bit memory-mapped I/O address range
- 64-bit prefetchable address range
- ISA-aware mode for legacy support in the first 64KB of I/O address range
- Extended commercial temperature range 0°C to 85°C
- 3.3V core; 3.3V and 5V signaling
- 160-pin MQFP package

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 #	Type	Description
P_AD[31:0]	70, 72, 73, 74, 76, 77, 78, 79, 84, 85, 87, 88, 89, 91, 92, 93, 109, 110, 111, 113, 114, 115, 117, 118, 123, 124, 126, 127, 129, 130, 132, 133	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, PI7C8152x drives P_AD to a valid logic level when P_GNT_L is asserted.
P_CBE[3:0]	82, 95, 107, 122	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, PI7C8152x drives P_CBE[3:0] to a valid logic level when P_GNT_L is asserted.
P_PAR	106	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, PI7C8152x drives P_PAR to a valid logic level when P_GNT_L is asserted.
P_FRAME_L	96	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.

Name	Pin #	Type	Description
P_IRDY_L	97	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	99	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	100	STS	**Primary Device Select (Active LOW).** Asserted by the target indicating that the device is accepting the transaction. As a master, PI7C8152x 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	101	I	**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	102	STS	**Primary LOCK (Active LOW).** Asserted by an initiator, one clock cycle after the first address phase of a transaction, attempting to perform an operation that may take more than one PCI transaction to complete.
P_IDSEL	83	I	**Primary ID Select.** Used as a chip select line for Type 0 configuration access to PI7C8152x configuration space.
P_PERR_L	104	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	105	OD	**Primary System Error (Active LOW).** Can be driven LOW by any device to indicate a system error condition. PI7C8152x drives this pin on: ▪ Address parity error ▪ Posted write data parity error on target bus ▪ Secondary S_SERR_L asserted ▪ Master abort during posted write transaction ▪ Target abort during posted write transaction ▪ Posted write transaction discarded ▪ Delayed write request discarded ▪ Delayed read request discarded ▪ Delayed transaction master timeout This signal requires an external pull-up resistor for proper operation.
P_REQ_L	69	TS	**Primary Request (Active LOW):** This is asserted by PI7C8152x to indicate that it wants to start a transaction on the primary bus. PI7C8152x de-asserts this pin for at least 2 PCI clock cycles before asserting it again.
P_GNT_L	68	I	**Primary Grant (Active LOW):** When asserted, PI7C8152x can access the primary bus. During idle and P_GNT_L asserted, PI7C8152x will drive P_AD, P_CBE, and P_PAR to valid logic levels.
P_RESET_L	64	I	**Primary RESET (Active LOW):** When P_RESET_L is active, all PCI signals should be asynchronously tri-stated.

2.2.2 SECONDARY BUS INTERFACE SIGNALS

Name	Pin #	Type	Description
S_AD[31:0]	36, 35, 33, 32, 31, 29, 28, 26, 24, 22, 21, 20, 18, 17, 16, 14, 156, 155, 153, 152, 150, 149, 148, 146, 144, 142, 141, 140, 138, 137, 136, 134	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, PI7C8152x drives S_AD to a valid logic level when S_GNT_L is asserted respectively.
S_CBE[3:0]	25, 13, 158, 145	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, PI7C8152x drives S_CBE[3:0] to a valid logic level when the internal grant is asserted.
S_PAR	2	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, PI7C8152x drives S_PAR to a valid logic level when the internal grant is asserted.
S_FRAME_L	11	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	10	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	9	STS	**Secondary TRDY (Active LOW):** Driven by the target of a transaction to indicate its ability to complete current data phase on the secondary side. Once asserted in a data phase, it is not de-asserted until the end of the data phase. Before tri-stated, it is driven to a de-asserted state for one cycle.
S_DEVSEL_L	7	STS	**Secondary Device Select (Active LOW):** Asserted by the target indicating that the device is accepting the transaction. As a master, PI7C8152x 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	6	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	5	STS	**Secondary LOCK (Active LOW):** Asserted by an initiator, one clock cycle after the first address phase of a transaction, when it is propagating a locked transaction downstream. PI7C8152x does not propagate locked transactions upstream.

Name	Pin #	Type	Description
S_PERR_L	4	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	3	I	**Secondary System Error (Active LOW):** Can be driven LOW by any device to indicate a system error condition.
S_REQ_L[3:0]	42, 39, 38, 37	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[3:0]	47, 45, 44, 43	TS	**Secondary Grant (Active LOW):** PI7C8152x asserts these pins to allow external masters to access the secondary bus. PI7C8152x de-asserts these pins for at least 2 PCI clock cycles before asserting it again. During idle and S_GNT_L deasserted, PI7C8152x will drive S_AD, S_CBE, and S_PAR.
S_RESET_L	48	O	**Secondary RESET (Active LOW):** Asserted when any of the following conditions are met: 1. Signal P_RESET_L is asserted. 2. Secondary reset bit in bridge control register in configuration space is set. When asserted, all control signals are tri-stated and zeroes are driven on S_AD, S_CBE, and S_PAR.
S_CFN_L	49	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.

2.2.3 CLOCK SIGNALS

Name	Pin #	Type	Description
P_CLK	66	I	**Primary Clock Input:** Provides timing for all transactions on the primary interface.
S_CLKIN	51	I	**Secondary Clock Input:** Provides timing for all transactions on the secondary interface.
S_CLKOUT[4:0]	61, 59, 57, 55, 53	O	**Secondary Clock Output:** Provides secondary clocks phase synchronous with the P_CLK. In synchronous mode, one of the clock outputs must be fed back to S_CLKIN. Unused outputs may be disabled by: 1. Writing the secondary clock disable bits in the configuration space 2. Terminating them electrically. In asynchronous mode, these pins may not be used. Devices on the secondary interface should use the same clock source that is used for S_CLKIN.

2.2.4 MISCELLANEOUS SIGNALS

Name	Pin #	Type	Description
P_VIO	67	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	52	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	159	I	**Bus/Power Clock Control Management Pin:** When this pin is tied HIGH and the PI7C8152x is placed in the D2 or $D3_{HOT}$ power state, it enables the PI7C8152x 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 PI7C8152x enters the D2 or $D3_{HOT}$ power state.
SCAN_EN	62	I/O	**Full-Scan Enable Control:** When SCAN_EN is LOW, full-scan is in shift operation. When SCAN_EN_H is HIGH, full-scan is in parallel operation. If SCAN_TM_L is HIGH, SCAN_EN is an output with logic 0. For normal operation, SCAN_TM_L should be pulled HIGH and SCAN_EN becomes an output.
SCAN_TM_L	63	I	**Full-Scan Teset Mode Enable:** When SCAN_TM_L is active (LOW), the scan chains will be enabled. For normal operation, pull SCAN_TM_L to HIGH.

2.2.5 POWER AND GROUND

Name	Pin #	Type	Description
VDD	8, 15, 23, 30, 40, 46, 56, 60, 75, 80, 90, 98, 108, 116, 120, 125, 131, 139, 147, 154, 160	P	**Power:** +3.3V Digital power.
VSS	1, 12, 19, 27, 34, 41, 50, 54, 58, 65, 71, 81, 86, 94, 103, 112, 119, 121, 128, 135, 143, 151, 157	P	**Ground:** Digital ground.

2.3 PIN LIST – 160-PIN MQFP

Table 2-1 PIN LIST – 160-PIN MQFP

Pin Number	Name	Type	Pin Number	Name	Type	Pin Number	Name	Type
1	VSS	P	2	S_PAR	TS	3	S_SERR_L	I
4	S_PERR_L	STS	5	S_LOCK_L	STS	6	S_STOP_L	STS
7	S_DEVSEL_L	STS	8	VDD	P	9	S_TRDY_L	STS
10	S_IRDY_L	STS	11	S_FRAME_L	STS	12	VSS	P
13	S_CBE_L[2]	TS	14	S_AD[16]	TS	15	VDD	P
16	S_AD[17]	TS	17	S_AD[18]	TS	18	S_AD[19]	TS
19	VSS	P	20	S_AD[20]	TS	21	S_AD[21]	TS
22	S_AD[22]	TS	23	VDD	P	24	S_AD[23]	TS
25	S_CBE_L[3]	TS	26	S_AD[24]	TS	27	VSS	P
28	S_AD[25]	TS	29	S_AD[26]	TS	30	VDD	P
31	S_AD[27]	TS	32	S_AD[28]	TS	33	S_AD[29]	TS
34	VSS	P	35	S_AD[30]	TS	36	S_AD[31]	TS
37	S_REQ_L[0]	I	38	S_REQ_L[1]	I	39	S_REQ_L[2]	I
40	VDD	P	41	VSS	P	42	S_REQ_L[3]	I
43	S_GNT_L[0]	TS	44	S_GNT_L[1]	TS	45	S_GNT_L[2]	TS
46	VDD	P	47	S_GNT_L[3]	TS	48	S_RESET_L	O

Pin Number	Name	Type	Pin Number	Name	Type	Pin Number	Name	Type
49	S_CFN_L	I	50	VSS	P	51	S_CLKIN	I
52	S_VIO	I	53	S_CLKOUT[0]	O	54	VSS	P
55	S_CLKOUT[1]	O	56	VDD	P	57	S_CLKOUT[2]	O
58	VSS	P	59	S_CLKOUT[3]	O	60	VDD	P
61	S_CLKOUT[4]	O	62	SCAN_EN	I/O	63	SCAN_TM_L	I
64	P_RESET_L	I	65	VSS	P	66	P_CLK	I
67	P_VIO	I	68	P_GNT_L	I	69	P_REQ_L	TS
70	P_AD[31]	TS	71	VSS	P	72	P_AD[30]	TS
73	P_AD[29]	TS	74	P_AD[28]	TS	75	VDD	P
76	P_AD[27]	TS	77	P_AD[26]	TS	78	P_AD[25]	TS
79	P_AD[24]	TS	80	VDD	P	81	VSS	P
82	P_CBE_L[3]	TS	83	P_IDSEL	I	84	P_AD[23]	TS
85	P_AD[22]	TS	86	VSS	P	87	P_AD[21]	TS
88	P_AD[20]	TS	89	P_AD[19]	TS	90	VDD	P
91	P_AD[18]	TS	92	P_AD[17]	TS	93	P_AD[16]	TS
94	VSS	P	95	P_CBE_L[2]	TS	96	P_FRAME_L	STS
97	P_IRDY_L	STS	98	VDD	P	99	P_TRDY_L	STS
100	P_DEVSEL_L	STS	101	P_STOP_L	STS	102	P_LOCK_L	I
103	VSS	P	104	P_PERR_L	STS	105	P_SERR_L	OD
106	P_PAR	TS	107	P_CBE_L[1]	TS	108	VDD	P
109	P_AD[15]	TS	110	P_AD[14]	TS	111	P_AD[13]	TS
112	VSS	P	113	P_AD[12]	TS	114	P_AD[11]	TS
115	P_AD[10]	TS	116	VDD	P	117	P_AD[9]	TS
118	P_AD[8]	TS	119	VSS	P	120	VDD	P
121	VSS	P	122	P_CBE_L[0]	TS	123	P_AD[7]	TS
124	P_AD[6]	TS	125	VDD	P	126	P_AD[5]	TS
127	P_AD[4]	TS	128	VSS	P	129	P_AD[3]	TS
130	P_AD[2]	TS	131	VDD	P	132	P_AD[1]	TS
133	P_AD[0]	TS	134	S_AD[0]	TS	135	VSS	P
136	S_AD[1]	TS	137	S_AD[2]	TS	138	S_AD[3]	TS
139	VDD	P	140	S_AD[4]	TS	141	S_AD[5]	TS
142	S_AD[6]	TS	143	VSS	P	144	S_AD[7]	TS
145	S_CBE_L[0]	TS	146	S_AD[8]	TS	147	VDD	P
148	S_AD[9]	TS	149	S_AD[10]	TS	150	S_AD[11]	TS
151	VSS	P	152	S_AD[12]	TS	153	S_AD[13]	TS
154	VDD	P	155	S_AD[14]	TS	156	S_AD[15]	TS
157	VSS	P	158	S_CBE_L[1]	TS	159	BPCCE	I
160	VDD	P						

3 PCI BUS OPERATION

This Chapter offers information about PCI transactions, transaction forwarding across PI7C8152x, and transaction termination. The PI7C8152x has two 128-byte buffers for read data buffering of upstream and downstream transactions. Also, PI7C8152x has two 128-byte buffers for write data buffering of upstream and downstream transactions.

3.1 TYPES OF TRANSACTIONS

This section provides a summary of PCI transactions performed by PI7C8152x. *Table 3-1* lists the command code and name of each PCI transaction. The Master and Target columns indicate support for each transaction when PI7C8152x initiates transactions as a master, on the primary and secondary buses, and when PI7C8152x responds to transactions as a target, on the primary and secondary 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 PI7C8152x:

- PI7C8152x never initiates a PCI transaction with a reserved command code and, as a target, PI7C8152x ignores reserved command codes.
- PI7C8152x does not generate interrupt acknowledge transactions. PI7C8152x ignores interrupt acknowledge transactions as a target.
- PI7C8152x does not respond to special cycle transactions. PI7C8152x 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.

- PI7C8152x neither generates Type 0 configuration transactions on the primary PCI bus nor responds to Type 0 configuration transactions on the secondary PCI bus.

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]. PI7C8152x supports the linear increment address mode only, which is indicated when the lowest two address bits are equal to zero. If either of the lowest two address bits is nonzero, PI7C8152x automatically disconnects the transaction after the first data transfer.

3.3 DUAL ADDRESS PHASE

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

For a 32-bit interface, the first address phase contains dual address command code on the CBE_L[3:0] lines, and the low 32 address bits on the AD[31:0] lines. The second address

phase consists of the specific memory transaction command code on the CBE_L[3:0] lines, and the high 32 address bits on the AD[31:0] lines. In this way, 64-bit addressing can be supported on 32-bit PCI buses.

The *PCI-to-PCI Bridge Architecture Specification* supports the use of dual address transactions in the prefetchable memory range only. See Section 4.3.2 for a discussion of prefetchable address space. The PI7C8152x supports dual address transactions in both the upstream and the downstream direction. The PI7C8152x supports a programmable 64-bit address range in prefetchable memory for downstream forwarding of dual address transactions. Dual address transactions falling outside the prefetchable address range are forwarded upstream. Prefetching and posting are performed in a manner consistent with the guidelines given in this specification for each type of memory transaction in prefetchable memory space.

3.4 DEVICE SELECT (DEVSEL_L) GENERATION

PI7C8152x always performs positive address decoding (medium decode) when accepting transactions on either the primary or secondary buses. PI7C8152x never does subtractive decode.

3.5 DATA PHASE

The address phase of a PCI transaction is followed by one or more data phases. A data phase is completed when IRDY_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 3.9 for further discussion of transaction termination.

Depending on the command type, PI7C8152x can support multiple data phase PCI transactions. For detailed descriptions of how PI7C8152x imposes disconnect boundaries, see Section 3.6.4 for write address boundaries and Section 3.7.3 read address boundaries.

3.6 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

3.6.1 MEMORY WRITE TRANSACTIONS

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

When PI7C8152x determines that a memory write transaction is to be forwarded across the bridge, PI7C8152x asserts DEVSEL_L with medium decode 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, PI7C8152x accepts write data without obtaining access to the target bus. The PI7C8152x 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 PI7C8152x continues to accept write data until one of the following events occurs:

- The initiator terminates the transaction by de-asserting FRAME_L and IRDY_L.
- 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 PI7C8152x 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, PI7C8152x asserts its request on the target bus. This can occur while PI7C8152x 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, PI7C8152x asserts FRAME_L and drives the stored write address out on the target bus. On the following cycle, PI7C8152x 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, PI7C8152x can drive one DWORD of write data in each PCI clock cycle; that is, no master wait states are inserted. If write data is flowing through PI7C8152x and the initiator stalls, PI7C8152x will signal the last data phase for the current transaction at the target bus if the queue empties. PI7C8152x will restart the follow-on transactions if the queue has new data.

PI7C8152x 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 (PI7C8152x starts another transaction to deliver the rest of the write data).
- The target returns a target abort (PI7C8152x discards remaining write data).
- The master latency timer expires, and PI7C8152x no longer has the target bus grant (PI7C8152x starts another transaction to deliver remaining write data).

Section 3.9.3.2 provides detailed information about how PI7C8152x responds to target termination during posted write transactions.

3.6.2 MEMORY WRITE AND INVALIDATE

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

The PI7C8152x disconnects Memory Write and Invalidate commands at aligned cache line boundaries. The cache line size value in the cache line size register gives the number of DWORD in a cache line.

If the value in the cache line size register does meet the memory write and invalidate conditions, the PI7C8152x returns a target disconnect to the initiator on a cache line boundary.

3.6.3 DELAYED WRITE TRANSACTIONS

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

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

When a write transaction is first detected on the initiator bus, and PI7C8152x forwards it as a delayed transaction, PI7C8152x claims the access by asserting DEVSEL_L and returns a target retry to the initiator. During the address phase, PI7C8152x samples the bus command, address, and address parity one cycle later. After IRDY_L is asserted, PI7C8152x 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 PI7C8152x initiates the transaction on the target bus. PI7C8152x transfers the write data to the target. If PI7C8152x 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 PI7C8152x is unable to deliver write data after 2^{24} (default) or 2^{32} (maximum) attempts, PI7C8152x will report a system error. PI7C8152x 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 PI7C8152x 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, PI7C8152x 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, PI7C8152x returns a target retry to the initiator. PI7C8152x 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, PI7C8152x does not make a new entry into the delayed transaction queue. Section 3.9.3.1 provides detailed

information about how PI7C8152x responds to target termination during delayed write transactions.

PI7C8152x 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, PI7C8152x discards the delayed write completion from the delayed transaction completion queue. PI7C8152x also conditionally asserts P_SERR_L (see Section 6.4).

3.6.4 WRITE TRANSACTION ADDRESS BOUNDARIES

PI7C8152x imposes internal address boundaries when accepting write data. The aligned address boundaries are used to prevent PI7C8152x from continuing a transaction over a device address boundary and to provide an upper limit on maximum latency. PI7C78152 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	Cache line boundary if posted memory write data FIFO does not have enough space for the next cache line
Posted Memory Write and Invalidate	Cache line size = 16	16-DWORD aligned address boundary

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

3.6.5 BUFFERING MULTIPLE WRITE TRANSACTIONS

PI7C8152x 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, PI7C8152x returns a target disconnect to the initiator.

Delayed write transactions are accepted as long as at least one open entry in the delayed transaction queue exists. Therefore, several posted and delayed write transactions can exist in data buffers at the same time. See Chapter 5 for information about how multiple posted and delayed write transactions are ordered.

3.6.6 FAST BACK-TO-BACK WRITE TRANSACTIONS

PI7C8152x is capable of decoding and forwarding fast back-to-back write transactions. When PI7C8152x 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.

3.7 READ TRANSACTIONS

Delayed read forwarding is used for all read transactions crossing PI7C8152x. 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.

3.7.1 PREFETCHABLE READ TRANSACTIONS

A prefetchable read transaction is a read transaction where PI7C8152x 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, PI7C8152x forces all byte enable bits to be on for all data phases.

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

The amount of data that is prefetched depends on the type of transaction. The amount of prefetching may also be affected by the amount of free buffer space available in PI7C8152x, and by any read address boundaries encountered.

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

3.7.2 NON-PREFETCHABLE READ TRANSACTIONS

A non-prefetchable read transaction is a read transaction where PI7C8152x 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, PI7C8152x 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.

3.7.3 READ PREFETCH ADDRESS BOUNDARIES

PI7C8152x imposes internal read address boundaries on read prefetched data. When a read transaction reaches one of these aligned address boundaries, the PI7C8152x stops pre-fetched data, unless the target signals a target disconnect before the read prefetched boundary is reached. When PI7C8152x 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 prefetched data is discarded.

Prefetchable read transactions in flow-through mode prefetch to the nearest aligned 4KB address boundary, or until the initiator de-asserts FRAME_L. Section 3.7.6 describes flow-through mode during read operations.

*Table 3-4*shows the read prefetch 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 Address Boundary
Configuration Read	-	*	One DWORD (no prefetch)
I/O Read	-	*	One DWORD (no prefetch)
Memory Read	Non-Prefetchable	*	One DWORD (no prefetch)
Memory Read	Prefetchable	CLS = 0 or 16	16-DWORD aligned address boundary
Memory Read	Prefetchable	CLS = 1, 2, 4, 8	Cache line address boundary
Memory Read Line	-	CLS = 0 or 16	16-DWORD aligned address boundary
Memory Read Line	-	CLS = 1, 2, 4, 8	Cache line boundary
Memory Read Multiple	-	CLS = 0 or 16	Queue full
Memory Read Multiple	-	CLS = 1, 2, 4, 8	Second cache line boundary

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

Table 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
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.

3.7.4 DELAYED READ REQUESTS

PI7C8152x 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.

PI7C8152x accepts a delayed read request by sampling the read address, read bus command, and address parity. When IRDY_L is asserted, PI7C8152x then samples the byte enable bits for the first data phase. This information is entered into the delayed transaction queue. PI7C8152x 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.

3.7.5 DELAYED READ COMPLETION ON TARGET BUS

When delayed read request reaches the head of the delayed transaction queue, PI7C8152x arbitrates for the target bus and initiates the read transaction only if all previously queued posted write transactions have been delivered. PI7C8152x 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, PI7C8152x 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 PI7C8152x 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, PI7C8152x does not initiate any further attempts to read more data.

If PI7C8152x is unable to obtain read data from the target after 2^{24} (default) or 2^{32} (maximum) attempts, PI7C8152x will report system error. The number of attempts is programmable. PI7C8152x 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 PI7C8152x 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 PI7C8152x can accept one DWORD of read data each PCI clock cycle; that is, no master wait states are inserted. The number of DWORD's transferred during a delayed read transaction matches the prefetch address boundary given in *Table 3-4* (assuming no disconnect is received from the target).

3.7.6 DELAYED READ COMPLETION ON INITIATOR BUS

When the transaction has been completed on the target bus, and the delayed read data is at the head of the read data queue, and all ordering constraints with posted write transactions have been satisfied, the PI7C8152x transfers the data to the initiator when the initiator repeats the transaction. For memory read transactions, PI7C8152x aliases memory read line and memory read multiple bus commands to memory read when matching the bus command of the transaction to the bus command in the delayed transaction queue if bit[3] of offset 74h is set to '1'. PI7C8152x returns a target disconnect along with the transfer of the last DWORD of read data to the initiator. If PI7C8152x 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, PI7C8152x 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, PI7C8152x 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.

PI7C8152x implements a master timeout timer that starts counting when the delayed read completion is at the head of the delayed transaction queue, and the read data is at the head of the read data queue. The initial value of this timer is programmable through configuration transaction. If the initiator does not repeat the read transaction and before the master timeout timer expires (2^{15} default), PI7C8152x discards the read transaction and read data from its queues. PI7C8152x also conditionally asserts P_SERR_L (see Section 6.4).

PI7C8152x 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 PI7C8152x.

3.7.7 FAST BACK-TO-BACK READ TRANSACTION

PI7C8152x is capable to decode fast back-to-back read transactions on both primary and secondary. PI7C8152x cannot generate fast back-to-back read transactions on both the secondary and primary even if bit[23] of offset 3Ch is set to ‘1’ or bit[9] of offset 04h is set to ‘1’.

3.8 CONFIGURATION TRANSACTIONS

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

In addition to accepting configuration transactions for initialization of its own configuration space, the PI7C8152x 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.

3.8.1 TYPE 0 ACCESS TO PI7C8152x

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 PI7C8152x 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.

PI7C8152x 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 PI7C8152x ignores all Type 0 transactions initiated on the secondary interface.

3.8.2 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.

PI7C8152x 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. PI7C8152x 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, PI7C8152x generates a Type 0 transaction only on the secondary bus, and never on the primary bus.

PI7C8152x responds to a Type 1 configuration transaction and translates it into a Type 0 transaction on the secondary 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 write transaction.

When PI7C8152x translates the Type 1 transaction to a Type 0 transaction on the secondary interface, it performs the following translations to the address:

- Sets the lowest two address bits on S_AD[1:0] to 0.
- Decodes the device number and drives the bit pattern specified in *Table 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.

PI7C8152x asserts a unique address line based on the device number. These address lines may be used as secondary bus IDSEL signals. The mapping of the address lines depends on the device number in the Type 1 address bits P_AD[15:11]. *Table 3-6* presents the mapping that PI7C8152x 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) 0000 0000 0000 0000 (P_AD[7:2] = 00h)	-

PI7C8152x can assert up to 16 unique address lines to be used as IDSEL signals for up to 16 devices on the secondary bus, for device numbers ranging from 0 through 15. Because of electrical loading constraints of the PCI bus, more than 16 IDSEL signals should not be necessary. However, if device numbers greater than 15 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.

PI7C8152x 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.

3.8.3 TYPE 1 TO TYPE 1 FORWARDING

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

When PI7C8152x detects a Type 1 configuration transaction intended for a PCI bus downstream from the secondary bus, PI7C8152x 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.

PI7C8152x 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 PI7C8152x 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.

3.8.4 SPECIAL CYCLES

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

PI7C8152x 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 PI7C8152x 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, PI7C8152x responds with TRDY_L to the next attempt of the con-figuration transaction from the initiator. If more than one data transfer is requested, PI7C8152x responds with a target disconnect operation during the first data phase.

3.9 TRANSACTION TERMINATION

This section describes how PI7C8152x 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_L 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.

3.9.1 MASTER TERMINATION INITIATED BY PI7C8152x

PI7C8152x, as an initiator, uses normal termination if DEVSEL_L is returned by target within five clock cycles of PI7C8152x's assertion of FRAME_L on the target bus. As an initiator, PI7C8152x 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 PI7C8152x's bus grant is de-asserted.

- The target terminates the transaction with a retry, disconnect, or target abort.

If PI7C8152x 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 PI7C8152x 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.

3.9.2 MASTER ABORT RECEIVED BY PI7C8152x

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, PI7C8152x 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, PI7C8152x is able to reflect the master abort condition back to the initiator. When PI7C8152x detects a master abort in response to a delayed transaction, and when the initiator repeats the transaction, PI7C8152x 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, PI7C8152x discards the posted write data and makes no more attempt to deliver the data. PI7C8152x 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, PI7C8152x 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 PI7C8152x 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.

3.9.3 TARGET TERMINATION RECEIVED BY PI7C8152x

When PI7C8152x 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

PI7C8152x handles these terminations in different ways, depending on the type of transaction being performed.

3.9.3.1 DELAYED WRITE TARGET TERMINATION RESPONSE

When PI7C8152x 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.

PI7C8152x repeats a delayed write transaction until one of the following conditions is met:

- PI7C8152x completes at least one data transfer.
- PI7C8152x receives a master abort.
- PI7C8152x receives a target abort.

PI7C8152x 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 PI7C8152x makes 2^{24} (default) attempts of the same delayed write trans-action on the target bus, PI7C8152x 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). PI7C8152x will report system error. See Section 6.4 for a description of system error conditions.

3.9.3.2 POSTED WRITE TARGET TERMINATION RESPONSE

When PI7C8152x 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, PI7C8152x 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, PI7C8152x 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 PI7C8152x makes 2^{24} (default) write transaction attempts and fails to deliver all posted write data associated with that transaction, PI7C8152x 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). PI7C8152x will report system error. See Section 6.4 for a discussion of system error conditions.

3.9.3.3 DELAYED READ TARGET TERMINATION RESPONSE

When PI7C8152x 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.

PI7C8152x repeats a delayed read transaction until one of the following conditions is met:

- PI7C8152x completes at least one data transfer.
- PI7C8152x receives a master abort.
- PI7C8152x receives a target abort.

PI7C8152x makes 2^{24} (default) read attempts resulting in a response of target retry.

Table 3-9 RESPONSE TO DELAYED READ TARGET TERMINATIOIN

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

After PI7C8152x makes 2^{24}(default) attempts of the same delayed read transaction on the target bus, PI7C8152x 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). PI7C8152x will report system error. See Section 6.4 for a description of system error conditions.

3.9.4 TARGET TERMINATION INITIATED BY PI7C8152x

PI7C8152x 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.

3.9.4.1 TARGET RETRY

PI7C8152x returns a target retry to the initiator when it cannot accept write data or return read data as a result of internal conditions. PI7C8152x 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 PI7C8152x, 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 the head of the read data queue if offset 40h bit[11:0]=11 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 PI7C8152x, and the read transaction is not a locked transaction.
- PI7C78152 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 PI7C8152x, 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.

3.9.4.2 TARGET DISCONNECT

PI7C8152x returns a target disconnect to an initiator when one of the following conditions is met:

- PI7C8152x hits an internal address boundary.
- PI7C8152x cannot accept any more write data.
- PI7C8152x has no more read data to deliver.

See Section 3.6.4 for a description of write address boundaries, and Section 3.7.3 for a description of read address boundaries.

3.9.4.3 TARGET ABORT

PI7C8152x returns a target abort to an initiator when one of the following conditions is met:

- PI7C8152x is returning a target abort from the intended target.
- When PI7C8152x 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

PI7C8152x 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

PI7C8152x 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 PI7C8152x. The addresses that are not marked for downstream are always forwarded upstream.

4.2 I/O ADDRESS DECODING

PI7C8152x 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, PI7C8152x 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, PI7C8152x 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

PI7C8152x 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. PI7C8152x supports 32-bit I/O addressing, which allows I/O addresses downstream of PI7C8152x 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 PI7C8152x 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

PI7C8152x supports ISA mode by providing an ISA enable bit in the bridge control register in configuration space. ISA mode modifies the response of PI7C8152x 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 PI7C8152x 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, PI7C8152x 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, PI7C8152x 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 PI7C8152x 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

PI7C8152x 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. PI7C8152x 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 PI7C8152x uses to determine when to forward memory commands. PI7C8152x forwards a memory transaction from the primary to the secondary interface if the transaction address falls within the memory-mapped I/O address range. PI7C8152x 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. PI7C8152x 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 PI7C8152x uses to determine when to forward memory commands. PI7C8152x forwards a memory transaction from the primary to the secondary interface if the transaction address falls within the prefetchable memory address range. PI7C8152x ignores memory transactions initiated on the secondary interface that fall into this address range. PI7C8152x 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

PI7C8152x 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 PI7C8152x, set the VGA mode bit in the bridge control register in configuration space to enable VGA mode. When PI7C8152x 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. PI7C8152x 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. PI7C8152x requests only a single data transfer from the target, and read byte enable bits are forwarded to the target bus.

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

4.4.2 VGA SNOOP MODE

PI7C8152x provides VGA snoop mode, allowing for VGA palette write transactions to be forwarded downstream. This mode is used when a graphics device downstream from PI7C8152x 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 PI7C8152x claims VGA palette write transactions by asserting DEVSEL_L in VGA snoop mode.

When VGA snoop bit is set, PI7C8152x 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, PI7C8152x behaves in the same way as if only the VGA mode bit were set.

5 TRANSACTION ORDERING

To maintain data coherency and consistency, PI7C8152x complies with the ordering rules set forth in the *PCI Local Bus Specification, Revision 2.2*, for transactions crossing the bridge. This chapter describes the ordering rules that control transaction forwarding across PI7C8152x.

5.1 TRANSACTIONS GOVERNED BY ORDERING RULES

Ordering relationships are established for the following classes of transactions crossing PI7C8152x:

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.

PI7C8152x does not combine or merge write transactions:

- PI7C8152x does not combine separate write transactions into a single write transaction—this optimization is best implemented in the originating master.
- PI7C8152x does not merge bytes on separate masked write transactions to the same DWORD address—this optimization is also best implemented in the originating master.
- PI7C8152x 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 PI7C8152x.

The following general ordering guidelines govern transactions crossing PI7C8152x:

- 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. PI7C8152x 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 PI7C8152x and must also be true for other bus agents. Otherwise, a deadlock can occur.

- PI7C8152x accepts posted write transactions, regardless of the state of completion of any delayed transactions being forwarded across PI7C8152x.

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

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

The entries without superscripts reflect the PI7C8152x'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 PI7C8152x in the same direction. Note that delayed completion transactions cross PI7C8152x 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 PI7C8152x 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 2.2, provides the following alternative methods for synchronizing data and interrupts:

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

PI7C8152x 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

PI7C8152x checks, forwards, and generates parity on both the primary and secondary interfaces. To maintain transparency, PI7C8152x always tries to forward the existing parity

condition on one bus to the other bus, along with address and data. PI7C8152x 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, PI7C8152x 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 PI7C8152x handles errors. It also describes error status reporting and error operation disabling.

6.1 ADDRESS PARITY ERRORS

PI7C8152x checks address parity for all transactions on both buses, for all address and all bus commands. When PI7C8152x 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, PI7C8152x 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, PI7C8152x proceeds normally and accepts the transaction if it is directed to or across PI7C8152x.
- PI7C8152x sets the detected parity error bit in the status register.
- PI7C8152x 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 PI7C8152x 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, PI7C8152x 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, PI7C8152x proceeds normally and accepts transaction if it is directed to or across PI7C8152x.
- PI7C8152x sets the detected parity error bit in the secondary status register.
- PI7C8152x 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, PI7C8152x 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 PI7C8152x.

6.2.1 CONFIGURATION WRITE TRANSACTIONS TO CONFIGURATION SPACE

When PI7C8152x detects a data parity error during a Type 0 configuration write transaction to PI7C8152x configuration space, the following events occur:

If the parity error response bit is set in the command register, PI7C8152x asserts P_TRDY_L and writes the data to the configuration register. PI7C8152x also asserts P_PERR_L. If the parity error response bit is not set, PI7C8152x does not assert P_PERR_L.

PI7C8152x 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 PI7C8152x 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 PI7C8152x detects a read data parity error on the secondary bus, the following events occur:

- PI7C8152x 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.
- PI7C8152x sets the detected parity error bit in the secondary status register.
- PI7C8152x 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.
- PI7C8152x 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.
- PI7C8152x completes the transaction normally.

For upstream transactions, when PI7C8152x detects a read data parity error on the primary bus, the following events occur:

- PI7C8152x asserts P_PERR_L two cycles following the data transfer, if the primary interface parity error response bit is set in the command register.
- PI7C8152x sets the detected parity error bit in the primary status register.
- PI7C8152x 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.
- PI7C8152x 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.
- PI7C8152x completes the transaction normally.

PI7C8152x 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 PI7C8152x detects PERR_L asserted while returning read data to the initiator, PI7C8152x does not take any further action and completes the transaction normally.

6.2.3 DELAYED WRITE TRANSACTIONS

When PI7C8152x 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 PI7C8152x 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 PI7C8152x 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, PI7C8152x 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, PI7C8152x also asserts PERR_L.
- If the parity-error-response bit is not set, PI7C8152x returns a target retry. It queues the transaction as usual. PI7C8152x does not assert PERR_L. In this case, the initiator repeats the transaction.
- PI7C8152x 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 PI7C8152x is delivering data to the target on the secondary bus and S_PERR_L is asserted by the target, the following events occur:

- PI7C8152x 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.
- PI7C8152x captures the parity error condition to forward it back to the initiator on the primary bus.

Similarly, for upstream transactions, when PI7C8152x is delivering data to the target on the primary bus and P_PERR_L is asserted by the target, the following events occur:

- PI7C8152x 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.
- PI7C8152x 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 PI7C8152x has write status to return, the following events occur:

- PI7C8152x 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.
- PI7C8152x 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 PI7C8152x has write status to return, the following events occur:

- PI7C8152x 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).

- PI7C8152x 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:

- PI7C8152x 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.
- PI7C8152x 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:

- PI7C8152x 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.
- PI7C8152x completes the transaction normally.

6.2.4 POSTED WRITE TRANSACTIONS

During downstream posted write transactions, when PI7C8152x responds as a target, it detects a data parity error on the initiator (primary) bus and the following events occur:

- PI7C8152x 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.
- PI7C8152x sets the parity error detected bit in the status register of the primary interface.
- PI7C8152x captures and forwards the bad parity condition to the secondary bus.
- PI7C8152x completes the transaction normally.

Similarly, during upstream posted write transactions, when PI7C8152x responds as a target, it detects a data parity error on the initiator (secondary) bus, the following events occur:

- PI7C8152x 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.
- PI7C8152x sets the parity error detected bit in the status register of the secondary interface.
- PI7C8152x captures and forwards the bad parity condition to the primary bus.
- PI7C8152x 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:

- PI7C8152x 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.
- PI7C8152x 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.
    - PI7C8152x 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:

- PI7C8152x 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.
- PI7C8152x 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.
- PI7C8152x 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 PI7C8152x to data parity errors are presented according to the type of transaction in progress. This section organizes the responses of PI7C8152x to data parity errors according to the status bits that PI7C8152x 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 PI7C8152x 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 PI7C8152x 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

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

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:

- PI7C8152x 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 BUS 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 PI7C8152x 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 BUS 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:

- PI7C8152x 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.
- PI7C8152x 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#	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:

- PI7C8152x 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.
- PI7C8152x 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#	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:

- PI7C8152x has detected P_PERR_L asserted on an upstream posted write transaction or S_PERR_L asserted on a downstream posted write transaction.
- PI7C8152x 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#	Transaction Type	Direction	Bus Where Error Was Detected	Primary / Secondary Parity Error Response Bits
1 (de-asserted)	Read	Downstream	Primary	x / x
1	Read	Downstream	Secondary	x / x
1	Read	Upstream	Primary	x / x
1	Read	Upstream	Secondary	x / x
1	Posted Write	Downstream	Primary	x / x
0[2] (asserted)	Posted Write	Downstream	Secondary	1 / 1
0[3]	Posted Write	Upstream	Primary	1 / 1
1	Posted Write	Upstream	Secondary	x / x
1	Delayed Write	Downstream	Primary	x / x
1	Delayed Write	Downstream	Secondary	x / x
1	Delayed Write	Upstream	Primary	x / x
1	Delayed Write	Upstream	Secondary	x / x

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

PI7C8152x 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 PI7C8152x to assert P_SERR_L for any reason, the SERR_L enable bit must be set in the command register.
- Whenever PI7C8152x asserts P_SERR_L, PI7C8152x must also set the signaled system error bit in the status register.

In compliance with the PCI-to-PCI Bridge Architecture Specification, PI7C8152x 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, PI7C8152x also sets the received system error bit in the secondary status register.

PI7C8152x 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 PI7C8152x.

7.1 CONCURRENT LOCKS

The primary and secondary bus lock mechanisms operate concurrently except when a locked transaction crosses PI7C8152x. 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 PI7C8152x

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 PI7C8152x 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 PI7C8152x detects on the primary bus, an initial locked transaction intended for a target on the secondary bus, PI7C8152x samples the address, transaction type, byte enable bits, and parity, as described in Section 3.7.4. It also samples the lock signal. If there is a lock established between 2 ports or the target bus is already locked by another master, then the current lock cycle is retried without forward. Because a target retry is signaled to the initiator, the initiator must relinquish the lock on the primary bus, and therefore the lock is not yet established.

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

When the locked delayed memory read request is queued, PI7C8152x does not queue any more transactions until the locked sequence is finished. PI7C8152x signals a target retry to all transactions initiated subsequent to the locked read transaction that are intended for targets on the other side of PI7C8152x. PI7C8152x 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, PI7C8152x 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), PI7C8152x 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 PI7C8152x. Otherwise, the pending queued locked transaction would not have been queued. When PI7C8152x 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, PI7C8152x transfers the read data back to the initiator, and the lock is then also established on the primary bus.

For PI7C8152x 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 PI7C8152x 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 PI7C8152x 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. PI7C8152x resumes forwarding unlocked transactions in both directions.

7.2.2 LOCKED TRANSACTION IN UPSTREAM DIRECTION

PI7C8152x ignores upstream lock and transactions. PI7C8152x 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, PI7C8152x 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. PI7C8152x 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, PI7C8152x has already completed the transaction on the target bus. In this example, as soon as PI7C8152x detects that the initiator has relinquished the LOCK_L signal by sampling it in the de-asserted state while FRAME_L is de-asserted, PI7C8152x 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 PI7C8152x 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, PI7C8152x 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 PI7C8152x receives a target abort or a master abort in response to a locked delayed transaction, PI7C8152x 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. PI7C8152x sets the appropriate status bits, flagging the abnormal target termination condition (see Section 3.9). Normal forwarding of unlocked posted and delayed transactions is resumed.

When PI7C8152x receives a target abort or a master abort in response to a locked posted write transaction, PI7C8152x cannot pass back that status to the initiator. PI7C8152x 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

PI7C8152x 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 PI7C8152x, typically on the motherboard. For the secondary PCI bus, PI7C8152x 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

PI7C8152x implements a request output pin, P_REQ_L, and a grant input pin, P_GNT_L, for primary PCI bus arbitration. PI7C8152x 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, PI7C8152x 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 PI7C8152x on the primary PCI bus, PI7C8152x 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 PI7C8152x has asserted P_REQ_L, PI7C8152x initiates a transaction on the primary bus during the next PCI clock cycle. When P_GNT_L is asserted to PI7C8152x when P_REQ_L is not asserted, PI7C8152x parks P_AD, P_CBE, and P_PAR by driving them to valid logic levels. When the primary bus is parked at PI7C8152x and PI7C8152x has a transaction to initiate on the primary bus, PI7C8152x starts the transaction if P_GNT_L was asserted during the previous cycle.

8.2 SECONDARY PCI BUS ARBITRATION

PI7C8152x implements an internal secondary PCI bus arbiter. This arbiter supports four external masters on the secondary bus in addition to PI7C8152x. The internal arbiter can be disabled, and an external arbiter can be used instead for secondary bus arbitration.

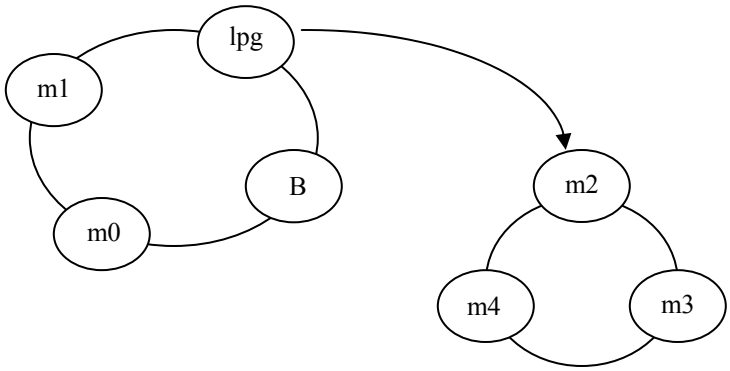
8.2.1 SECONDARY BUS ARBITRATION USING THE INTERNAL ARBITER

To use the internal arbiter, the secondary bus arbiter enable pin, S_CFN_L, must be tied LOW. PI7C8152x has four secondary bus request input pins, S_REQ_L[3:0], and has four secondary bus output grant pins, S_GNT_L[3:0], 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 LOW.

The secondary arbiter supports a 2-sets programmable 2-level rotating algorithm with each set taking care of 4 requests / grants. Each set of masters can be assigned to a high priority

group and a low priority group. The low priority group as a whole represents one entry in the high priority group; that is, if the high priority group consists of n masters, then in at least every n+1 transactions the highest priority is assigned to the low priority group. Priority rotates evenly among the low priority group. Therefore, members of the high priority group can be serviced n transactions out of n+1, while one member of the low priority group is serviced once every n+1 transactions. *Figure 8-1* shows an example of an internal arbiter where three masters, including PI7C8152x, are in the high priority group, and two 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***, B, m0, m1,* **m3***, B, m0, m1,* **m4**.

Figure 8-1 SECONDARY ARBITER EXAMPLE



Each bus master, including PI7C8152x, 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 PI7C8152x is assigned to the high priority group. PI7C8152x 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 PI7C8152x 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_L) 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, PI7C8152x 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, PI7C8152x uses the S_GNT_L[0] pin to request the secondary bus. When the reconfigured S_REQ_L[0] pin is asserted LOW after PI7C8152x has asserted S_GNT_L[0], PI7C8152x initiates a transaction on the secondary bus one cycle later. If grant is asserted and PI7C8152x has not asserted the request, PI7C8152x parks AD, CBE and PAR pins by driving them to valid logic levels.

The unused secondary bus grant outputs, S_GNT_L[3:1] are driven HIGH. The unused secondary bus request inputs, S_REQ_L[3:1], should be pulled HIGH.

8.2.4 BUS PARKING

Bus parking refers to driving the AD[31:0], CBE_L[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.

PI7C8152x 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, PI7C8152x 3-states the P_AD, P_CBE, and P_PAR signals on the next PCI clock cycle. If PI7C8152x is parking the primary PCI bus and wants to initiate a transaction on that bus, then PI7C8152x 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, PI7C8152x keeps the secondary bus grant asserted to a particular master until a new secondary bus request comes along. After reset, PI7C8152x 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 PI7C8152x. By default, offset 48h, bit 1, is set to 0. If the internal arbiter is disabled, PI7C8152x 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 INPUT

PI7C8152x implements a primary clock input for the PCI interface. In synchronous mode, 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 from the primary clock, and runs at the same frequency in synchronous mode. PI7C8152x operates at a maximum frequency of 66 MHz.

9.2 SECONDARY CLOCK OUTPUTS

PI7C8152x has 5 secondary clock outputs, S_CLKOUT[4:0] that can be used as clock inputs for up to four external secondary bus devices when PI7C8152x is in synchronous mode. The S_CLKOUT[4:0] outputs are derived from P_CLK. The secondary clock edges are delayed from P_CLK edges by a minimum of 0ns. For the PI7C8152B in asynchronous mode, the S_CLKOUT[4:0] outputs cannot be used for external secondary bus devices. These are the rules for using secondary clocks:

- Each secondary clock output is limited to no more than one load.
- One of the secondary clock outputs must be used for the S_CLKIN input (in synchronous mode).
- Each secondary clock output cannot be used for external secondary bus devices when PI7C8152B is in asynchronous mode.

9.3 ASYNCHRONOUS MODE (PI7C8152B ONLY)

In asynchronous mode, the PI7C8152B can be run in the following frequency configuration:

Primary	Secondary
25MHz to 66MHz	25MHz to 66MHz

P_CLK is the input source for the primary clock and S_CLKIN is the input source for the secondary clock. The S_CLKOUT[4:0] outputs cannot be used for any external secondary bus devices in asynchronous mode. Instead, devices on the secondary bus must utilize the same clock that is used for S_CLKIN.

9.4 SYNCHRONOUS MODE

In synchronous mode, the primary bus and the secondary bus must both be running at the same frequency. The S_CLKOUT[4:0] outputs are derived directly from P_CLK and S_CLKOUT[4] is used as a feedback to S_CLKIN. PI7C8152x will not operate at split frequencies (primary different than secondary) in synchronous mode. The frequency on the secondary bus will be the same as the frequency on the primary bus unless asynchronous mode is utilized.

10 PCI POWER MANAGEMENT

PI7C8152x incorporates functionality that meets the requirements of the *PCI Power Management Specificatio*n, *Revision 1.1*. These features include:

- PCI Power Management registers using the Enhanced Capabilities Port (ECP) address mechanism
- Support for D0, D1, D2, 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 D2 or D3 hot power management state
- Support of the B1 secondary bus power state when in the D1 power management state

Table 10-1 shows the states and related actions that PI7C8152x performs during power management transitions. (No other transactions are permitted.)

Table 10-1 POWER MANAGEMENT TRANSITIONS

Current Status	Next State	Action
D0	D3cold	Power has been removed from PI7C8152x. A power-up reset must be performed to bring PI7C8152x to D0.
D0	D3hot	If enabled to do so by the BPCCE pin, PI7C8152x will disable the secondary clocks and drive them LOW.
D0	D2	If enabled to do so by the BPCEE pin, PI7C8152x will disable the secondary clocks and driver them LOW.
D0	D1	PI7C8152x only accepts Type 0 configuration cycles on the primary and ignores all others.
D3hot	D0	PI7C8152x 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 PI7C8152x. A power-up reset must be performed to bring PI7C8152x to D0.
D3cold	D0	Power-up reset. PI7C8152x performs the standard power-up reset functions as described in Section 11.

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

11 RESET

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

11.1 PRIMARY INTERFACE RESET

PI7C8152x has a reset input, P_RESET_L. When P_RESET_L is asserted, the following events occur:

- PI7C8152x immediately tri-states all primary PCI interface signals. On the secondary, S_AD and S_CBE are driven LOW, while other control signals are tri-stated.
- PI7C8152x 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. PI7C8152x is not accessible during P_RESET_L. After P_RESET_L is de-asserted, PI7C8152x remains inaccessible for 16 PCI clocks before the first configuration transaction can be accepted.

11.2 SECONDARY INTERFACE RESET

PI7C8152x is responsible for driving the secondary bus reset signals, S_RESET_L. PI7C8152x 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.

The chip reset bit in the diagnostic / control register is set. When S_RESET_L is asserted, PI7C8152x immediately tri-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. S_RESET_L remains asserted until a configuration write operation clears the secondary reset bit.

When S_RESET_L is asserted, all secondary PCI interface control signals, including the secondary grant outputs, are immediately tri-stated. Signals S_AD, S_CBE_L[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, PI7C8152x remains accessible during secondary interface reset and continues to respond to accesses to its configuration space from the primary interface.

11.3 CHIP RESET

The chip reset bit in the diagnostic control register can be used to reset the PI7C8152x 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. Within 20 PCI clock cycles after completion of the configuration write operation, PI7C8152x's reset bit automatically clears and PI7C8152x is ready for configuration.

During reset, PI7C8152x is inaccessible.

12 CONFIGURATION REGISTERS

PCI configuration defines a 64-byte DWORD to define various attributes of PI7C8152x as shown below.

12.1 CONFIGURATION REGISTER

31-24	23-16	15-8	7-0	DWORD Address
Device ID		Vendor ID		00h
Primary Status		Command		04h
Class Code			Revision ID	08h
Reserved	Header Type	Primary Latency Timer	Cache Line Size	0Ch
Reserved				10h – 14h
Secondary Latency Timer	Subordinate Bus Number	Secondary Bus Number	Primary Bus Number	18h
Secondary Status		I/O Limit Address	I/O Base Address	1Ch
Memory Limit Address		Memory Base Address		20h
Prefetchable Memory Limit Address		Prefetchable Memory Base Address		24h
Prefetchable Memory Base Address Upper 32-bit				28h
Prefetchable Memory Limit Address Upper 32-bit				2Ch
I/O Limit Address Upper 16-bit		I/O Base Address Upper 16-bit		30h
Reserved			Capability Pointer to DCh	34h
Reserved				38h
Bridge Control		Interrupt Pin	Reserved	3Ch
Arbiter Control		Diagnostic / Chip Control		40h
Reserved				44h
Reserved		Extended Chip Control		48h
Secondary Bus Arbiter Preemption Control	Reserved			4Ch
Reserved				50h – 60h
Reserved	Reserved	Reserved	P_SERR# Event Disable	64h
Reserved	P_SERR_L Status	Secondary Clock Control		68h
Reserved				6Ch - 70h
Reserved		Port Option		74h
Reserved				78h – 7Ch
Secondary Master Timeout Counter		Primary Master Timeout Counter		80h
Reserved				84h – D8h
Power Management Capabilities		Next Item Pointer	Capability ID	DCh
Reserved	PPB Support Extensions	Power Management Data		E0h
Reserved				E4h-FFh

12.1.1 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.

12.1.2 DEVICE ID REGISTER – OFFSET 00h

Bit	Function	Type	Description
31:16	Device ID	R/O	Identifies this device as the PI7C8152. Hardwired as 8152h.

12.1.3 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 0: ignore I/O transactions on the primary interface 1: enable response to I/O transactions on the primary interface Reset to 0
1	Memory Space Enable	R/W	Controls response to memory accesses on the primary interface 0: ignore memory transactions on the primary interface 1: enable response to memory transactions on the primary interface Reset to 0
2	Bus Master Enable	R/W	Controls ability to operate as a bus master on the primary interface 0: do not initiate memory or I/O transactions on the primary interface and disable response to memory and I/O transactions on the secondary interface 1: enables PI7C8152x to operate as a master on the primary interfaces for memory and I/O transactions forwarded from the secondary interface Reset to 0
3	Special Cycle Enable	R/O	No special cycles defined. Bit is defined as read only and returns 0 when read
4	Memory Write And Invalidate Enable	R/O	PI7C8152x does not generate memory write and invalidate transactions except for forwarding a transaction for another master. 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 0: ignore VGA palette accesses on the primary 1: enable positive decoding response to VGA palette writes on the primary interface with I/O address bits AD[9:0] equal to 3C6h, 3C8h, and 3C9h (inclusive of ISA alias; AD[15:10] are not decoded and may be any value)
6	Parity Error Response	R/W	Controls response to parity errors 0: PI7C8152x may ignore any parity errors that it detects and continue normal operation 1: PI7C8152x must take its normal action when a parity error is detected Reset to 0
7	Wait Cycle Control	R/O	Controls the ability to perform address / data stepping Read as 0 to indicate PI7C8152x does not perform address / data stepping. Reset to 0

Bit	Function	Type	Description
8	P_SERR_L enable	R/W	Controls the enable for the P_SERR_L pin 0: disable the P_SERR_L driver 1: enable the P_SERR_L driver Reset to 0
9	Fast Back-to-Back Enable	R/W	Controls PI7C8152x's ability to generate fast back-to-back transactions to different devices on the primary interface. 0: no fast back-to-back transactions 1: enable fast back-to-back transactions Reset to 0
15:10	Reserved	R/O	Returns 000000 when read

12.1.4 PRIMARY 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) Reset to 1
21	66MHz Capable	R/O	Set to 1 to indicate the primary may be run at 66MHz operation 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 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 Reset to 0
26:25	DEVSEL_L timing	R/O	DEVSEL_L timing (medium decoding) 00: fast DEVSEL_L decoding 01: medium DEVSEL_L decoding 10: slow DEVSEL_L decoding 11: reserved Reset to 01
27	Signaled Target Abort	R/WC	Set to 1 (by a target device) whenever a target abort cycle occurs Reset to 0
28	Received Target Abort	R/WC	Set to 1 (by a master device) whenever transactions are terminated with target aborts Reset to 0
29	Received Master Abort	R/WC	Set to 1 (by a master) when transactions are terminated with Master Abort Reset to 0
30	Signaled System Error	R/WC	Set to 1 when P_SERR_L is asserted Reset to 0

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

12.1.5 REVISION ID REGISTER – OFFSET 08h

Bit	Function	Type	Description
7:0	Revision	R/O	Indicates revision number of device. Hardwired to 01h

12.1.6 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

12.1.7 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. Only cache line sizes (in units of 4-byte) which are a power of two are valid (only one bit can be set in this register; only 00h, 01h, 02h, 04h, 08h, and 10h are valid values). Reset to 0

12.1.8 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. Reset to 0

12.1.9 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.

12.1.10 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. Reset to 0

12.1.11 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. Reset to 0

12.1.12 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. Reset to 0

12.1.13 SECONDARY LATENCY TIMER REGISTER – OFFSET 18h

Bit	Function	Type	Description
31:24	Secondary Latency Timer	R/W	Latency timer for secondary. Indicates the number of PCI clocks from the assertion of S_FRAME_L to the expiration of the timer when the PI7C8152x is acting as a master on the secondary. 0: PI7C8152x ends the transaction after the first data transfer when the PI7C8152x's secondary bus grant has been deasserted, with the exception of memory write and invalidate transactions. Reset to 0

12.1.14 I/O BASE ADDRESS REGISTER – OFFSET 1Ch

Bit	Function	Type	Description
3:0	32-bit Indicator	R/O	Read as 01h to indicate 32-bit I/O addressing
7:4	I/O Base Address [15:12]	R/W	Defines the bottom address of the I/O address range for the bridge to determine when to forward I/O transactions from one interface to the other. The upper 4 bits correspond to address bits [15:12] and are writable. The lower 12 bits corresponding to address bits [11:0] are assumed to be 0. The upper 16 bits corresponding to address bits [31:16] are defined in the I/O base address upper 16 bits address register Reset to 0

12.1.15 I/O LIMIT ADDRESS 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 Limit Address [15:12]	R/W	Defines the top address of the I/O address range for the bridge to determine when to forward I/O transactions from one interface to the other. The upper 4 bits correspond to address bits [15:12] and are writable. The lower 12 bits corresponding to address bits [11:0] are assumed to be FFFh. The upper 16 bits corresponding to address bits [31:16] are defined in the I/O limit address upper 16 bits address register Reset to 0

12.1.16 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 indicate PI7C8152x is capable of 66MHz operation on the secondary interface Reset to 1
22	Reserved	R/O	Reset to 0
23	Fast Back-to-Back Capable	R/O	Set to 1 to indicate PI7C8152x is capable of decoding fast back-to-back transactions on the secondary interface to different targets Reset to 1
24	Data Parity Error Detected	R/WC	Set to 1 when S_PERR_L is asserted and bit 6 of command register is set Reset to 0
26:25	DEVSEL_L timing	R/O	DEVSEL# timing (medium decoding) 00: fast DEVSEL_L decoding 01: medium DEVSEL_L decoding 10: slow DEVSEL_L decoding 11: reserved Reset to 01
27	Signaled Target Abort	R/WC	Set to 1 (by a target device) whenever a target abort cycle occurs on its secondary interface Reset to 0
28	Received Target Abort	R/WC	Set to 1 (by a master device) whenever transactions on its secondary interface are terminated with target abort Reset to 0
29	Received Master Abort	R/WC	Set to 1 (by a master) when transactions on its secondary interface are terminated with Master Abort Reset to 0
30	Received System Error	R/WC	Set to 1 when S_SERR_L is asserted Reset to 0
31	Detected Parity Error	R/WC	Set to 1 when address or data parity error is detected on the secondary interface Reset to 0

12.1.17 MEMORY BASE ADDRESS REGISTER – OFFSET 20h

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

12.1.18 MEMORY LIMIT ADDRESS REGISTER – OFFSET 20h

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

12.1.19 PEFETCHABLE MEMORY BASE ADDRESS REGISTER – OFFSET 24h

Bit	Function	Type	Description
3:0	64-bit addressing	R/O	Indicates 64-bit addressing 0000: 32-bit addressing 0001: 64-bit addressing Reset to 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. The memory base register upper 32 bits contains the upper half of the base address.

12.1.20 PREFETCHABLE MEMORY LIMIT ADDRESS REGISTER – OFFSET 24h

Bit	Function	Type	Description
19:16	64-bit addressing	R/O	Indicates 64-bit addressing 0000: 32-bit addressing 0001: 64-bit addressing Reset to 1

Bit	Function	Type	Description
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. The memory limit upper 32 bits register contains the upper half of the limit address.

12.1.21 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. Reset to 0

12.1.22 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. Reset to 0

12.1.23 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. Reset to 0

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

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

12.1.25 ECP POINTER REGISTER – OFFSET 34h

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

12.1.26 INTERRUPT PIN REGISTER – OFFSET 3Ch

Bit	Function	Type	Description
15:8	Interrupt Pin	R/O	Interrupt pin not supported on the PI7C8152x Read as 0 to indicate PI7C8152x does not support the interrupt pin

12.1.27 BRIDGE CONTROL REGISTER – OFFSET 3Ch

Bit	Function	Type	Description
16	Parity Error Response	R/W	Controls the bridge's response to parity errors on the secondary interface. 0: ignore address and data parity errors on the secondary interface 1: enable parity error reporting and detection on the secondary interface Reset to 0
17	S_SERR_L enable	R/W	Controls the forwarding of S_SERR_L to the primary interface. 0: disable the forwarding of S_SERR_L to primary interface 1: enable the forwarding of S_SERR_L to primary interface Reset to 0
18	ISA enable	R/W	Modifies the bridge's response to ISA I/O addresses, applying only to those addresses falling within the I/O base and limit address registers and within the first 64KB of PCI I/O space. 0: forward all I/O addresses in the range defined by the I/O base and I/O limit registers 1: blocks forwarding of ISA I/O addresses in the range defined by the I/O base and I/O limit registers that are in the first 64KB of I/O space that address the last 768 bytes in each 1KB block. Secondary I/O transactions are forwarded upstream if the address falls within the last 768 bytes in each 1KB block Reset to 0
19	VGA enable	R/W	Controls the bridge's response to VGA compatible addresses. 0: does not forward VGA compatible memory and I/O addresses from primary to secondary 1: forward VGA compatible memory and I/O addresses from primary to secondary regardless of other settings Reset to 0
20	Reserved	R/O	Reserved. Returns 0 when read. Reset to 0
21	Master Abort Mode	R/W	Control's bridge's behavior responding to master aborts on secondary interface. 0: does not report master aborts (returns FFFF_FFFFh on reads and discards data on writes) 1: reports master aborts by signaling target abort if possible or by the assertion of P_SERR_L if enabled Reset to 0

Bit	Function	Type	Description
22	Secondary Interface Reset	R/W	Controls the assertion of S_RESET_L signal pin on the secondary interface 0: does not force the assertion of S_RESET_L pin 1: forces the assertion of S_RESET_L 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. 0: does not generate fast back-to-back transactions on the secondary 1: enables fast back-to-back transaction generation on the secondary Reset to 0
24	Primary Master Timeout	R/W	Determines the maximum number of PCI clock cycles the PI7C8152x waits for an initiator on the primary interface to repeat a delayed transaction request. 0: Primary discard timer counts 2^{15} PCI clock cycles. 1: Primary discard timer counts 2^{10} PCI clock cycles. Reset to 0
25	Secondary Master Timeout	R/W	Determines the maximum number of PCI clock cycles the PI7C8152x waits for an initiator on the primary interface to repeat a delayed transaction request. 0: Primary discard timer counts 2^{15} PCI clock cycles. 1: Primary discard timer counts 2^{10} PCI clock cycles. Reset to 0
26	Master Timeout Status	R/WC	This bit is set to 1 when either the primary master timeout counter or secondary master timeout counter expires. Reset to 0
27	Discard Timer P_SERR_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. 0: P_SERR_L is not asserted on the primary interface as a result of the expiration of either the Primary Discard Timer or the Secondary Discard Timer. 1: P_SERR_L is asserted on the primary interface as a result of the expiration of either the Primary Discard Timer or the Secondary Discard Timer. Reset to 0
31-28	Reserved	R/O	Reserved. Returns 0 when read. Reset to 0.

12.1.28 DIAGNOSTIC / CHIP CONTROL REGISTER – OFFSET 40h

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

Bit	Function	Type	Description
1	Memory Write Disconnect Control	R/W	Controls when the bridge (as a target) disconnects memory write transactions. 0: memory write disconnects at 4KB aligned address boundary 1: memory write disconnects at cache line aligned address boundary Reset to 0
3:2	Reserved	R/O	Reserved. Returns 0 when read. Reset to 0.
4	Secondary Bus Prefetch Disable	R/W	Controls the bridge's ability to prefetch during upstream memory read transactions 0: PI7C8152x prefetches and does not forward byte enable bits during upstream memory read transactions. 1: PI7C8152x requests only 1 DWORD from the target and forwards read byte enable bits during upstream memory reads. Reset to 0
7:5	Reserved	R/O	Reserved. Returns 0 when read. Reset to 0
8	Chip Reset	R/WR	Controls the chip and secondary bus reset. 0: PI7C8152x is ready for operation 1: Causes PI7C8152x to perform a chip reset. Data buffers, configuration registers, and both primary and secondary are reset to their initial states. PI7C8152x clears this bit once chip reset is complete. PI7C8152x can then be reconfigured.
9	Test Mode 1	R/W	Controls the ability to test PI7C8152x's behavior 0: minimum of 8 free space in data FIFO to accept memory burst writes 1: minimum of 1 free space in data FIFO to accept memory burst writes Reset to 0
11:10	Test Mode 2	R/W	Controls the ability to test PI7C8152x's behavior 00: enable out of order transactions between all 4 DTR requests 01: accept 3 DTR requests at a time and they may be out of order 10: only the 2 DTR requests at the top of the 2 FIFO's may be out of order 11: no out of order transactions supported between DTR requests Reset to 00
12	Test Mode 3	R/W	Controls the ability to test PI7C8152x's behavior 0: 4 memory write transactions can be accepted at a time 1: 2 memory write transactions can be accepted at a time Reset to 0
15:13	Reserved	R/O	Reserved. Returns 000 when read. Reset to 000.

12.1.29 ARBITER CONTROL REGISTER – OFFSET 40h

Bit	Function	Type	Description
24:16	Arbiter Control	R/W	Each bit controls whether a secondary bus master is assigned to the high priority group or the low priority group. Bits [19:16] correspond to request inputs S_REQ[3:0] 0: low priority 1: high priority Reset to 0
25	Priority of Secondary Interface	R/W	Controls whether the secondary interface of the bridge is in the high priority group or the low priority group. 0: low priority 1: high priority Reset to 1
31:26	Reserved	R/O	Reserved. Returns 0 when read. Reset to 0.

12.1.30 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 0: Disable flow through during a memory read transaction 1: Enable flow through during a memory read transaction Reset to 0
1	Park	R/W	Controls bus arbiter's park function 0: Park to last master 1: Park to the bridge – secondary port Reset to 0
3:2	Reserved	R/W	Reserved. Returns 0 when read. Reset to 0
4	Memory Read Data Buffer Control	R/W	Ability to control PI7C8152x's behavior when the data buffer is empty 0: start returning memory read data right away and inserts wait states if the data buffer is empty 1: start returning memory read data after 1 cache line of data and disconnects the master if the data buffer is empty Reset to 0
15:5	Reserved	R/O	Reserved. Returns 0 when read. Reset to 0

12.1.31 SECONDARY BUS ARBITER PREEMPTION CONTROL REGISTER – OFFSET 4Ch

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

12.1.32 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 PI7C8152x's ability to assert P_SERR_L when it is unable to transfer any read data from the target after 2^{24} attempts. 0: P_SERR_L is asserted if this event occurs and the SERR_L enable bit in the command register is set. 1: P_SERR_L is not asserted if this event occurs. Reset to 0
2	Posted Write Non-Delivery	R/W	Controls PI7C8152x's ability to assert P_SERR_L when it is unable to transfer delayed write data after 2^{24} attempts. 0: P_SERR_L is asserted if this event occurs and the SERR_L enable bit in the command register is set 1: P_SERR_L is not asserted if this event occurs Reset to 0
3	Target Abort During Posted Write	R/W	Controls PI7C8152x's ability to assert P_SERR_L when it receives a target abort when attempting to deliver posted write data. 0: P_SERR_L is asserted if this event occurs and the SERR_L enable bit in the command register is set 1: P_SERR_L is not asserted if this event occurs Reset to 0

Bit	Function	Type	Description
4	Master Abort On Posted Write	R/W	Controls PI7C8152x's ability to assert P_SERR_L when it receives a master abort when attempting to deliver posted write data. 0: P_SERR_L is asserted if this event occurs and the SERR# enable bit in the command register is set 1: P_SERR_L is not asserted if this event occurs Reset to 0
5	Delayed Write Non-Delivery	R/W	Controls PI7C8152x's ability to assert P_SERR# when it is unable to transfer delayed write data after 2^{24} attempts. 0: P_SERR_L is asserted if this event occurs and the SERR_L enable bit in the command register is set 1: P_SERR_L is not asserted if this event occurs Reset to 0
6	Delayed Read – No Data From Target	R/W	Controls PI7C8152x's ability to assert P_SERR_L when it is unable to transfer any read data from the target after 2^{24} attempts. 0: P_SERR_L is asserted if this event occurs and the SERR_L enable bit in the command register is set 1: P_SERR_L is not asserted if this event occurs Reset to 0
7	Reserved	R/O	Reserved. Returns 0 when read. Reset to 0

12.1.33 SECONDARY CLOCK CONTROL REGISTER – OFFSET 68h

Bit	Function	Type	Description
1:0	S_CLKOUT[0] disable	R/W	S_CLKOUT[0] (slot 0) Enable 00: enable S_CLKOUT[0] 01: enable S_CLKOUT[0] 10: enable S_CLKOUT[0] 11: disable S_CLKOUT[0] and driven HIGH Reset to 00
3:2	Clock 1 disable	R/W	S_CLKOUT[1] (slot 1) Enable 00: enable S_CLKOUT[1] 01: enable S_CLKOUT[1] 10: enable S_CLKOUT[1] 11: disable S_CLKOUT[1] and driven HIGH Reset to 00
5:4	Clock 2 disable	R/W	S_CLKOUT[2] (slot 2) Enable 00: enable S_CLKOUT[2] 01: enable S_CLKOUT[2] 10: enable S_CLKOUT[2] 11: disable S_CLKOUT[2] and driven HIGH Reset to 00

Bit	Function	Type	Description
7:6	Clock 3 disable	R/W	S_CLKOUT[3] (slot 3) Enable 00: enable S_CLKOUT[3] 01: enable S_CLKOUT[3] 10: enable S_CLKOUT[3] 11: disable S_CLKOUT[3] and driven HIGH Reset to 00
8	Clock 4 disable	R/W	S_CLKOUT[4] (device 1) Enable 0: enable S_CLKOUT[4] 1: disable S_CLKOUT[4] and driven HIGH Reset to 0
13:9	Reserved	RO	Reserved. Reset to 1Fh
15:14	Reserved	RO	Reserved. Reset to 00

12.1.34 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. 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. 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. Reset to 0
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. 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 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. 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. 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. Reset to 0.

12.1.35 PORT OPTION REGISTER – OFFSET 74h

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

Bit	Function	Type	Description
1	Primary Memory Read Command Alias Enable	R/W	Controls PI7C8152x's detection mechanism for matching memory read retry cycles from the initiator on the primary interface 0: exact matching memory read retry cycles from initiator on the primary interface 1: alias MEMRL or MEMRM to MEMR for memory read retry cycles from the initiator on the primary interface Reset to 1
2	Primary Memory Write Command Alias Enable	R/W	Controls PI7C8152x's detection mechanism for matching non-posted memory write retry cycles from the initiator on the primary interface 0: exact matching for non-posted memory read retry cycles from initiator on the primary interface 1: alias MEMWI to MEMW for non-posted memory read retry cycles from initiator on the primary interface Reset to 0
3	Secondary Memory Read Command Alias Enable	R/W	Controls PI7C8152x's detection mechanism for matching memory read retry cycles from the initiator on the secondary 0: exact matching for memory read retry cycles from initiator on the secondary interface 1: alias MEMRL or MEMRM to MEMR for memory read retry cycles from initiator on the secondary interface Reset to 1
4	Secondary Memory Write Command Alias Enable	R/W	Controls PI7C8152x's detection mechanism for matching non-posted memory write retry cycles from the initiator on the primary interface 0: exact matching for non-posted memory write retry cycles from initiator on the secondary interface 1: alias MEMWI to MEMW for non-posted memory write retry cycles from initiator on the secondary interface Reset to 0
5	Primary Memory Read Line/Multiple Alias Enable	R/W	Control's PI7C8152x's detection mechanism for matching memory read line/multiple cycles from the initiator on the primary interface 0: exact matching for memory read line/multiple retry cycles from the initiator on the primary interface 1: alias MEMRL to MEMRM or MEMRM to MEMRL for memory read retry cycles from the initiator on the primary interface Reset to 1
6	Secondary Memory Read Line/Multiple Alias Enable	R/W	Control's PI7C8152x's detection mechanism for matching memory read line/multiple cycles from the initiator on the secondary interface 0: exact matching for memory read line/multiple retry cycles from the initiator on the secondary interface 1: alias MEMRL to MEMRM or MEMRM to MEMRL for memory read retry cycles from the initiator on the secondary interface Reset to 1

Bit	Function	Type	Description
7	Primary Memory Write and Invalidate Command Alias Disable	R/W	Controls PI7C8152x's detection mechanism for matching non-posted memory write and invalidate cycles from the initiator on the primary interface 0: When accepting MEMWI command at the primary interface, PI7C8152x converts MEMWI to MEMW command on the destination interface 1: When accepting MEMWI command at the primary interface, PI7C8152x does not convert MEMWI to MEMW command on the destination interface Reset to 0
8	Secondary Memory Write and Invalidate Command Alias Disable	R/W	Controls PI7C8152x's detection mechanism for matching non-posted memory write and invalidate cycles from the initiator on the secondary interface 0: When accepting MEMWI command at the secondary interface, PI7C8152x converts MEMWI to MEMW command on the destination interface 1: When accepting MEMWI command at the secondary interface, PI7C8152x does not convert MEMWI to MEMW command on the destination interface Reset to 0
9	Enable Long Request	R/W	Controls PI7C8152x's ability to enable long requests for lock cycles 0: normal lock operation 1: enable long request for lock cycle Reset to 0
10	Enable Secondary To Hold Request Longer	R/W	Control's PI7C8152x's ability to enable the secondary bus to hold requests longer. 0: internal secondary master will release REQ_L after FRAME_L assertion 1: internal secondary master will hold REQ_L until there is no transactions pending in FIFO or until terminated by target Reset to 1
11	Enable Primary To Hold Request Longer	R/W	Control's PI7C8152x's ability to hold requests longer at the Primary Port. 0: internal Primary master will release REQ_L after FRAME_L assertion 1: internal Primary master will hold REQ_L until there is no transactions pending in FIFO or until terminated by target Reset to 1
15:12	Reserved	R/O	Reserved. Returns 0 when read. Reset to 0.

12.1.36 PRIMARY MASTER TIMEOUT COUNTER – OFFSET 80h

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

12.1.37 SECONDARY MASTER TIMEOUT COUNTER – OFFSET 80h

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

12.1.38 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.

12.1.39 NEXT ITEM POINTER REGISTER – OFFSET DCh

Bit	Function	Type	Description
15:8	Next Item Pointer	R/O	Read as 00h. No other ECP registers.

12.1.40 POWER MANAGEMENT CAPABILITIES REGISTER – OFFSET DCh

Bit	Function	Type	Description
18:16	Power Management Revision	R/O	Read as 010 to indicate the device is compliant to Revision 1.1 of *PCI Power Management Interface Specifications.*
19	PME_L Clock	R/O	Read as 0 to indicate PI7C8152x does not support the PME_L pin.
20	Auxiliary Power	R/O	Read as 0 to indicate PI7C8152x does not support the PME_L pin or an auxiliary power source.
21	Device Specific Initialization	R/O	Read as 0 to indicate PI7C8152x does not have device specific initialization requirements.
24:22	Reserved	R/O	Read as 0
25	D1 Power State Support	R/O	Read as 1 to indicate PI7C8152x supports the D1 power management state.
26	D2 Power State Support	R/O	Read as 1 to indicate PI7C8152x supports the D2 power management state.
31:27	PME_L Support	R/O	Read as 0 to indicate PI7C8152x does not support the PME_L pin.

12.1.41 POWER MANAGEMENT DATA REGISTER – OFFSET E0h

Bit	Function	Type	Description
1:0	Power State	R/W	Indicates the current power state of PI7C8152x. If an unimplemented power state is written to this register, PI7C8152x 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 00: D0 state 01: D1 state 10: D2 state 11: D3 state Reset to 0
7:2	Reserved	R/O	Read as 0

Bit	Function	Type	Description
8	PME_L Enable	R/O	Read as 0 as PI7C8152x does not support the PME_L 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_L pin is not implemented.

12.1.42 PPB SUPPORT EXTENSIONS REGISER – OFFSET E0h

Bit	Function	Type	Description
21:16	Reserved	RO	Reserved Reset to 0
22	B2_B3 Support	RO	B2_B3 Support for $D3_{HOT}$ When BPCCE is HIGH, this bit is read as '1' to indicate that the secondary clock outputs will be stopped and driven LOW when the bridge is in $D3_{HOT}$. This bit is not defined if BPCCE is read as '0'.
23	Bus Power / Clock Control Enable	RO	Bus Power / Clock Control Enable When BPCCE is pulled HIGH, this bit is read as '1' to indicate that the bus power/clock control is enabled. When the BPCCE is tied LOW, this bit is read as '0' to indicate that the bus power/clock is disabled (secondary clocks are not disabled when this device is placed in $D3_{HOT}$).

13 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:

13.1 BRIDGE ACTIONS FOR VARIOUS CYCLE TYPES

Initiator	Target	Response
Master on Primary	Target on Primary	PI7C8152x does not respond. It detects this situation by decoding the address as well as monitoring the P_DEVSEL_L for other fast devices on the Primary Port.
Master on Primary	Target on Secondary	PI7C8152x 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 secondary 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	PI7C8152x does not respond and the cycle will terminate as master abort.
Master on Secondary	Target on the same Secondary Port	PI7C8152x does not respond.
Master on Secondary	Target on Primary	PI7C8152x 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 primary 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.

Initiator	Target	Response
Master on Secondary	Target not on Primary	PI7C8152x does not respond.

13.2 ABNORMAL TERMINATION (INITIATED BY BRIDGE MASTER)

13.2.1 MASTER ABORT

Master abort indicates that when PI7C8152x 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.

13.2.2 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.

13.2.3 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 tri-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 address parity error during a read cycle results in the bridge master initiating a Master Abort.

13.2.4 SECONDARY IDSEL MAPPING

When PI7C8152x 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 PI7C8152x.

14 ELECTRICAL AND TIMING SPECIFICATIONS

14.1 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 (Inputs and AV_{CC}, V_{DD} only]	-0.3V to 3.6V
DC Input Voltage	-0.5V to 5.5V
Junction Temperature (T_j)	125°C
Max Power (P_{MAX})	1.2W

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.

14.2 DC SPECIFICATIONS

Symbol	Parameter	Condition	Min.	Max.	Units	Notes
V_{DD}	Supply Voltage		3	3.6	V	
V_{ih}	Input HIGH Voltage		0.5 V_{DD}	V_{DD} + 0.5	V	1
V_{il}	Input LOW Voltage		-0.5	0.3 V_{DD}	V	1
V_{oh}	Output HIGH Voltage	I_{out} = -500µA	0.9V_{DD}		V	
V_{ol}	Output LOW Voltage	I_{out} = 1500µA		0.1 V_{DD}	V	
V_{oh5V}	5V Signaling Output HIGH Voltage	I_{out} = -2 mA	2.4		V	
V_{ol5V}	5V Signaling Output LOW Voltage	I_{out} = 6 mA		0.5	V	
I_{il}	Input Leakage Current	0 < V_{in} < V_{DD}		±10	µA	
C_{in}	Input Pin Capacitance			10	pF	
C_{CLK}	CLK Pin Capacitance		5	12	pF	
C_{IDSEL}	IDSEL Pin Capacitance			8	pF	
L_{pin}	Pin Inductance			20	nH	

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

14.3 AC SPECIFICATIONS

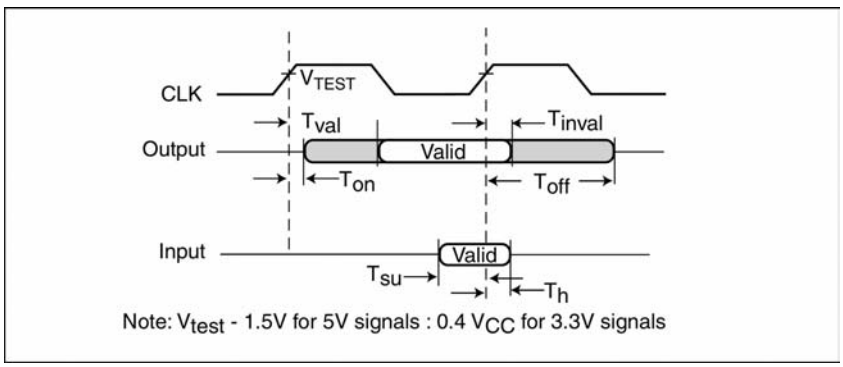


Figure 14-1 PCI SIGNAL TIMING MEASUREMENT CONDITIONS

		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 14-1* PCI Signal Timing Measurement Conditions.

2. All primary interface signals are synchronized to P_CLK. All secondary interface signals are synchronized to S_CLKIN.

3. Point-to-point signals are P_REQ_L, S_REQ_L[3:0], P_GNT_L, and S_GNT_L[3:0]. Bused signals are P_AD, P_CBE_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.

14.4 66MHZ PCI SIGNALING TIMING

Symbol	**Parameter**	**Condition**	**Min.**	**Max.**	**Units**
T_{SKEW}	SKEW among S_CLKOUT[4:0]		0	0.250	ns
T_{DELAY}	DELAY between PCLK and S_CLKOUT[4:0]	20pF load	2.82	4.22	
T_{CYCLE}	PCLK, S_CLKOUT[4:0] cycle time		15	30	
T_{HIGH}	PCLK, S_CLKOUT[4:0] HIGH time		6		
T_{LOW}	PCLK, S_CLKOUT[4:0] LOW time		6		

14.5 33MHZ PCI SIGNALING TIMING

Symbol	**Parameter**	**Condition**	**Min.**	**Max.**	**Units**
T_{SKEW}	SKEW among S_CLKOUT[4:0]		0	0.250	ns
T_{DELAY}	DELAY between PCLK and S_CLKOUT[4:0]	20pF load	2.82	4.22	
T_{CYCLE}	PCLK, S_CLKOUT[4:0] cycle time		30		
T_{HIGH}	PCLK, S_CLKOUT[4:0] HIGH time		11		
T_{LOW}	PCLK, S_CLKOUT[4:0] LOW time		11		

14.6 RESET TIMING

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

14.7 POWER CONSUMPTION

Parameter	Typical	Units
Power Consumption at 66MHz	812	mW
Supply Current, I_{CC}	246	mA

15 PACKAGE INFORMATION

15.1 160-PIN MQFP PACKAGE DIAGRAM

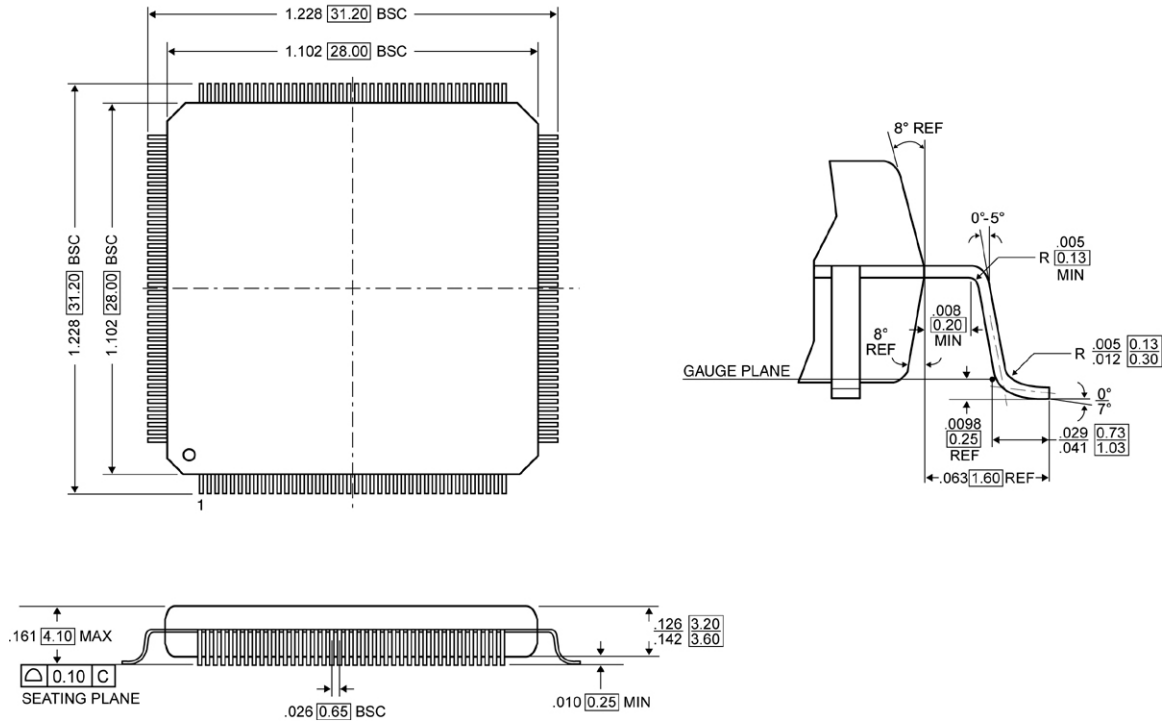


Figure 15-1 160-PIN MQFP PACKAGE OUTLINE

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

15.2 PART NUMBER ORDERING INFORMATION

Part Number	Speed	Pin – Package	Temperature
PI7C8152AMAE	66MHz	160 – MQFP	0°C to 85°C
PI7C8152BMAE	66MHz	160 – MQFP	0°C to 85°C
PI7C8152BMAIE	66MHz	160 – MQFP	-40°C to 85°C

NOTES: