21152 PCI-to-PCI Bridge

Datasheet

Product Features

- Complies fully with *Revision 2.1* of the *PCI Local Bus Specification*,
- Complies fully with *Revision 1.1* of the *PCI-to-PCI Bridge Architecture Specification*,
- Complies fully with the *Advanced Configuration Power Interface* (ACPI) *Specification*
- Complies fully with the *PCI Power Management Specification*, Revision 1.0[1]
- Implements delayed transactions for all PCI configuration, I/O, and memory read commands—up to three transactions simultaneously in each direction
- Allows 88 bytes of buffering (data and address) for posted memory write commands in each direction—up to three transactions simultaneously in each direction
- Allows 72 bytes of read data buffering in each direction
- Provides concurrent primary and secondary bus operation to isolate traffic
- Provides five secondary clock outputs
  - —Low skew, permitting direct drive of option slots
  - —Individual clock control through configuration space
- Provides arbitration support for four secondary bus devices
  - —A programmable 2-level arbiter
  - —Hardware disable control, permitting use of an external arbiter
- Provides enhanced address decoding
  - —A 32-bit I/O address range
  - —A 32-bit memory-mapped I/O address range
  - —A 64-bit prefetchable memory address range
  - —ISA-aware mode for legacy support in the first 64 KB of I/O address range
  - —VGA addressing and VGA palette snooping support
- Supports PCI transaction forwarding for the following commands
  - —All I/O and memory commands
  - —Type 1 to Type 1 configuration commands
  - —Type 1 to Type 0 configuration commands (downstream only)
  - —All Type 1 to special cycle configuration commands
- Includes downstream lock support
- Supports both 5 V and 3.3 V signaling environments

1. 21152–AB and later revisions only. The 21152–AA does not implement this feature.

Order Number: 278060, Revision: 002US
April 2005

Contents

Figures

Tables

Revision History

Date	Revision	Description
April 2005	002	Initial Public Release
December 2003	001	Initial Preliminary Release

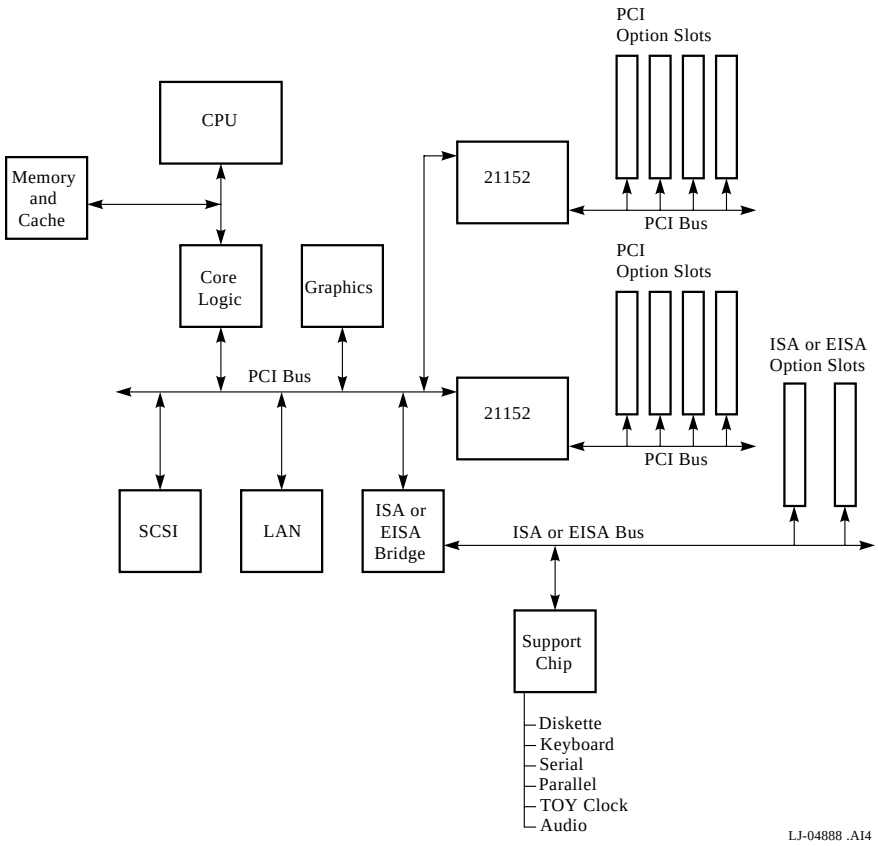
1.0 Introduction

The 21152 is a second-generation PCI-to-PCI bridge and is fully compliant with *PCI Local Bus Specification, Revision 2.1*. The 21152 is pin-to-pin compatible with the 21052, which is fully compliant with *PCI Local Bus Specification, Revision 2.0*.The 21152 provides full support for delayed transactions, enabling buffering of memory read, I/O, and configuration transactions. The 21152 has separate posted write, read data, and delayed transaction queues with significantly more buffering capability than first-generation bridges. In addition, the 21152 supports bi-directional buffering of simultaneous multiple posted write and delayed transactions.Among the features provided by the 21152 are a programmable 2-level secondary bus arbiter, individual secondary clock software control, and enhanced address decoding. The 21152 has sufficient clock and arbitration pins to support four PCI bus master devices directly on its secondary interface.

The 21152 allows the two PCI buses to operate concurrently. This means that a master and a target on the same PCI bus can communicate while the other PCI bus is busy. This traffic isolation may increase system performance in applications such as multimedia.

The 21152 makes it possible to extend a system's load capability limit beyond that of a single PCI bus by allowing motherboard designers to add more PCI devices, or more PCI option card slots, than a single PCI bus can support. Figure 1 illustrates the use of two 21152 PCI-to-PCI bridges on a system board. Each 21152 that is added to the board creates a new PCI bus that provides support for the additional PCI slots or devices.

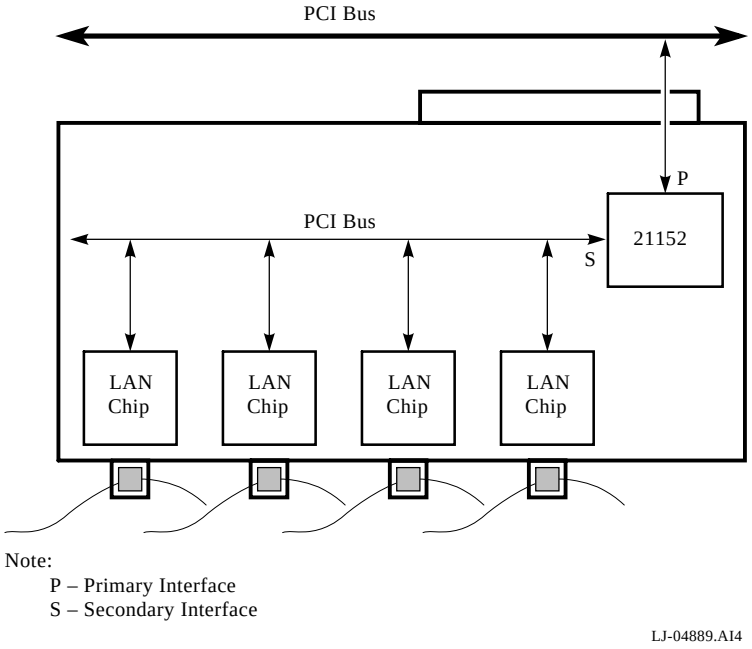
Figure 1. 21152 on the System Board



Option card designers can use the 21152 to implement multiple-device PCI option cards. Without a PCI-to-PCI bridge, PCI loading rules would limit option cards to one device. The *PCI Local Bus Specification* loading rules limit PCI option cards to a single connection per PCI signal in the option card connector. However, the 21152 overcomes this restriction by providing, on the option card, an independent PCI bus to which up to four devices can be attached.

Figure 2 shows how the 21152 enables the design of a multicomponent option card.

Figure 2. 21152 with Option Cards



1.1 Architecture

The 21152 internal architecture consists of the following major functions:

- PCI interface control logic for the primary and secondary PCI interfaces
- Data path and data path control logic
- Configuration register and configuration control logic
- Secondary bus arbiter

Figure 3 shows the major functional blocks of the 21152.

Figure 3. 21152 Block Diagram

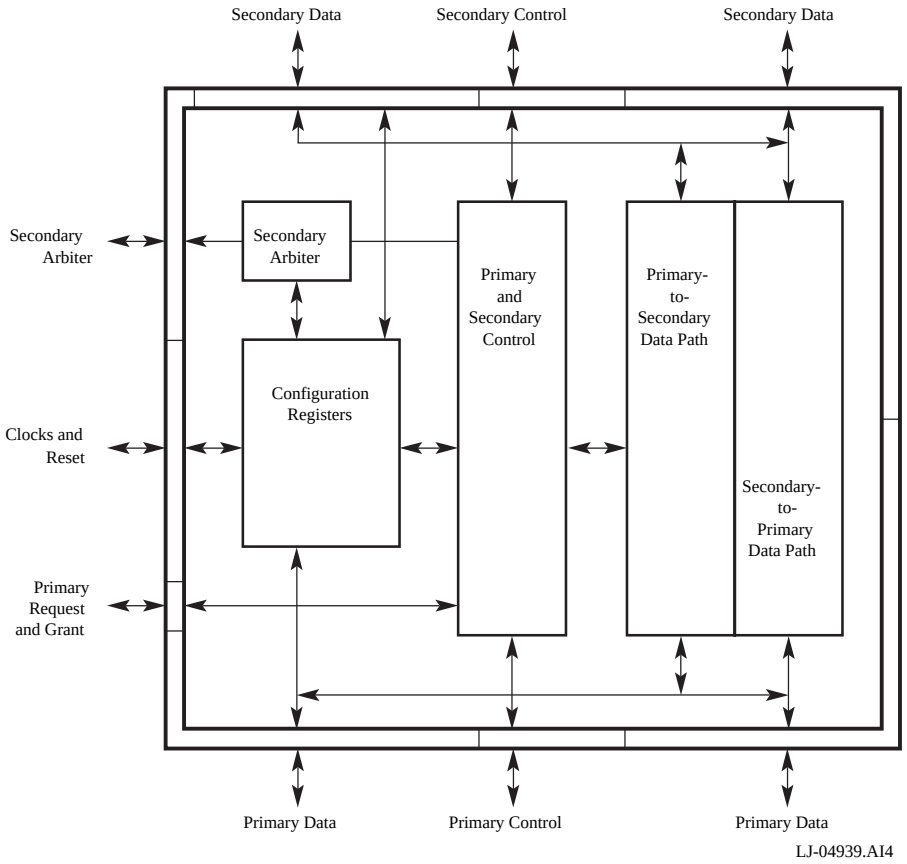


Table 1 describes the major functional blocks of the 21152.

Table 1. 21152 Function Blocks

Function Block	Description
Primary and Secondary Control	PCI interface control logic. This block contains state machines and control logic for the primary target interface, the primary master interface, the secondary target interface, and the secondary master interface. This block also contains logic that interfaces to the data path and the configuration block.
Primary-to-Secondary Data Path	Data path for data received on the primary interface and driven on the secondary interface. This block is used for write transactions initiated on the primary PCI bus and for returning read data for read transactions initiated on the secondary PCI bus. This block contains logic to store and, for posted write transactions, to increment the address of the current transaction. This block also performs bus command and configuration address format translations.
Secondary-to-Primary Data Path	Data path for data received on the secondary interface and driven on the primary interface. This block is used for write transactions initiated on the secondary PCI bus and for returning read data for read transactions initiated on the primary PCI bus. This block contains logic to store and, for posted write transactions, to increment the address of the current transaction. This block also performs bus command and configuration address format translations.
Configuration Registers	Configuration space registers and corresponding control logic. These registers are accessible from the primary interface only.
Secondary Bus Arbiter Control	Logic for secondary bus arbitration. This block receives **s_req_l<3:0>**, as well as the 21152 secondary bus request, and drives one of the **s_gnt_l<3:0>** lines or the 21152 secondary bus grant.

1.2 Data Path

The data path consists of a primary-to-secondary data path for transactions and data flowing in the downstream direction and a secondary-to-primary data path for transactions and data flowing in the upstream direction.

Both data paths have the following queues:

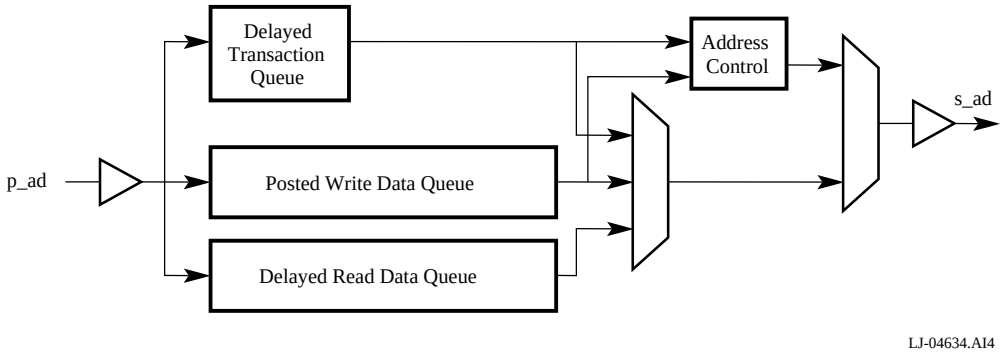
- Posted write queue
- Delayed transaction queue
- Read data queue

To prevent deadlocks and to maintain data coherency, a set of ordering rules is imposed on the forwarding of posted and delayed transactions across the 21152. The queue structure, along with the order in which the transactions in the queues are initiated and completed, supports these ordering requirements. Chapter 6.0 describes the 21152 ordering rules in detail.

Refer to Chapter 4.0 for a detailed description of 21152 PCI bus operation.

Figure 4 shows the 21152 data path for the downstream direction, and the following sections describe the data path queues.

Figure 4. 21152 Downstream Data Path



1.2.1 Posted Write Queue

The posted write queue contains the address and data of memory write transactions targeted for the opposite interface. The posted write transaction can consist of an arbitrary number of data phases, subject to the amount of space in the queue and disconnect boundaries. The posted write queue can contain multiple posted write transactions. The number of posted write transactions that can be queued at one time is dependent upon their burst size. The posted write queue consists of 88 bytes in each direction.

1.2.2 Delayed Transaction Queue

For a delayed write request transaction, the delayed transaction queue contains the address, bus command, 1 Dword of write data, byte enable bits, and parity. When the delayed write transaction is completed on the target bus, the write completion status is added to the corresponding entry.

For a delayed read request transaction, the delayed transaction queue contains the address and bus command, and for nonprefetchable read transactions, the byte enable bits. When the delayed read transaction is completed on the target bus, the read completion status corresponding to that transaction is added to the delayed request entry. Read data is placed in the read data queue.

The delayed transaction queue can hold up to three transactions (any combination of read and write transactions).

1.2.3 Read Data Queue

The read data queue contains read data transferred from the target during a delayed read completion. Read data travels in the opposite direction of the transaction. The primary-to-secondary read data queue contains read data corresponding to a delayed read transaction residing in the secondary-to-primary delayed transaction queue. The secondary-to-primary read data queue contains read data corresponding to a delayed read transaction in the primary-to-secondary delayed transaction queue. The amount of read data per transaction depends on the amount of space in the queue and disconnect boundaries.

Read data for up to three transactions, subject to the burst size of the read transactions and available queue space, can be stored. The read data queue for the 21152 consists of 72 bytes in each direction.

§ §

2.0 Signal Pins

This chapter provides detailed descriptions of the 21152 signal pins, grouped by function.

Table 2 describes the signal pin functional groups, and the following sections describe the signals in each group.

Table 2. Signal Pin Functional Groups

Function	Description
Primary PCI bus interface signal pins	All PCI pins required by the *PCI-to-PCI Bridge Architecture Specification,* Revision 1.1.
Secondary PCI bus interface signal pins	All PCI pins required by the *PCI-to-PCI Bridge Architecture Specification,* Revision 1.1.
Secondary PCI bus arbiter signal pins	Four request/grant pairs of pins for the secondary PCI bus. An arbiter enable control pin.
Clock signal pins	Two clock inputs (one for each PCI interface). Five clock outputs (for four external secondary PCI bus devices and also for the 21152).
Reset signal pins	A primary interface reset input. A secondary interface reset output.
Miscellaneous signal pins	Two input voltage signaling level pins.
Nand tree signal pins	An input and an output for the Nand tree diagnostic mechanism.

Table 3 defines the signal type abbreviations used in the signal tables:

Table 3. Signal Type Abbreviations

Signal Type	Description
I	Standard input only.
O	Standard output only.
TS	Tristate bidirectional.
STS	Sustained tristate. Active low signal must be pulled high for one cycle when deasserting.
OD	Standard open drain.

Note: The **_l** signal name suffix indicates that the signal is asserted when it is at a low voltage level and corresponds to the "#" suffix in the *PCI Local Bus Specification*. If this suffix is not present, the signal is asserted when it is at a high voltage level.

2.1 Primary PCI Bus Interface Signals

Table 4 describes the primary PCI bus interface signals.

Table 4. Primary PCI Bus Interface Signals (Sheet 1 of 3)

Signal Name	Type	Description
p_ad<31:0>	TS	**Primary PCI interface address/data.** These signals are a multiplexed address and data bus. During the address phase or phases of a transaction, the initiator drives a physical address on **p_ad<31:0>**. During the data phases of a transaction, the initiator drives write data, or the target drives read data, on **p_ad<31:0>**. When the primary PCI bus is idle, the 21152 drives **p_ad** to a valid logic level when **p_gnt_l** is asserted.
p_cbe_l<3:0>	TS	**Primary PCI interface command/byte enables.** These signals are a multiplexed command field and byte enable field. During the address phase or phases of a transaction, the initiator drives the transaction type on **p_cbe_l<3:0>.** When there are two address phases, the first address phase carries the dual address command and the second address phase carries the transaction type. For both read and write transactions, the initiator drives byte enables on **p_cbe_l<3:0>** during the data phases. When the primary PCI bus is idle, the 21152 drives **p_cbe_l** to a valid logic level when **p_gnt_l** is asserted.
p_par	TS	**Primary PCI interface parity.** Signal **p_par** carries the even parity of the 36 bits of **p_ad<31:0>** and **p_cbe_l<3:0>** for both address and data phases. Signal **p_par** is driven by the same agent that has driven the address (for address parity) or the data (for data parity). Signal **p_par** contains valid parity one cycle after the address is valid (indicated by assertion of **p_frame_l**), or one cycle after data is valid (indicated by assertion of **p_irdy_l** for write transactions and **p_trdy_l** for read transactions). Signal **p_par** is driven by the device driving read or write data one cycle after **p_ad** is driven. Signal **p_par** is tristated one cycle after the **p_ad** lines are tristated. Devices receiving data sample **p_par** as an input to check for possible parity errors. When the primary PCI bus is idle, the 21152 drives **p_par** to a valid logic level when **p_gnt_l** is asserted (one cycle after the **p_ad** bus is parked).
p_frame_l	STS	**Primary PCI interface FRAME#.** Signal **p_frame_l** is driven by the initiator of a transaction to indicate the beginning and duration of an access on the primary PCI bus. Signal **p_frame_l** assertion (falling edge) indicates the beginning of a PCI transaction. While **p_frame_l** remains asserted, data transfers can continue. The deassertion of **p_frame_l** indicates the final data phase requested by the initiator. When the primary PCI bus is idle, **p_frame_l** is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor.
p_irdy_l	STS	**Primary PCI interface IRDY#**. Signal **p_irdy_l** is driven by the initiator of a transaction to indicate the initiator's ability to complete the current data phase on the primary PCI bus. During a write transaction, assertion of **p_irdy_l** indicates that valid write data is being driven on the **p_ad** bus. During a read transaction, assertion of **p_irdy_l** indicates that the initiator is able to accept read data for the current data phase. Once asserted during a given data phase, **p_irdy_l** is not deasserted until the data phase completes. When the primary bus is idle, **p_irdy_l** is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor.
p_trdy_l	STS	**Primary PCI interface TRDY#.** Signal **p_trdy_l** is driven by the target of a transaction to indicate the target's ability to complete the current data phase on the primary PCI bus. During a write transaction, assertion of **p_trdy_l** indicates that the target is able to accept write data for the current data phase. During a read transaction, assertion of **p_trdy_l** indicates that the target is driving valid read data on the **p_ad** bus. Once asserted during a given data phase, **p_trdy_l** is not deasserted until the data phase completes. When the primary bus is idle, **p_trdy_l** is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor.

Table 4. Primary PCI Bus Interface Signals (Sheet 2 of 3)

Signal Name	Type	Description
p_devsel_l	STS	**Primary PCI interface DEVSEL#**. Signal **p_devsel_l** is asserted by the target, indicating that the device is accepting the transaction. As a target, the 21152 performs positive decoding on the address of a transaction initiated on the primary bus to determine whether to assert **p_devsel_l**. As an initiator of a transaction on the primary bus, the 21152 looks for the assertion of **p_devsel_l** within five cycles of **p_frame_l** assertion; otherwise, the 21152 terminates the transaction with a master abort. When the primary bus is idle, **p_devsel_l** is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor.
p_stop_l	STS	**Primary PCI interface STOP#**. Signal **p_stop_l** is driven by the target of the current transaction, indicating that the target is requesting the initiator to stop the current transaction on the primary bus. When **p_stop_l** is asserted in conjunction with **p_trdy_l** and **p_devsel_l** assertion, a disconnect with data transfer is being signaled. When **p_stop_l** and **p_devsel_l** are asserted, but **p_trdy_l** is deasserted, a target disconnect without data transfer is being signaled. When this occurs on the first data phase, that is, no data is transferred during the transaction, this is referred to as a target retry. When **p_stop_l** is asserted and **p_devsel_l** is deasserted, the target is signaling a target abort. When the primary bus is idle, **p_stop_l** is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor.
p_lock_l	I	**Primary PCI interface LOCK#**. Signal **p_lock_l** is deasserted during the first address phase of a transaction and is asserted one clock cycle later by an initiator attempting to perform an atomic operation that may take more than one PCI transaction to complete. The 21152 samples **p_lock_l** as a target and can propagate the lock across to the secondary bus. The 21152 does not drive **p_lock_l** as an initiator; that is, the 21152 does not propagate locked transactions upstream. When released by an initiator, **p_lock_l** is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor.
p_idsel	I	**Primary PCI interface IDSEL#**. Signal **p_idsel** is used as the chip select line for Type 0 configuration accesses to 21152 configuration space. When **p_idsel** is asserted during the address phase of a Type 0 configuration transaction, the 21152 responds to the transaction by asserting **p_devsel_l**.
p_perr_l	STS	**Primary PCI interface PERR#**. Signal **p_perr_l** is asserted when a data parity error is detected for data received on the primary interface. The timing of **p_perr_l** corresponds to **p_par** driven one cycle earlier and **p_ad** and **p_cbe_l** driven two cycles earlier. Signal **p_perr_l** is asserted by the target during write transactions, and by the initiator during read transactions. When the primary bus is idle, **p_perr_l** is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor.

Table 4. Primary PCI Bus Interface Signals (Sheet 3 of 3)

Signal Name	Type	Description
p_serr_l	OD	**Primary PCI interface SERR#**. Signal **p_serr_l** can be driven low by any device on the primary bus to indicate a system error condition. The 21152 can assert **p_serr_l** for the following reasons: Address parity error Posted write data parity error on target bus Secondary bus **s_serr_l** assertion 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 Signal **p_serr_l** is pulled up through an external resistor
p_req_l	TS	**Primary PCI bus REQ#**. Signal **p_req_l** is asserted by the 21152 to indicate to the primary bus arbiter that it wants to start a transaction on the primary bus.
p_gnt_l	I	**Primary PCI bus GNT#**. When asserted, **p_gnt_l** indicates to the 21152 that access to the primary bus is granted. The 21152 can start a transaction on the primary bus when the bus is idle and **p_gnt_l** is asserted. When the 21152 has not requested use of the bus and **p_gnt_l** is asserted, the 21152 must drive **p_ad**, **p_cbe_l**, and **p_par** to valid logic levels.

2.2 Secondary PCI Bus Interface Signals

Table 5 describes the secondary PCI bus interface signals.

Table 5. Secondary PCI Bus Interface Signals (Sheet 1 of 2)

Signal Name	Type	Description
s_ad<31:0>	TS	**Secondary PCI interface address/data**. These signals are a multiplexed address and data bus. During the address phase or phases of a transaction, the initiator drives a physical address on **s_ad<31:0>**. During the data phases of a transaction, the initiator drives write data, or the target drives read data, on **s_ad<31:0>**. When the secondary PCI bus is idle, the 21152 drives **s_ad** to a valid logic level when its secondary bus grant is asserted.
s_cbe_l<3:0>	TS	**Secondary PCI interface command/byte enables**. These signals are a multiplexed command field and byte enable field. During the address phase or phases of a transaction, the initiator drives the transaction type on **s_cbe_l<3:0>**. When there are two address phases, the first address phase carries the dual address command and the second address phase carries the transaction type. For both read and write transactions, the initiator drives byte enables on **s_cbe_l<3:0>** during the data phases. When the secondary PCI bus is idle, the 21152 drives **s_cbe_l** to a valid logic level when its secondary bus grant is asserted.
s_par	TS	**Secondary PCI interface parity**. Signal **s_par** carries the even parity of the 36 bits of **s_ad<31:0>** and **s_cbe_l<3:0>** for both address and data phases. Signal **s_par** is driven by the same agent that has driven the address (for address parity) or the data (for data parity). Signal **s_par** contains valid parity one cycle after the address is valid (indicated by assertion of **s_frame_l**), or one cycle after data is valid (indicated by assertion of **s_irdy_l** for write transactions and **s_trdy_l** for read transactions). Signal **s_par** is driven by the device driving read or write data one cycle after **s_ad** is driven. Signal **s_par** is tristated one cycle after the **s_ad** lines are tristated. Devices receive data sample **s_par** as an input to check for possible parity errors. When the secondary PCI bus is idle, the 21152 drives **s_par** to a valid logic level when its secondary bus grant is asserted (one cycle after the **s_ad** bus is parked).
s_frame_l	STS	**Secondary PCI interface FRAME#**. Signal **s_frame_l** is driven by the initiator of a transaction to indicate the beginning and duration of an access on the secondary PCI bus. Signal **s_frame_l** assertion (falling edge) indicates the beginning of a PCI transaction. While **s_frame_l** remains asserted, data transfers can continue. The deassertion of **s_frame_l** indicates the final data phase requested by the initiator. When the secondary PCI bus is idle, **s_frame_l** is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor.
s_irdy_l	STS	**Secondary PCI interface IRDY#**. Signal **s_irdy_l** is driven by the initiator of a transaction to indicate the initiator's ability to complete the current data phase on the secondary PCI bus. During a write transaction, assertion of **s_irdy_l** indicates that valid write data is being driven on the **s_ad** bus. During a read transaction, assertion of **s_irdy_l** indicates that the initiator is able to accept read data for the current data phase. Once asserted during a given data phase, **s_irdy_l** is not deasserted until the data phase completes. When the secondary bus is idle, **s_irdy_l** is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor.
s_trdy_l	STS	**Secondary PCI interface TRDY#**. Signal **s_trdy_l** is driven by the target of a transaction to indicate the target's ability to complete the current data phase on the secondary PCI bus. During a write transaction, assertion of **s_trdy_l** indicates that the target is able to accept write data for the current data phase. During a read transaction, assertion of **s_trdy_l** indicates that the target is driving valid read data on the **s_ad** bus. Once asserted during a given data phase, **s_trdy_l** is not deasserted until the data phase completes. When the secondary bus is idle, **s_trdy_l** is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor.

Table 5. Secondary PCI Bus Interface Signals (Sheet 2 of 2)

Signal Name	Type	Description
s_devsel_l	STS	**Secondary PCI interface DEVSEL#**. Signal **s_devsel_l** is asserted by the target, indicating that the device is accepting the transaction. As a target, the 21152 performs positive decoding on the address of a transaction initiated on the secondary bus in order to determine whether to assert **s_devsel_l**. As an initiator of a transaction on the secondary bus, the 21152 looks for the assertion of **s_devsel_l** within five cycles of **s_frame_l** assertion; otherwise, the 21152 terminates the transaction with a master abort. When the secondary bus is idle, **s_devsel_l** is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor.
s_stop_l	STS	**Secondary PCI interface STOP#**. Signal **s_stop_l** is driven by the target of the current transaction, indicating that the target is requesting the initiator to stop the current transaction on the secondary bus. When **s_stop_l** is asserted in conjunction with **s_trdy_l** and **s_devsel_l** assertion, a disconnect with data transfer is being signaled. When **s_stop_l** and **s_devsel_l** are asserted, but **s_trdy_l** is deasserted, a target disconnect without data transfer is being signaled. When this occurs on the first data phase, that is, no data is transferred during the transaction, this is referred to as a target retry. When **s_stop_l** is asserted and **s_devsel_l** is deasserted, the target is signaling a target abort. When the secondary bus is idle, **s_stop_l** is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor.
s_lock_l	STS	**Secondary PCI interface LOCK#**. Signal **s_lock_l** is deasserted during the first address phase of a transaction and is asserted one clock cycle later by the 21152 when it is propagating a locked transaction downstream. The 21152 does not propagate locked transactions upstream. The 21152 continues to assert **s_lock_l** until the address phase of the next locked transaction, or until the lock is released. When the lock is released, **s_lock_l** is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor.
s_perr_l	STS	**Secondary PCI interface PERR#**. Signal **s_perr_l** is asserted when a data parity error is detected for data received on the secondary interface. The timing of **s_perr_l** corresponds to **s_par** driven one cycle earlier and **s_ad** driven two cycles earlier. Signal **s_perr_l** is asserted by the target during write transactions, and by the initiator during read transactions. When the secondary bus is idle, **s_perr_l** is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor.
s_serr_l	I	**Secondary PCI interface SERR#**. Signal **s_serr_l** can be driven low by any device except the 21152 on the secondary bus to indicate a system error condition. The 21152 samples **s_serr_l** as an input and conditionally forwards it to the primary bus on **p_serr_l**. The 21152 does not drive **s_serr_l**. Signal **s_serr_l** is pulled up through an external resistor.

2.3 Secondary Bus Arbitration Signals

Table 6 describes the secondary bus arbitration signals.

Table 6. Secondary PCI Bus Arbitration Signals

Signal Name	Type	Description
s_req_l<3:0>	I	**Secondary PCI interface REQ#s**. The 21152 accepts four request inputs, **s_req_l<3:0>**, into its secondary bus arbiter. The 21152 request input to the arbiter is an internal signal. Each request input can be programmed to be in either a high priority rotating group or a low priority rotating group. An asserted level on an **s_req_l** pin indicates that the corresponding master wants to initiate a transaction on the secondary PCI bus. If the internal arbiter is disabled (**s_cfn_l** tied high), **s_req_l<0>** is reconfigured to be an external secondary grant input for the 21152. In this case, an asserted level on **s_req_l<0>** indicates that the 21152 can start a transaction on the secondary PCI bus if the bus is idle.
s_gnt_l<3:0>	TS	**Secondary PCI interface GNT#s**. The 21152 secondary bus arbiter can assert one of four secondary bus grant outputs, **s_gnt_l<3:0>**, to indicate that an initiator can start a transaction on the secondary bus if the bus is idle. The 21152's secondary bus grant is an internal signal. A programmable 2-level rotating priority algorithm is used. If the internal arbiter is disabled (**s_cfn_l** tied high), **s_gnt_l<0>** is reconfigured to be an external secondary bus request output for the 21152. The 21152 asserts this signal whenever it wants to start a transaction on the secondary bus.
s_cfn_l	I	**Secondary PCI central function enable**. When tied low, **s_cfn_l** enables the 21152 secondary bus arbiter. When tied high, **s_cfn_l** disables the internal arbiter. An external secondary bus arbiter must then be used. Signal **s_req_l<0>** is reconfigured to be the 21152 secondary bus grant input, and **s_gnt_l<0>** is reconfigured to be the 21152 secondary bus request output, when an external arbiter is used. Secondary bus parking is done when **s_req_l<0>** is asserted, the secondary bus is idle, and the 21152 does not want to initiate a transaction.

2.4 Clock Signals

Table 7 describes the clock signals.

Table 7. Clock Signals

Signal Name	Type	Description
p_clk	I	**Primary interface PCI CLK**. Provides timing for all transactions on the primary PCI bus. All primary PCI inputs are sampled on the rising edge of **p_clk**, and all primary PCI outputs are driven from the rising edge of **p_clk**. Frequencies supported by the 21152 range from 0 MHz to 33 MHz.
s_clk	I	**Secondary interface PCI CLK**. Provides timing for all transactions on the secondary PCI bus. All secondary PCI inputs are sampled on the rising edge of **s_clk**, and all secondary PCI outputs are driven from the rising edge of **s_clk**. Frequencies supported by the 21152 range from 0 MHz to 33 MHz.
s_clk_o<4:0>	O	**Secondary interface PCI CLK outputs**. Signals **s_clk_o<4:0>** are 5 clock outputs generated from the primary interface clock input, **p_clk**. These clocks operate at the same frequency of **p_clk**. When these clocks are used, one of the clock outputs must be fed back to the secondary clock input, **s_clk**. Unused clock outputs can be disabled by writing the secondary clock disable bits in configuration space; otherwise, terminate them electrically.

2.5 Reset Signals

Table 8 describes the reset signals.

Table 8. Reset Signals

Signal Name	Type	Description
bpcc[1]	I	**Bus power/clock control management pin**. When tied high and when the 21152 is placed in the $D3_{hot}$ power state, enables the 21152 to place the secondary bus in the B2 power state. The 21152 will disable the secondary clocks and drive them to 0. When tied low, placing the 21152 in the $D3_{hot}$ power state has no effect on the secondary bus clocks.
p_rst_l	I	**Primary PCI bus RST#**. Signal **p_rst_l** forces the 21152 to a known state. All register state is cleared, and all primary PCI bus outputs are tristated. Signal **p_rst_l** is asynchronous to **p_clk**.
s_rst_l	O	**Secondary PCI bus RST#**. Signal **s_rst_l** is driven by the 21152 and acts as the PCI reset for the secondary bus. The 21152 asserts **s_rst_l** when any of the following conditions are met: Signal **p_rst_l** is asserted. The secondary reset bit in the bridge control register in configuration space is set. The chip reset bit in the diagnostic control register in configuration space is set. When the 21152 asserts **s_rst_l**, it tristates all secondary control signals and drives zeros on **s_ad**, **s_cbe_l**, and **s_par**. Signal **s_rst_l** remains asserted until **p_rst_l** is deasserted, and the secondary reset bit is clear. Assertion of **s_rst_l** by itself does not clear register state, and configuration registers are still accessible from the primary PCI interface.

1. 21152–AB and later revisions only.

2.6 Miscellaneous Signals

Table 9 describes the miscellaneous signals.

Table 9. Miscellaneous Signals

Signal Name	Type	Description
p_vio	I	**Primary interface I/O voltage**. This signal must be tied to either 3.3 V or 5 V, corresponding to the signaling environment of the primary PCI bus as described in the *PCI Local Bus Specification*, *Revision 2.1*. When any device on the primary PCI bus uses 5-V signaling levels, tie **p_vio** to 5 V. Signal **p_vio** is tied to 3.3 V only when all the devices on the primary bus use 3.3-V signaling levels.
s_vio	I	**Secondary interface I/O voltage**. This signal must be tied to either 3.3 V or 5 V, corresponding to the signaling environment of the secondary PCI bus as described in the *PCI Local Bus Specification, Revision 2.1*. When any device on the secondary PCI bus uses 5-V signaling levels, tie **s_vio** to 5 V. Signal **s_vio** is tied to 3.3 V only when all the devices on the secondary bus use 3.3-V signaling levels.

2.7 Nand Tree Signals

Table 10 describes the Nand tree signals.

Table 10. Nand Tree Signals

Signal Name	Type	Description
goz_l	I	**Diagnostic tristate control.** This signal, when asserted, tristates all bidirectional and tristatable output pins.
nand_out	O	**Nand tree diagnostic output**. This signal is dedicated to the diagnostic Nand tree. The Nand tree starts at **s_cfn_l** and runs clockwise. All inputs, except **p_clk** and **s_clk,** are used in the Nand tree. The **goz_l** signal should be asserted when the Nand tree mechanism is used.

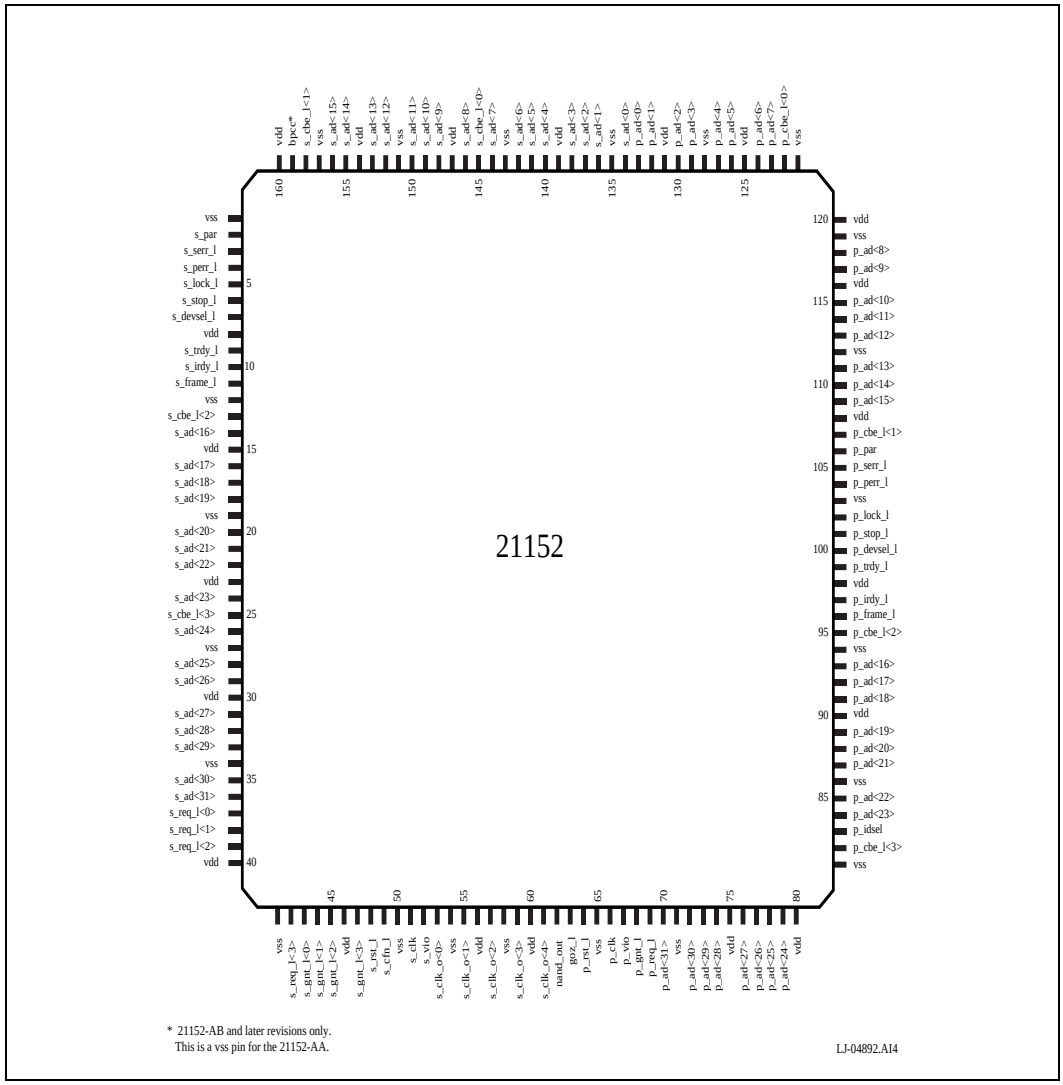
§ §

3.0 Pin Assignment

This chapter describes the 21152 pins. It provides numeric and alphanumeric lists of the pins and includes a diagram showing 21152 pin assignment.

Figure 5 shows the 21152 pins. Table 12 lists pins by location in numeric order and Table 14 lists pins by signal names in alphanumeric order.

Figure 5. 21152 Pinout Diagram



3.1 Pin Location List (Numeric)

Table 11 provides the signal type abbreviations used in Table 12. Table 12 lists the 21152 pins in numeric order, showing the location, signal name, and signal type of each pin.

Table 11. Signal Type Abbreviations

Signal Type	Description
I	Standard input only.
O	Standard output only.
P	Power.
TS	Tristate bidirectional.
STS	Sustained tristate. Active low signal must be pulled high for one cycle when deasserting.
OD	Standard open drain.

Table 12. Pin Location List (Numeric) (Sheet 1 of 3)

PQFP Location	Signal Name	Type	PQFP Location	Signal Name	Type
1	**vss**	P	24	**s_ad<23>**	TS
2	**s_par**	TS	25	**s_cbe_l<3>**	TS
3	**s_serr_l**	I	26	**s_ad<24>**	TS
4	**s_perr_l**	STS	27	**vss**	P
5	**s_lock_l**	STS	28	**s_ad<25>**	TS
6	**s_stop_l**	STS	29	**s_ad<26>**	TS
7	**s_devsel_l**	STS	30	**vdd**	P
8	**vdd**	P	31	**s_ad<27>**	TS
9	**s_trdy_l**	STS	32	**s_ad<28>**	TS
10	**s_irdy_l**	STS	33	**s_ad<29>**	TS
11	**s_frame_l**	STS	34	**vss**	P
12	**vss**	P	35	**s_ad<30>**	TS
13	**s_cbe_l<2>**	TS	36	**s_ad<31>**	TS
14	**s_ad<16>**	TS	37	**s_req_l<0>**	I
15	**vdd**	P	38	**s_req_l<1>**	I
16	**s_ad<17>**	TS	39	**s_req_l<2>**	I
17	**s_ad<18>**	TS	40	**vdd**	P
18	**s_ad<19>**	TS	41	**vss**	P
19	**vss**	P	42	**s_req_l<3>**	I
20	**s_ad<20>**	TS	43	**s_gnt_l<0>**	TS
21	**s_ad<21>**	TS	44	**s_gnt_l<1>**	TS
22	**s_ad<22>**	TS	45	**s_gnt_l<2>**	TS
23	**vdd**	P	46	**vdd**	P

Table 12. Pin Location List (Numeric) (Sheet 2 of 3)

PQFP Location	Signal Name	Type	PQFP Location	Signal Name	Type
47	**s_gnt_l<3>**	TS	84	**p_ad<23>**	TS
48	**s_rst_l**	O	85	**p_ad<22>**	TS
49	**s_cfn_l**	I	86	**vss**	P
50	**vss**	P	87	**p_ad<21>**	TS
51	**s_clk**	I	88	**p_ad<20>**	TS
52	**s_vio**	I	89	**p_ad<19>**	TS
53	**s_clk_o<0>**	O	90	**vdd**	P
54	**vss**	P	91	**p_ad<18>**	TS
55	**s_clk_o<1>**	O	92	**p_ad<17>**	TS
56	**vdd**	P	93	**p_ad<16>**	TS
57	**s_clk_o<2>**	O	94	**vss**	P
58	**vss**	P	95	**p_cbe_l<2>**	TS
59	**s_clk_o<3>**	O	96	**p_frame_l**	STS
60	**vdd**	P	97	**p_irdy_l**	STS
61	**s_clk_o<4>**	O	98	**vdd**	P
62	**nand_out**	O	99	**p_trdy_l**	STS
63	**goz_l**	I	100	**p_devsel_l**	STS
64	**p_rst_l**	I	101	**p_stop_l**	STS
65	**vss**	P	102	**p_lock_l**	STS
66	**p_clk**	I	103	**vss**	P
67	**p_vio**	I	104	**p_perr_l**	STS
68	**p_gnt_l**	I	105	**p_serr_l**	OD
69	**p_req_l**	TS	106	**p_par**	TS
70	**p_ad<31>**	TS	107	**p_cbe_l<1>**	TS
71	**vss**	P	108	**vdd**	P
72	**p_ad<30>**	TS	109	**p_ad<15>**	TS
73	**p_ad<29>**	TS	110	**p_ad<14>**	TS
74	**p_ad<28>**	TS	111	**p_ad<13>**	TS
75	**vdd**	P	112	**vss**	P
76	**p_ad<27>**	TS	113	**p_ad<12>**	TS
77	**p_ad<26>**	TS	114	**p_ad<11>**	TS
78	**p_ad<25>**	TS	115	**p_ad<10>**	TS
79	**p_ad<24>**	TS	116	**vdd**	P
80	**vdd**	P	117	**p_ad<9>**	TS
81	**vss**	P	118	**p_ad<8>**	TS
82	**p_cbe_l<3>**	TS	119	**vss**	P
83	**p_idsel**	I	120	**vdd**	P

Table 12. Pin Location List (Numeric) (Sheet 3 of 3)

PQFP Location	Signal Name	Type	PQFP Location	Signal Name	Type
121	**vss**	P	141	**s_ad<5>**	TS
122	**p_cbe_l<0>**	TS	142	**s_ad<6>**	TS
123	**p_ad<7>**	TS	143	**vss**	P
124	**p_ad<6>**	TS	144	**s_ad<7>**	TS
125	**vdd**	P	145	**s_cbe_l<0>**	TS
126	**p_ad<5>**	TS	146	**s_ad<8>**	TS
127	**p_ad<4>**	TS	147	**vdd**	P
128	**vss**	P	148	**s_ad<9>**	TS
129	**p_ad<3>**	TS	149	**s_ad<10>**	TS
130	**p_ad<2>**	TS	150	**s_ad<11>**	TS
131	**vdd**	P	151	**vss**	P
132	**p_ad<1>**	TS	152	**s_ad<12>**	TS
133	**p_ad<0>**	TS	153	**s_ad<13>**	TS
134	**s_ad<0>**	TS	154	**vdd**	P
135	**vss**	P	155	**s_ad<14>**	TS
136	**s_ad<1>**	TS	156	**s_ad<15>**	TS
137	**s_ad<2>**	TS	157	**vss**	P
138	**s_ad<3>**	TS	158	**s_cbe_l<1>**	TS
139	**vdd**	P	159	**bpcc**[1]	I
140	**s_ad<4>**	TS	160	**vdd**	P

1. Pertains to the 21152–AB and later revisions only. For the 21152–AA, this pin is **vss**.

3.2 Pin Signal List (Alphanumeric)

Table 13 provides the signal type abbreviations used in Table 14. Table 14 lists the 21152 signal names in alphanumeric order, showing the signal name, location, and signal type of each pin.

Table 13. Signal Type Abbreviations

Signal Type	Description
I	Standard input only.
O	Standard output only.
P	Power.
TS	Tristate bidirectional.
STS	Sustained tristate. Active low signal must be pulled high for one cycle when deasserting.
OD	Standard open drain.

Table 14. Pin Signal List (Alphanumeric) (Sheet 1 of 3)

Pin Name	PQFP Location	Type	Pin Name	PQFP Location	Type
bpcc[1]	159	I	**p_ad<20>**	88	TS
goz_l	63	I	**p_ad<21>**	87	TS
nand_out	62	O	**p_ad<22>**	85	TS
p_ad<0>	133	TS	**p_ad<23>**	84	TS
p_ad<1>	132	TS	**p_ad<24>**	79	TS
p_ad<2>	130	TS	**p_ad<25>**	78	TS
p_ad<3>	129	TS	**p_ad<26>**	77	TS
p_ad<4>	127	TS	**p_ad<27>**	76	TS
p_ad<5>	126	TS	**p_ad<28>**	74	TS
p_ad<6>	124	TS	**p_ad<29>**	73	TS
p_ad<7>	123	TS	**p_ad<30>**	72	TS
p_ad<8>	118	TS	**p_ad<31>**	70	TS
p_ad<9>	117	TS	**p_cbe_l<0>**	122	TS
p_ad<10>	115	TS	**p_cbe_l<1>**	107	TS
p_ad<11>	114	TS	**p_cbe_l<2>**	95	TS
p_ad<12>	113	TS	**p_cbe_l<3>**	82	TS
p_ad<13>	111	TS	**p_clk**	66	I
p_ad<14>	110	TS	**p_devsel_l**	100	STS
p_ad<15>	109	TS	**p_frame_l**	96	STS
p_ad<16>	93	TS	**p_gnt_l**	68	I
p_ad<17>	92	TS	**p_idsel**	83	I
p_ad<18>	91	TS	**p_irdy_l**	97	STS
p_ad<19>	89	TS	**p_lock_l**	102	STS

Table 14. Pin Signal List (Alphanumeric) (Sheet 2 of 3)

Pin Name	PQFP Location	Type	Pin Name	PQFP Location	Type
p_par	106	TS	**s_ad<29>**	33	TS
p_perr_l	104	STS	**s_ad<30>**	35	TS
p_req_l	69	TS	**s_ad<31>**	36	TS
p_rst_l	64	I	**s_cbe_l<0>**	145	TS
p_serr_l	105	OD	**s_cbe_l<1>**	158	TS
p_stop_l	101	STS	**s_cbe_l<2>**	13	TS
p_trdy_l	99	STS	**s_cbe_l<3>**	25	TS
p_vio	67	I	**s_cfn_l**	49	I
s_ad<0>	134	TS	**s_clk**	51	I
s_ad<1>	136	TS	**s_clk_o<0>**	53	O
s_ad<2>	137	TS	**s_clk_o<1>**	55	O
s_ad<3>	138	TS	**s_clk_o<2>**	57	O
s_ad<4>	140	TS	**s_clk_o<3>**	59	O
s_ad<5>	141	TS	**s_clk_o<4>**	61	O
s_ad<6>	142	TS	**s_devsel_l**	7	STS
s_ad<7>	144	TS	**s_frame_l**	11	STS
s_ad<8>	146	TS	**s_gnt_l<0>**	43	TS
s_ad<9>	148	TS	**s_gnt_l<1>**	44	TS
s_ad<10>	149	TS	**s_gnt_l<2>**	45	TS
s_ad<11>	150	TS	**s_gnt_l<3>**	47	TS
s_ad<12>	152	TS	**s_irdy_l**	10	STS
s_ad<13>	153	TS	**s_lock_l**	5	STS
s_ad<14>	155	TS	**s_par**	2	TS
s_ad<15>	156	TS	**s_perr_l**	4	STS
s_ad<16>	14	TS	**s_req_l<0>**	37	I
s_ad<17>	16	TS	**s_req_l<1>**	38	I
s_ad<18>	17	TS	**s_req_l<2>**	39	I
s_ad<19>	18	TS	**s_req_l<3>**	42	I
s_ad<20>	20	TS	**s_rst_l**	48	O
s_ad<21>	21	TS	**s_serr_l**	3	I
s_ad<22>	22	TS	**s_stop_l**	6	STS
s_ad<23>	24	TS	**s_trdy_l**	9	STS
s_ad<24>	26	TS	**s_vio**	52	I
s_ad<25>	28	TS	**vdd**	8	P
s_ad<26>	29	TS	**vdd**	15	P
s_ad<27>	31	TS	**vdd**	23	P
s_ad<28>	32	TS	**vdd**	30	P

Table 14. Pin Signal List (Alphanumeric) (Sheet 3 of 3)

Pin Name	PQFP Location	Type	Pin Name	PQFP Location	Type
vdd	40	P	**vss**	27	P
vdd	46	P	**vss**	34	P
vdd	56	P	**vss**	41	P
vdd	60	P	**vss**	50	P
vdd	75	P	**vss**	54	P
vdd	80	P	**vss**	58	P
vdd	90	P	**vss**	65	P
vdd	98	P	**vss**	71	P
vdd	108	P	**vss**	81	P
vdd	116	P	**vss**	86	P
vdd	120	P	**vss**	94	P
vdd	125	P	**vss**	103	P
vdd	131	P	**vss**	112	P
vdd	139	P	**vss**	119	P
vdd	147	P	**vss**	121	P
vdd	154	P	**vss**	128	P
vdd	160	P	**vss**	135	P
vss	1	P	**vss**	143	P
vss	12	P	**vss**	151	P
vss	19	P	**vss**	157	P

1. Pertains to the 21152–AB and later revisions only. For the 21152–AA, this pin is **vss**.

§ §

4.0 PCI Bus Operation

This chapter presents detailed information about PCI transactions, transaction forwarding across the 21152, and transaction termination.

4.1 Types of Transactions

This section provides a summary of PCI transactions performed by the 21152.

Table 15 lists the command code and name of each PCI transaction. The Master and Target columns indicate 21152 support for each transaction when the 21152 initiates transactions as a master, on the primary bus and on the secondary bus, and when the 21152 responds to transactions as a target, on the primary bus and on the secondary bus.

Table 15. 21152 PCI Transactions

Type of Transaction	21152 Initiates as Master		21152 Responds as Target	
	Primary	Secondary	Primary	Secondary
0000 Interrupt acknowledge	No	No	No	No
0001 Special cycle	Yes	Yes	No	No
0010 I/O read	Yes	Yes	Yes	Yes
0011 I/O write	Yes	Yes	Yes	Yes
0100 Reserved	No	No	No	No
0101 Reserved	No	No	No	No
0110 Memory read	Yes	Yes	Yes	Yes
0111 Memory write	Yes	Yes	Yes	Yes
1000 Reserved	No	No	No	No
1001 Reserved	No	No	No	No
1010 Configuration read	No	Yes	Yes	No
1011 Configuration write	Type 1	Yes	Yes	Type 1
1100 Memory read multiple	Yes	Yes	Yes	Yes
1101 Dual address cycle	Yes	Yes	Yes	Yes
1110 Memory read line	Yes	Yes	Yes	Yes
1111 Memory write and invalidate	Yes	Yes	Yes	Yes

As indicated in Table 15, the following PCI commands are not supported by the 21152:

- The 21152 never initiates a PCI transaction with a reserved command code and, as a target, the 21152 ignores reserved command codes.
- The 21152 never initiates an interrupt acknowledge transaction and, as a target, the 21152 ignores interrupt acknowledge transactions. Interrupt acknowledge transactions are expected to reside entirely on the primary PCI bus closest to the host bridge.
- The 21152 does not respond to special cycle transactions and cannot guarantee delivery of a special cycle transaction to downstream buses, due to the broadcast nature of the special cycle command, and the inability to control the transaction as a target. To generate special cycle transactions on other PCI buses, either upstream or downstream, a Type 1 configuration command must be used.
- The 21152 does not generate Type 0 configuration transactions on the primary interface, nor does it respond to Type 0 configuration transactions on the secondary PCI interface. The *PCI-to-PCI Bridge Architecture Specification* does not support configuration from the secondary bus.

4.2 Address Phase

The standard PCI transaction consists of one or two address phases, followed by one or more data phases. An address phase always lasts one PCI clock cycle. The first address phase is designated by an asserting (falling) edge on the FRAME# signal.

The number of address phases depends on whether the address is 32 bits or 64 bits.

4.2.1 Single Address Phase

A 32-bit address uses a single address phase. This address is driven on AD<31:0>, and the bus command is driven on C/BE#<3:0>. The 21152 supports the linear increment address mode only, which is indicated when the low 2 address bits are equal to 0. If either of the low 2 address bits is nonzero, the 21152 automatically disconnects the transaction after the first data transfer.

4.2.2 Dual Address Phase

Dual address transactions are PCI transactions that contain two address phases specifying a 64-bit address.

The first address phase is denoted by the asserting edge of FRAME#.

The second address phase always follows on the next clock cycle.

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

The *PCI-to-PCI Bridge Architecture Specification* supports the use of dual address transactions in the prefetchable memory range only. Refer to Section 5.3.3 for a discussion of prefetchable address space. The 21152 supports dual address transactions in both the upstream and the downstream direction. The 21152 supports a programmable 64-bit address range in prefetchable memory for downstream forwarding of dual address transactions. Dual address transactions falling outside the prefetchable address range are forwarded upstream, but not downstream. Prefetching and posting are performed in a manner consistent with the guidelines given in this specification for each type of memory transaction in prefetchable memory space.

The 21152 uses the following transaction command codes:

- Memory write
- Memory write and invalidate
- Memory read
- Memory read line
- Memory read multiple

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

Any memory transactions addressing the first 4 GB space should use a single address phase; that is, the high 32 bits of a dual address transaction should never be 0.

4.3 Device Select (DEVSEL#) Generation

The 21152 always performs positive address decoding when accepting transactions on either the primary or secondary buses. The 21152 never subtractively decodes. Medium DEVSEL# timing is used on both interfaces.

4.4 Data Phase

The address phase or phases of a PCI transaction are followed by one or more data phases. A data phase is completed when IRDY# and either TRDY# or STOP# are asserted. A transfer of data occurs only when both IRDY# and TRDY# are asserted during the same PCI clock cycle. The last data phase of a transaction is indicated when FRAME# is deasserted and both TRDY# and IRDY# are asserted, or when IRDY# and STOP# are asserted. Refer to Section 4.8 for further discussion of transaction termination.

Depending on the command type, the 21152 can support multiple data phase PCI transactions. For a detailed description of how the 21152 imposes disconnect boundaries, refer to Section 4.5.4 for a description of write address boundaries and Section 4.6.3 for a description of read address boundaries.

4.5 Write Transactions

Write transactions are treated as either posted write or delayed write transactions.

Table 16 shows the method of forwarding used for each type of write operation.

Table 16. Write Transaction Forwarding

Type of Transaction	Type of Forwarding
Memory write	Posted
Memory write and invalidate	Posted
I/O write	Delayed
Type 1 configuration write	Delayed

4.5.1 Posted Write Transactions

Posted write forwarding is used for memory write and, for memory write and invalidate transactions.

When the 21152 determines that a memory write transaction is to be forwarded across the bridge, the 21152 asserts DEVSEL# with medium timing and TRDY# in the same cycle, provided that enough buffer space is available in the posted data queue for the address and at least 8 Dwords of data. This enables the 21152 to accept write data without obtaining access to the target bus. The 21152 can accept 1 Dword of write data every PCI clock cycle; that is, no target wait states are inserted. This write data is stored in internal posted write buffers and is subsequently delivered to the target.

The 21152 continues to accept write data until one of the following events occurs:

- The initiator terminates the transaction by deasserting FRAME# and IRDY#.
- An internal write address boundary is reached, such as a cache line boundary or an aligned 4 KB boundary, depending on the transaction type.
- The posted write data buffer fills up.
- The 21152 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 (the 21152 starts another transaction to deliver the rest of the write data).
- The target returns a target abort (the 21152 discards remaining write data).
- The master latency timer expires, and the 21152 no longer has the target bus grant (the 21152 starts another transaction to deliver remaining write data).

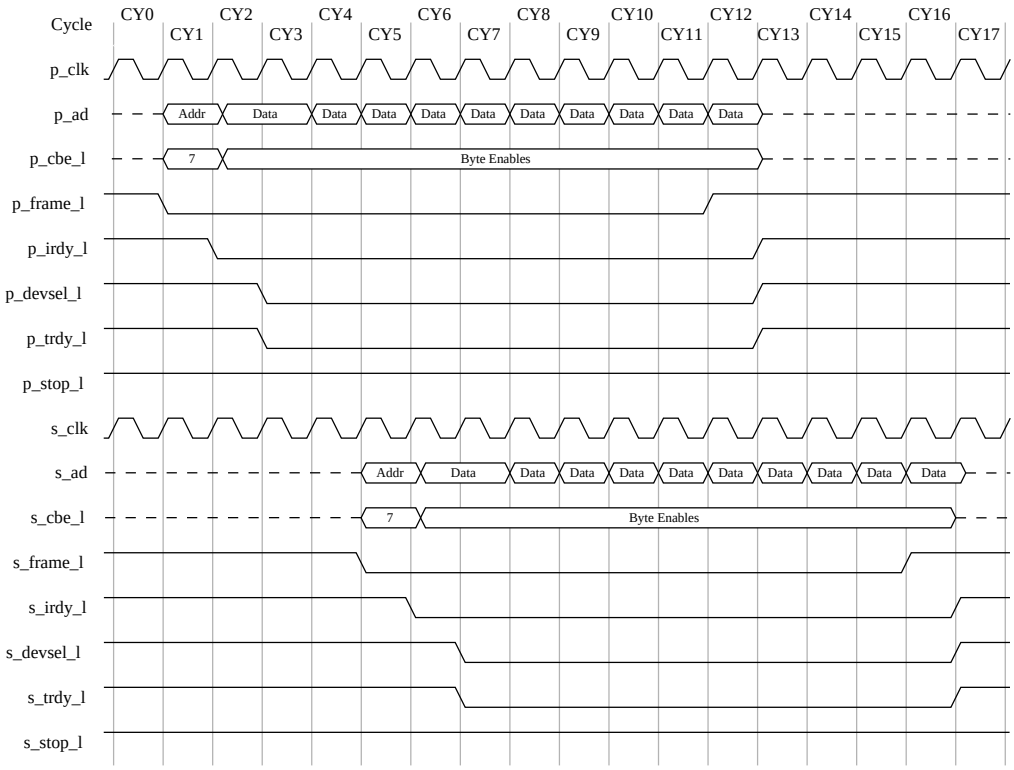
When one of the last two events occurs, the 21152 returns a target disconnect to the requesting initiator on this data phase to terminate the transaction.

Section 4.8.3.2 provides detailed information about how the 21152 responds to target termination during posted write transactions.

Once the posted write data moves to the head of the posted data queue, the 21152 asserts its request on the target bus. This can occur while the 21152 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, the 21152 asserts FRAME# and drives the stored write address out on the target bus. On the following cycle, the 21152 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, the 21152 can drive 1 Dword of write data each PCI clock cycle; that is, no master wait states are inserted. If write data is flowing through the 21152 and the initiator stalls, the 21152 may have to insert wait states on the target bus if the queue empties.

Figure 6 shows a memory write transaction in flow-through mode, where data is being removed from buffers on the target interface while more data is being transferred into the buffers on the master interface.

Figure 6. Flow-Through Posted Memory Write Transaction



LJ-04843.AI4

4.5.2 Memory Write and Invalidate Transactions

Posted write forwarding is used for memory write and invalidate transactions.

Memory write and invalidate transactions guarantee transfer of entire cache lines. If the write buffer fills before an entire cache line is transferred, the 21152 disconnects the transaction and converts it to a memory write transaction.

The 21152 disconnects memory write and invalidate commands at aligned cache line boundaries. The cache line size value in the 21152 cache line size register gives the number of Dwords in a cache line. For the 21152 to generate memory write and invalidate transactions, this cache line size value must be written to a value that is a nonzero power of 2 and less than or equal to 16 (that is, 1, 2, 4, 8, or 16 Dwords).

If the cache line size does not meet the memory write and invalidate conditions, that is, the value is 0, or is not a power of 2, or is greater than 16 Dwords, the 21152 treats the memory write and invalidate command as a memory write command. In this case, when the 21152 forwards the memory write and invalidate transaction to the target bus, it converts the command code to a memory write code and does not observe cache line boundaries.

If the value in the cache line size register does meet the memory write and invalidate conditions, that is, the value is a nonzero power of 2 less than or equal to 16 Dwords, the 21152 returns a target disconnect to the initiator either on a cache line boundary or when the posted write buffer fills.

For a cache line size of 16 Dwords, the 21152 disconnects a memory write and invalidate transaction on every cache line boundary. When the cache line size is 1, 2, 4, or 8 Dwords, the 21152 accepts another cache line if at least 8 Dwords of empty space remains in the posted write buffer. If less than Dwords of empty space remains, the 21152 disconnects on that cache line boundary.

When the memory write and invalidate transaction is disconnected before a cache line boundary is reached, typically because the posted write buffer fills, the transaction is converted to a memory write transaction.

4.5.3 Delayed Write Transactions

Delayed write forwarding is used for I/O write transactions and for 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 the 21152 forwards it as a delayed transaction, the 21152 claims the access by asserting DEVSEL# and returns a target retry to the initiator. During the address phase, the 21152 samples the bus command, address, and address parity one cycle later. After IRDY# is asserted, the 21152 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 (refer to Chapter 6.0), the 21152 initiates the transaction on the target bus. The 21152 transfers the write data to the target.

If the 21152 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.

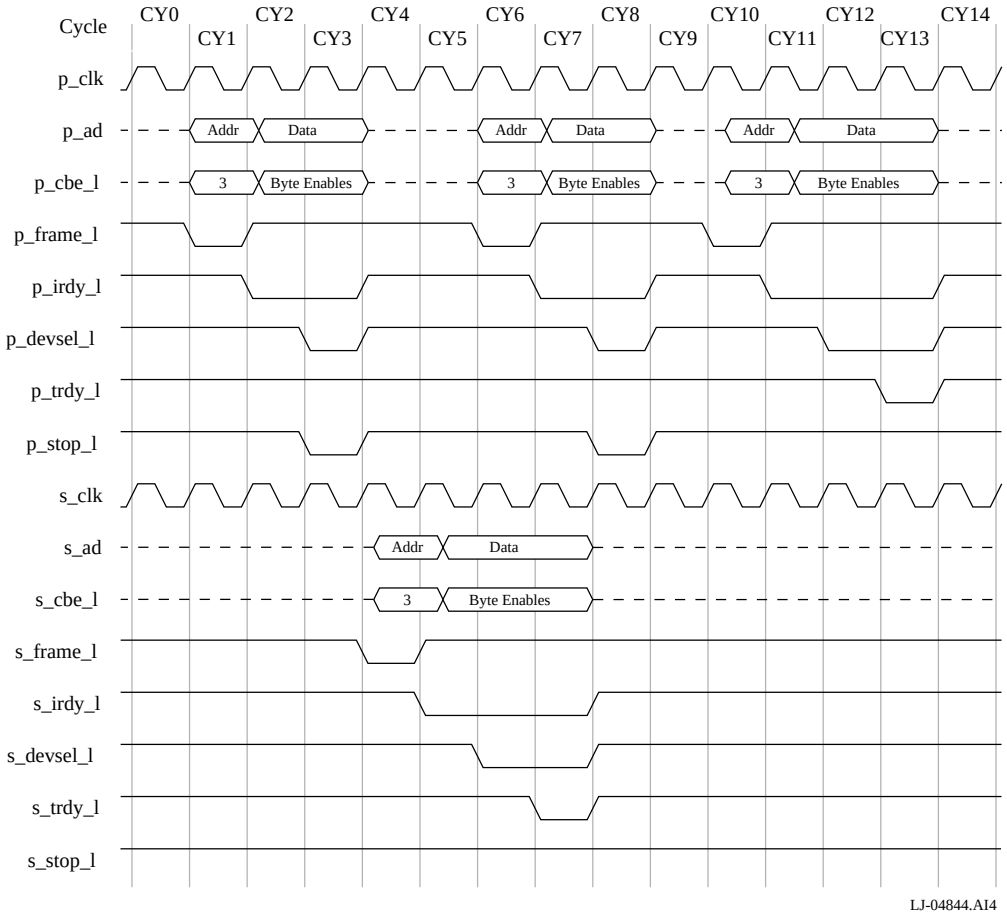
If the 21152 is unable to deliver write data after 2^{24} attempts, the 21152 ceases further write attempts and returns a target abort to the initiator. The delayed transaction is removed from the delayed transaction queue. The 21152 also asserts **p_serr_l** if the primary SERR# enable bit is set in the command register. Refer to Section 7.4 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 21152 claims the access by asserting DEVSEL# and returns TRDY# to the initiator, to indicate that the write data was transferred. If the initiator requests multiple Dwords, the 21152 also asserts STOP# in conjunction with TRDY# to signal a target disconnect. Note that only those bytes of write data with valid byte enable bits are compared. If any of the byte enable bits are turned off (driven high), the corresponding byte of write data is not compared.

If the initiator repeats the write transaction before the data has been transferred to the target, the 21152 returns a target retry to the initiator. The 21152 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, the 21152 does not make a new entry into the delayed transaction queue. Section 4.8.3.1 provides detailed information about how the 21152 responds to target termination during delayed write transactions.

Figure 7 shows a delayed write transaction forwarded downstream across the 21152.

Figure 7. Downstream Delayed Write Transaction



The 21152 implements a discard timer that starts counting when the delayed write completion is at the head of the delayed transaction queue. The initial value of this timer can be set to one of two values, selectable through both the primary and secondary master time-out bits in the bridge control register. If the initiator does not repeat the delayed write transaction before the discard timer expires, the 21152 discards the delayed write transaction from the delayed transaction queue. The 21152 also conditionally asserts **p_serr_l** (refer to Section 7.4).

4.5.4 Write Transaction Address Boundaries

The 21152 imposes internal address boundaries when accepting write data. The aligned address boundaries are used to prevent the 21152 from continuing a transaction over a device address boundary and to provide an upper limit on maximum latency. The 21152 returns a target disconnect to the initiator when it reaches the aligned address boundaries under the conditions shown in Table 17.

Table 17. 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]	4 KB 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	4 KB aligned address boundary
Posted memory write and invalidate	Cache line size = 1, 2, 4, 8	*n*th cache line boundary, where a cache line boundary is reached and less than 8 free Dwords of posted write buffer space remains
Posted memory write and invalidate	Cache line size = 16	16-Dword aligned address boundary

1. The memory write disconnect control bit is located in the chip control register at offset 40h in configuration space.

4.5.5 Buffering Multiple Write Transactions

The 21152 continues to accept posted memory write transactions as long as space for at least 1 Dword of data in the posted write data buffer remains. If the posted write data buffer fills before the initiator terminates the write transaction, the 21152 returns a target disconnect to the initiator.

Delayed write transactions are posted as long as at least one open entry in the 21152 delayed transaction queue exists. Therefore, several posted and delayed write transactions can exist in data buffers at the same time.

Refer to Chapter 6.0 for information about how multiple posted and delayed write transactions are ordered.

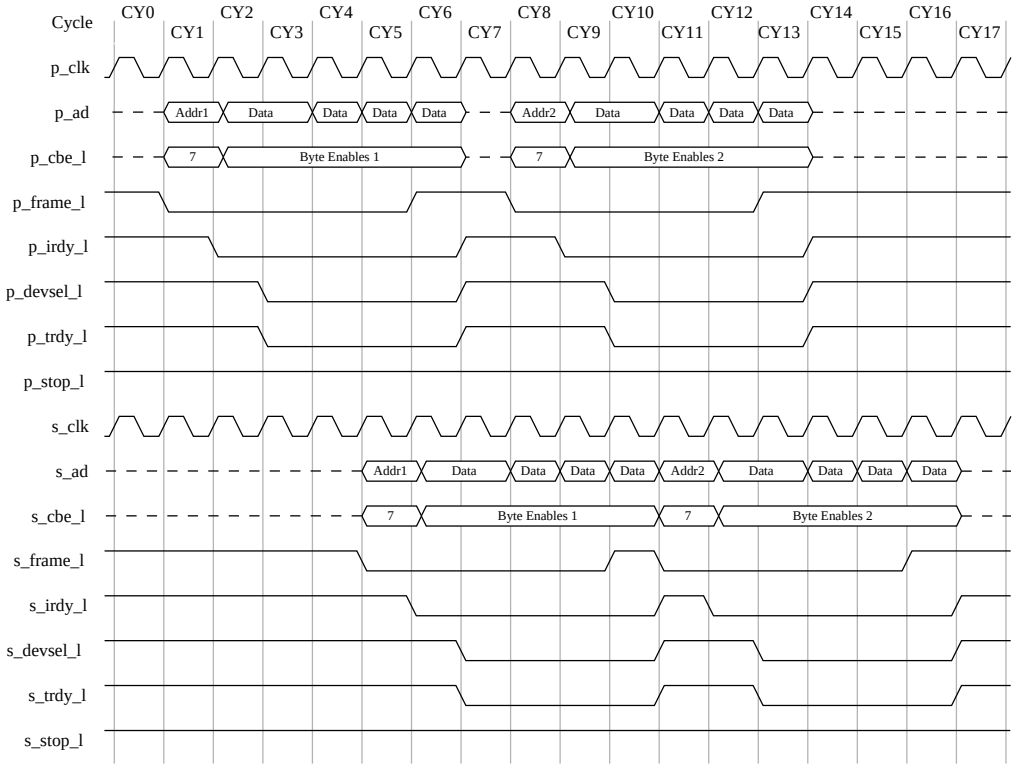
4.5.6 Fast Back-to-Back Write Transactions

The 21152 can recognize and post fast back-to-back write transactions. When the 21152 cannot accept the second transaction because of buffer space limitations, it returns a target retry to the initiator.

When the 21152 has posted multiple write transactions, it can initiate fast back-to-back write transactions if the fast back-to-back enable bit is set in the command register for upstream write transactions, and in the bridge control register for downstream write transactions. The 21152 does not perform write combining or merging.

Figure 8 shows how multiple memory write transactions can be posted and then initiated as fast back-to-back transactions on the target bus.

Figure 8. Fast Back-to-Back Transactions on the Target Bus



LJ-04845.AI4

4.6 Read Transactions

Delayed read forwarding is used for all read transactions crossing the 21152.

Delayed read transactions are treated as either prefetchable or nonprefetchable.

Table 18 shows the read behavior, prefetchable or nonprefetchable, for each type of read operation.

Table 18. Read Transaction Prefetching

Type of Transaction	Read Behavior
I/O read	Prefetching never done
Configuration read	Prefetching never done
Memory read	Downstream: Prefetching used if address in prefetchable space Upstream: Prefetching used if prefetch disable is off (default)
Memory read line	Prefetching always used
Memory read multiple	Prefetching always used

Refer to Section 5.3 for detailed information about prefetchable and nonprefetchable address spaces.

4.6.1 Prefetchable Read Transactions

A prefetchable read transaction is a read transaction where the 21152 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 nonprefetchable read transaction. For prefetchable read transactions, the 21152 forces all byte enable bits to be turned on for all data phases.

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

The amount of data that is prefetched depends on the type of transaction. The amount of prefetching may also be affected by the amount of free buffer space available in the 21152, 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, FIFOs, and so on. The target device's base address register or registers indicate if a memory address region is prefetchable.

4.6.2 Nonprefetchable Read Transactions

A nonprefetchable read transaction is a read transaction where the 21152 requests 1—and only 1—Dword from the target and disconnects the initiator after delivery of the first Dword of read data. Unlike prefetchable read transactions, the 21152 forwards the read byte enable information for the data phase.

Nonprefetchable behavior is used for I/O and configuration read transactions, as well as for memory read transactions that fall into nonprefetchable memory space.

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

4.6.3 Read Prefetch Address Boundaries

The 21152 imposes internal read address boundaries on read prefetching. When a read transaction reaches one of these aligned address boundaries, the 21152 stops prefetching data, unless the target signals a target disconnect before the read prefetch boundary is reached. When the 21152 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 prefetched 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 deasserts FRAME#. Section 4.6.6 describes flow-through mode during read operations.

Table 19 shows the read prefetch address boundaries for read transactions during non-flow-through mode.

Table 19. Read Prefetch Address Boundaries

Type of Transaction	Address Space	Cache Line Size	Prefetch Aligned Address Boundary
Configuration read	—	—	1 Dword (no prefetch)
I/O read	—	—	1 Dword (no prefetch)
Memory read	Nonprefetchable	—	1 Dword (no prefetch)
Memory read	Prefetchable	CLS ≠ 1, 2, 4, 8	16-Dword aligned address boundary
Memory read	Prefetchable	CLS = 1, 2, 4, 8	Cache line address boundary
Memory read line	—	CLS ≠ 1, 2, 4, 8	16-Dword aligned address boundary
Memory read line	—	CLS = 1, 2, 4, 8	Cache line boundary
Memory read multiple	—	CLS ≠ 1, 2, 4, 8	Queue full
Memory read multiple	—	CLS = 1, 2, 4, 8	Second cache line boundary

4.6.4 Delayed Read Requests

The 21152 treats all read transactions as delayed read transactions, which means that the read request from the initiator is posted into a delayed transaction queue. Read data from the target is placed in the read data queue directed toward the initiator bus interface and is transferred to the initiator when the initiator repeats the read transaction.

When the 21152 accepts a delayed read request, it first samples the read address, read bus command, and address parity. When IRDY# is asserted, the 21152 then samples the byte enable bits for the first data phase. This information is entered into the delayed transaction queue. The 21152 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 other than a target retry (target abort, or master abort) is received.

4.6.5 Delayed Read Completion with Target

When the delayed read request reaches the head of the delayed transaction queue, and all previously queued posted write transactions have been delivered, the 21152 arbitrates for the target bus and initiates the read transaction, using the exact read address and read command captured from the initiator during the initial delayed read request. If the read transaction is a nonprefetchable read, the 21152 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 0 for all data phases. If the 21152 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, the 21152 does not initiate any further attempts to read more data.

If the 21152 is unable to obtain read data from the target after 2^{24} attempts, the 21152 ceases further read attempts and returns a target abort to the initiator. The delayed transaction is removed from the delayed transaction queue. The 21152 also asserts **p_serr_l** if the primary SERR# enable bit is set in the command register. Refer to Section 7.4 for information on the assertion of **p_serr_l**.

Once the 21152 receives DEVSEL# and TRDY# from the target, it transfers the data read to the opposite direction read data queue, pointing toward the opposite interface, 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 21152 can accept 1 Dword of read data each PCI clock cycle; that is, no master wait states are inserted. The number of Dwords transferred during a delayed read transaction depends on the conditions given in Table 19 (assuming no disconnect is received from the target).

4.6.6 Delayed Read Completion on Initiator Bus

When the transaction has been completed on the target bus, and the delayed read data is at the head of the read data queue, and all ordering constraints with posted write transactions have been satisfied, the 21152 transfers the data to the initiator when the initiator repeats the transaction. For memory read transactions, the 21152 aliases the memory read, memory read line, and memory read multiple bus commands when matching the bus command of the transaction to the bus command in the delayed transaction queue. The 21152 returns a target disconnect along with the transfer of the last Dword of read data to the initiator. If the initiator terminates the transaction before all read data has been transferred, the remaining read data left in data buffers is discarded.

Figure 9 shows a nonprefetchable delayed read transaction.

Figure 9. Nonprefetchable Delayed Read Transaction

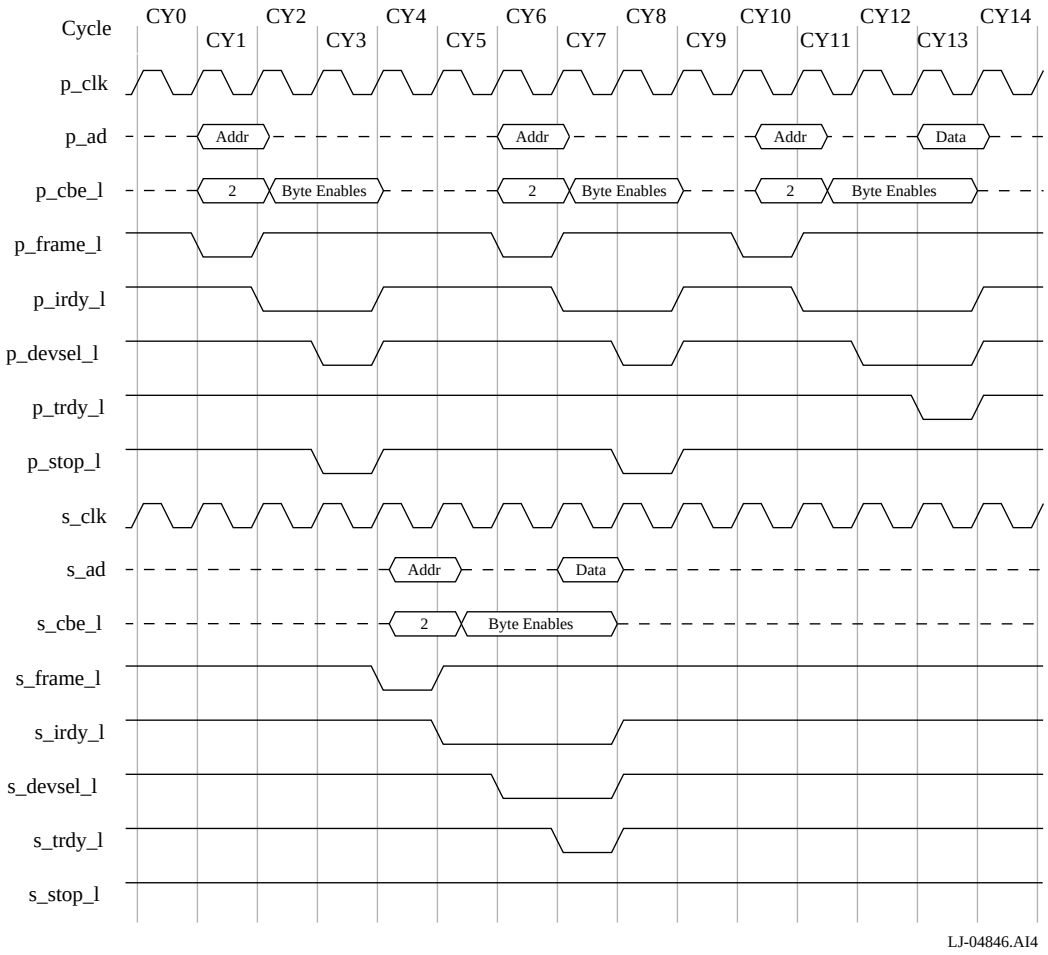
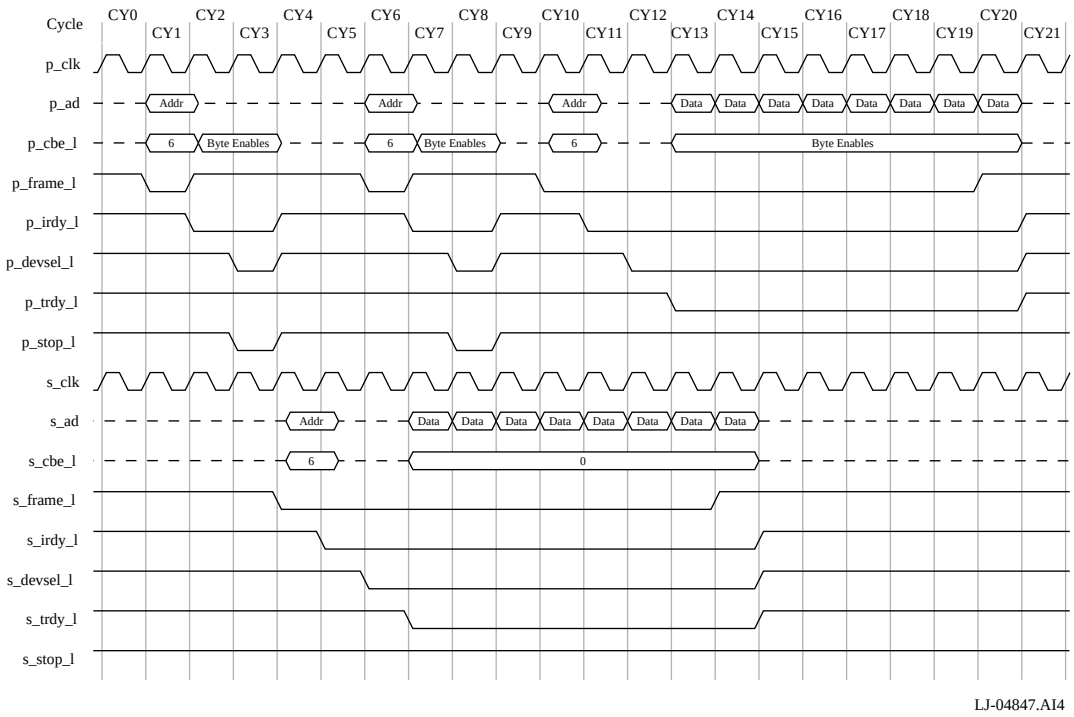


Figure 10 shows a prefetchable delayed read transaction.

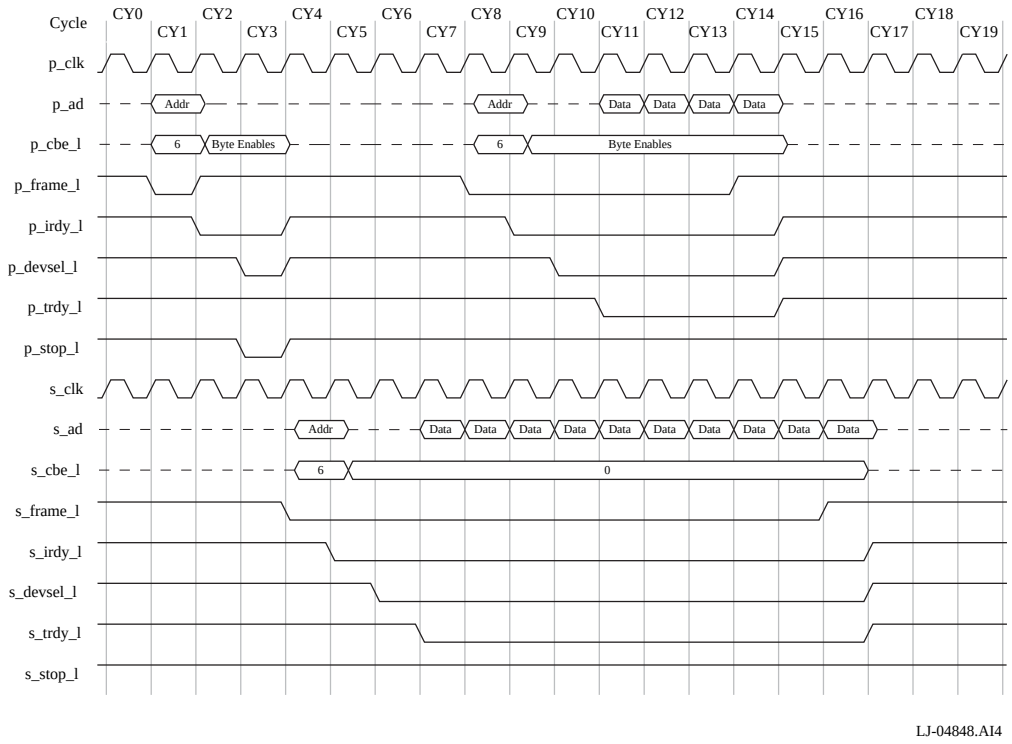
Figure 10. Prefetchable Delayed Read Transaction



When the master repeats the transaction and starts transferring prefetchable read data from 21152 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 4 KB address boundary is reached, or until the buffer fills, whichever comes first. When the buffer empties, the 21152 reflects the stalled condition to the initiator by deasserting TRDY# until more read data is available; otherwise, the 21152 does not insert any target wait states. When the initiator terminates the transaction, the deassertion of FRAME# on the initiator bus is forwarded to the target bus. Any remaining read data is discarded.

Figure 11 shows a flow-through prefetchable read transaction.

Figure 11. Flow-Through Prefetchable Read Transaction



The 21152 implements a discard 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 can be set to one of two values, selectable through both the primary and secondary master time-out value bits in the bridge control register. If the initiator does not repeat the read transaction before the discard timer expires, the 21152 discards the read transaction and the read data from its queues. The 21152 also conditionally asserts **p_serr_l** (refer to Section 7.4).

The 21152 has the capability to post multiple delayed read requests, up to a maximum of three 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.

Refer to Chapter 6.0 for a discussion of how delayed read transactions are ordered when crossing the 21152.

4.7 Configuration Transactions

Configuration transactions are used to initialize a PCI system. Every PCI device has a configuration space that is accessed by configuration commands. All 21152 registers are accessible in configuration space only.

In addition to accepting configuration transactions for initialization of its own configuration space, the 21152 also forwards configuration transactions for device initialization in hierarchical PCI systems, as well as for special cycle generation.

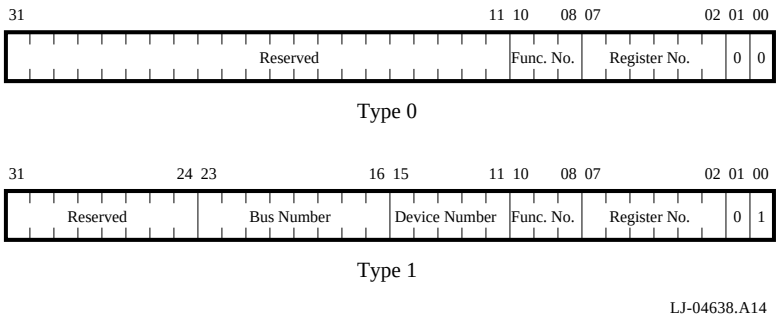
To support hierarchical PCI bus systems, two types of configuration transactions are specified: Type 0 and Type 1.

Type 0 configuration transactions are issued when the intended target resides on the same PCI bus as the initiator. A Type 0 configuration transaction is identified by the configuration command and the lowest 2 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 2 address bits set to 01b.

Figure 12 shows the address formats for Type 0 and Type 1 configuration transactions.

Figure 12. Configuration Transaction Address Formats



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

4.7.1 Type 0 Access to the 21152

The 21152 configuration space is accessed by a Type 0 configuration transaction on the primary interface. The 21152 configuration space cannot be accessed from the secondary bus. The 21152 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.
- Low 2 address bits **p_ad<1:0>** must be 00b.
- Signal **p_idsel** must be asserted.

The function code is ignored because the 21152 is a single-function device.

The 21152 limits all configuration accesses to a single Dword data transfer and returns a target disconnect with the first data transfer if additional data phases are requested. Because read transactions to 21152 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 21152 data buffers; that is, these transactions are completed immediately, regardless of the state of the data buffers.

The 21152 ignores all Type 0 transactions initiated on the secondary interface.

4.7.2 Type 1 to Type 0 Translation

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

The 21152 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. The 21152 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, the 21152 generates a Type 0 transaction only on the secondary bus, and never on the primary bus.

The 21152 responds to a Type 1 configuration transaction and translates it into a Type 0 transaction on the secondary bus when the following conditions are met during the address phase:

- The low 2 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 21152 configuration space.
- The bus command on **p_cbe_l<3:0>** is a configuration read or configuration write transaction.

When the 21152 translates the Type 1 transaction to a Type 0 transaction on the secondary interface, it performs the following translations to the address:

- Sets the low 2 address bits on **s_ad<10>** to 00b
- Decodes the device number and drives the bit pattern specified in Table 20 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

The 21152 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 20 presents the mapping that the 21152 uses.

Table 20. Device Number to IDSEL s_ad Pin Mapping

Device Number	p_ad<15:11>	Secondary IDSEL s_ad<31:16>	s_ad Bit
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)	—

The 21152 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.

The 21152 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.

4.7.3 Type 1 to Type 1 Forwarding

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

When the 21152 detects a Type 1 configuration transaction intended for a PCI bus downstream from the secondary bus, the 21152 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 low 2 address bits are equal to 01b.
- The bus number falls in the range defined by the lower limit (exclusive) in the secondary bus number register and the upper limit (inclusive) in the subordinate bus number register.
- The bus command is a configuration read or write transaction.

The 21152 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 low 2 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 21152 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.

4.7.4 Special Cycles

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

The 21152 initiates a special cycle on the target bus when a Type 1 configuration write transaction is detected on the initiating bus and the following conditions are met during the address phase:

- The low 2 address bits on AD<1:0> are equal to 01b.
- The device number in address bits AD<15:11> is equal to 11111b.
- The function number in address bits AD<10:8> is equal to 111b.
- The register number in address bits AD<7:2> is equal to 000000b.
- The bus number is equal to the value in the secondary bus number register in configuration space for downstream forwarding or equal to the value in the primary bus number register in configuration space for upstream forwarding.
- The bus command on C/BE# is a configuration write command.

When the 21152 initiates the transaction on the target interface, the bus command is changed from configuration write to special cycle. The address and data are forwarded unchanged. Devices that use special cycles ignore the address and decode only the bus command. The data phase contains the special cycle message. The transaction is forwarded as a 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, the 21152 responds with TRDY# to the next attempt of the configuration transaction from the initiator. If more than one data transfer is requested, the 21152 responds with a target disconnect operation during the first data phase.

4.8 Transaction Termination

This section describes how the 21152 returns transaction termination conditions back to the initiator.

- Normal termination

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

- Master abort

  A master abort occurs when no target response is detected. When the initiator does not detect a DEVSEL# from the target within five clock cycles after asserting FRAME#, the initiator terminates the transaction with a master abort. If FRAME# is still asserted, the initiator deasserts FRAME# on the next cycle, and then deasserts IRDY# on the following cycle. IRDY# must be asserted in the same cycle in which FRAME# deasserts. If FRAME# is already deasserted, IRDY# can be deasserted on the next clock cycle following detection of the master abort condition.

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

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

4.8.1 Master Termination Initiated by the 21152

The 21152, as an initiator, uses normal termination if DEVSEL# is returned by the target within five clock cycles of the 21152's assertion of FRAME# on the target bus. As an initiator, the 21152 terminates a transaction when the following conditions are met:

- During a delayed write transaction, a single Dword is delivered.
- During a nonprefetchable read transaction, a single Dword is transferred from the target.
- During a prefetchable read transaction, a prefetch boundary is reached.
- For a posted write transaction, all write data for the transaction is transferred from 21152 data buffers to the target.
- For a burst transfer, with the exception of memory write and invalidate transactions, the master latency timer expires and the 21152's bus grant is deasserted.
- The target terminates the transaction with a retry, disconnect, or target abort.

If the 21152 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 the 21152 is prefetching read data when it terminates the transaction because the master latency timer expires, it does not repeat the transaction to obtain more data.

4.8.2 Master Abort Received by the 21152

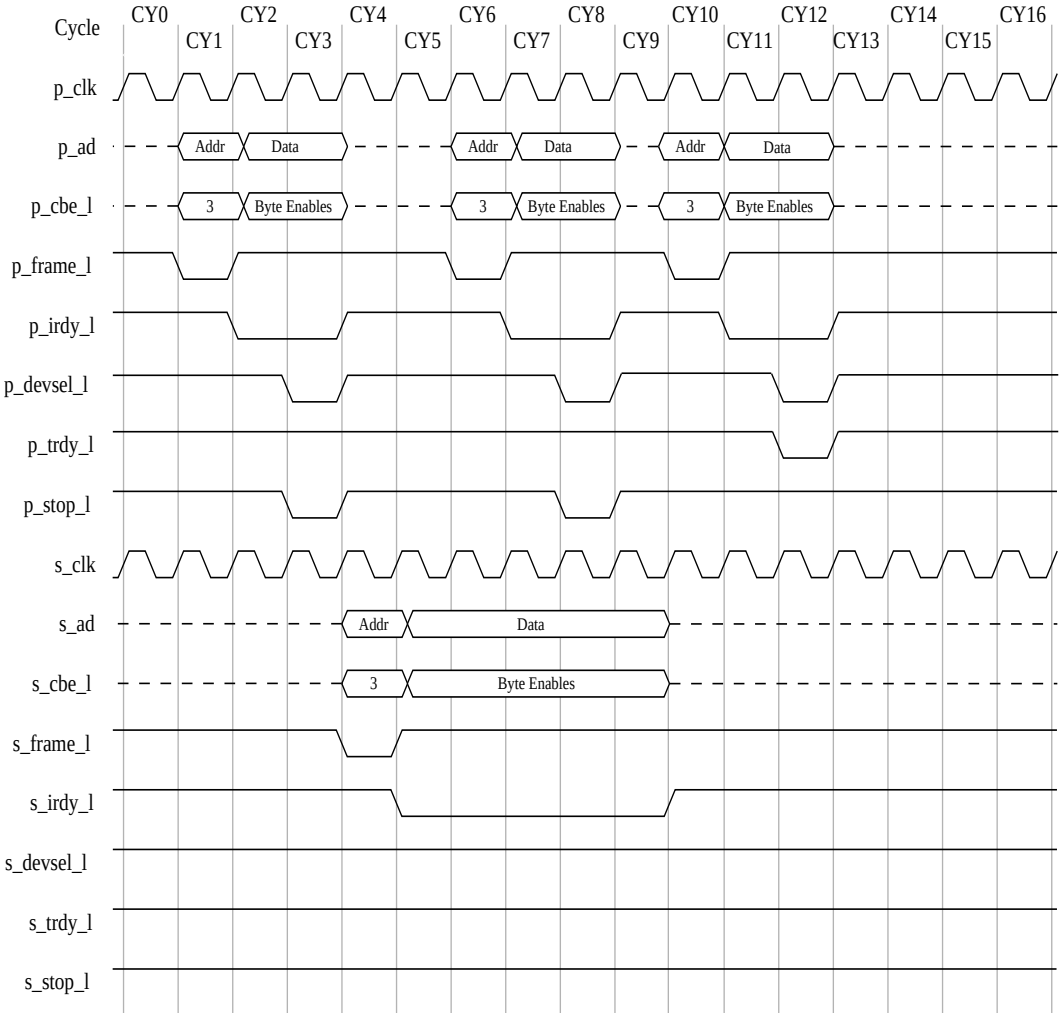
If the 21152 initiates a transaction on the target bus and does not detect DEVSEL# returned by the target within five clock cycles of the 21152's assertion of FRAME#, the 21152 terminates the transaction with a master abort. The 21152 sets the received master abort bit in the status register corresponding to the target bus.

For delayed read and write transactions, when the master abort mode bit in the bridge control register is 0, the 21152 returns TRDY# on the initiator bus and, for read transactions, returns FFFF FFFFh as data.

When the master abort mode bit is 1, the 21152 returns target abort on the initiator bus. The 21152 also sets the signaled target abort bit in the register corresponding to the initiator bus.

Figure 13 shows a delayed write transaction that is terminated with a master abort.

Figure 13. Delayed Write Transaction Terminated with Master Abort



LJ-04849.AI4

When a master abort is received in response to a posted write transaction, the 21152 discards the posted write data and makes no more attempts to deliver the data. The 21152 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 a master abort is detected in response to a posted write transaction and the master abort mode bit is set, the 21152 also asserts **p_serr_l** if enabled by the SERR# enable bit in the command register and if not disabled by the device-specific **p_serr_l** disable bit for master abort during posted write transactions [that is, master abort mode = 1; SERR# enable bit = 1; and **p_serr_l** disable bit for master aborts = 0].

Note: When the 21152 performs a Type 1 to special cycle translation, a master abort is the expected termination for the special cycle on the target bus. In this case, the master abort received bit is *not* set, and the Type 1 configuration transaction is disconnected after the first data phase.

4.8.3 Target Termination Received by the 21152

When the 21152 initiates a transaction on the target bus and the target responds with DEVSEL#, the target can end the transaction with one of the following types of termination:

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

The 21152 handles these terminations in different ways, depending on the type of transaction being performed.

4.8.3.1 Delayed Write Target Termination Response

When the 21152 initiates a delayed write transaction, the type of target termination received from the target can be passed back to the initiator.

Table 21 shows the 21152 response to each type of target termination that occurs during a delayed write transaction.

Table 21. 21152 Response to Delayed Write Target Termination

Target Termination	21152 Response
Normal	Return disconnect to initiator with first data transfer only if multiple data phases requested.
Target retry	Return target retry to initiator. Continue write attempts to target.
Target disconnect	Return disconnect to initiator with first data transfer only if multiple data phases requested.
Target abort	Return target abort to initiator. Set received target abort bit in target interface status register. Set signaled target abort bit in initiator interface status register.

The 21152 repeats a delayed write transaction until one of the following conditions is met:

- The 21152 completes at least one data transfer.
- The 21152 receives a master abort.
- The 21152 receives a target abort.
- The 21152 makes 2^{24} write attempts resulting in a response of target retry.

After the 21152 makes 2^{24} attempts of the same delayed write transaction on the target bus, the 21152 asserts **p_serr_l** if the primary SERR# enable bit is set in the command register and the implementation-specific **p_serr_l** disable bit for this condition is not set in the **p_serr_l** event disable register. The 21152 stops initiating transactions in response to that delayed write transaction. The delayed write request is discarded. Upon a subsequent write transaction attempt by the initiator, the 21152 returns a target abort. Refer to Section 7.4 for a description of system error conditions.

4.8.3.2 Posted Write Target Termination Response

When the 21152 initiates a posted write transaction, the target termination cannot be passed back to the initiator.

Table 22 shows the 21152 response to each type of target termination that occurs during a posted write transaction.

Table 22. 21152 Response to Posted Write Target Termination

Target Termination	21152 Response
Normal	No additional action.
Target retry	Repeat write transaction to target.
Target disconnect	Initiate write transaction to deliver remaining posted write data.
Target abort	Set received target abort bit in the target interface status register. Assert **p_serr_l** if enabled, and set the signaled 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, the 21152 initiates another write transaction to attempt to deliver the rest of the write data. In the case of a target retry, the exact same address will be driven as for the initial write transaction attempt. If a target disconnect is received, the address that is driven on a subsequent write transaction attempt is updated to reflect the address of the current Dword. If the initial write transaction is a memory write and invalidate transaction, and a partial delivery of write data to the target is performed before a target disconnect is received, the 21152 uses the memory write command to deliver the rest of the write data because less than a cache line will be transferred in the subsequent write transaction attempt.

After the 21152 makes 2^{24} write transaction attempts and fails to deliver all the posted write data associated with that transaction, the 21152 asserts **p_serr_l** if the primary SERR# enable bit is set in the command register and the device-specific **p_serr_l** disable bit for this condition is not set in the **p_serr_l** event disable register. The write data is discarded. Refer to Section 7.4 for a discussion of system error conditions.

4.8.3.3 Delayed Read Target Termination Response

When the 21152 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.

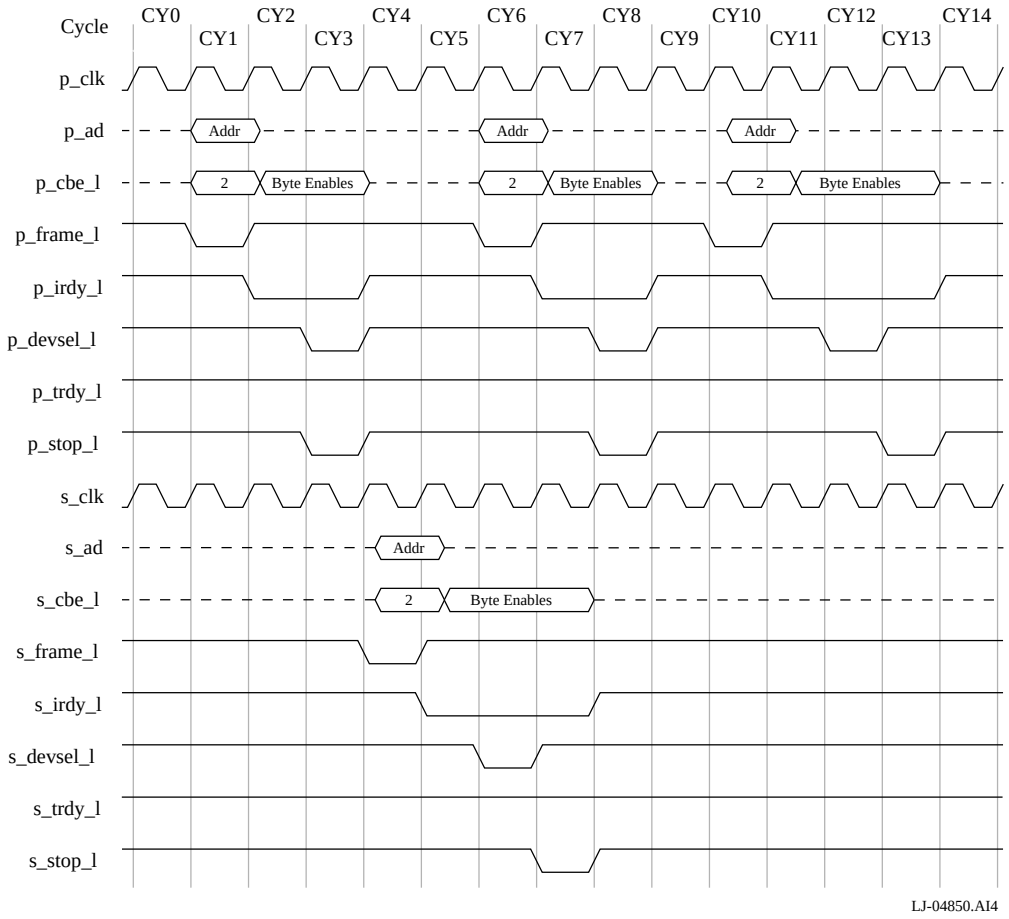
Table 23 shows the 21152 response to each type of target termination that occurs during a delayed read transaction.

Table 23. 21152 Response to Delayed Read Target Termination

Target Termination	21152 Response
Normal	If prefetchable, target disconnect only if initiator requests more data than read from target. If nonprefetchable, target disconnect on first data phase.
Target retry	Reinitiate 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.

Figure 14 shows a delayed read transaction that is terminated with a target abort.

Figure 14. Delayed Read Transaction Terminated with Target Abort



The 21152 repeats a delayed read transaction until one of the following conditions is met:

- The 21152 completes at least one data transfer.
- The 21152 receives a master abort.
- The 21152 receives a target abort.
- The 21152 makes 2^{24} read attempts resulting in a response of target retry.

After the 21152 makes 2^{24} attempts of the same delayed read transaction on the target bus, the 21152 asserts **p_serr_l** if the primary SERR# enable bit is set in the command register and the implementation-specific **p_serr_l** disable bit for this condition is not set in the **p_serr_l** event disable register. The 21152 stops initiating transactions in response to that delayed read transaction. The delayed read request is discarded. Upon a subsequent read transaction attempt by the initiator, the 21152 returns a target abort. Refer to Section 7.4 for a description of system error conditions.

4.8.4 Target Termination Initiated by the 21152

The 21152 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.

4.8.4.1 Target Retry

The 21152 returns a target retry to the initiator when it cannot accept write data or return read data as a result of internal conditions. The 21152 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.
  - The transaction has already been entered into the 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 the 21152, and the write transaction is not a locked 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 the target, but it is not yet at the head of the read data queue, or a posted write transaction precedes it.
  - The delayed transaction queue is full, and the transaction cannot be queued.
  - A delayed read request with the same address and bus command has already been queued.
  - A locked sequence is being propagated across the 21152, and the read transaction is not a locked transaction.
  - The 21152 is currently discarding previously prefetched read data.
- For posted write transactions:
  - The posted write data buffer does not have enough space for address and at least 1 Dword of write data.
  - A locked sequence is being propagated across the 21152, 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 this is a write transaction, within the time frame specified by the master time-out value; otherwise, the transaction is discarded from the 21152 buffers.

4.8.4.2 Target Disconnect

The 21152 returns a target disconnect to an initiator when one of the following conditions is met:

- The 21152 hits an internal address boundary.
- The 21152 cannot accept any more write data.
- The 21152 has no more read data to deliver.

Refer to Section 4.5.4 for a description of write address boundaries, and Section 4.6.3 for a description of read address boundaries.

4.8.4.3 Target Abort

The 21152 returns a target abort to an initiator when one of the following conditions is met:

- The 21152 is returning a target abort from the intended target.
- The 21152 is unable to obtain delayed read data from the target or to deliver delayed write data to the target after 2^{24} attempts.

When the 21152 returns a target abort to the initiator, it sets the signaled target abort bit in the status register corresponding to the initiator interface.

§ §

5.0 Address Decoding

The 21152 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 21152 configuration space.

This chapter describes these address ranges, as well as ISA-mode and VGA-addressing support.

5.1 Address Ranges

The 21152 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:

- One 32-bit I/O address range
- One 32-bit memory-mapped I/O (nonprefetchable memory)
- One 64-bit prefetchable memory address range

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.

The 21152 uses a flat address space; that is, it does not perform any address translations. The address space has no “gaps” — addresses that are not marked for downstream forwarding are always forwarded upstream.

5.2 I/O Address Decoding

The 21152 uses the following mechanisms that are defined in the 21152 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 5.4 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 21152 configuration space. If the I/O enable bit is not set, all I/O transactions initiated on the primary bus are ignored. 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, the 21152 ignores all I/O and memory transactions initiated on the secondary bus. Setting the master enable bit also allows upstream forwarding of memory transactions.

Caution: If any 21152 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, the 21152 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 the I/O enable and master enable bits, and change them subsequently only when the primary and secondary PCI buses are idle.

5.2.1 I/O Base and Limit Address Registers

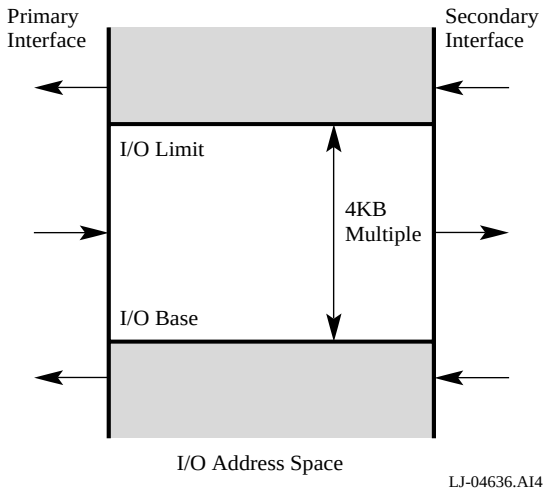
The 21152 implements one set of I/O base and limit address registers in configuration space that define an I/O address range for downstream forwarding. The 21152 supports 32-bit I/O addressing, which allows I/O addresses downstream of the 21152 to be mapped anywhere in a 4 GB I/O address space.

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 transactions are forwarded upstream, and no I/O transactions are forwarded downstream.

Figure 15 illustrates transaction forwarding within and outside the I/O address range.

Figure 15. I/O Transaction Forwarding Using Base and Limit Addresses



The 21152 I/O range has a minimum granularity of 4 KB and is aligned on a 4 KB boundary. The maximum I/O range is 4 GB 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 the 21152 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 4 KB 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 4 KB 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 4 KB 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.

5.2.2 ISA Mode

The 21152 supports ISA mode by providing an ISA enable bit in the bridge control register in configuration space. ISA mode modifies the response of the 21152 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 the 21152 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 64 KB of I/O space (address bits <31:16> are 0000h).

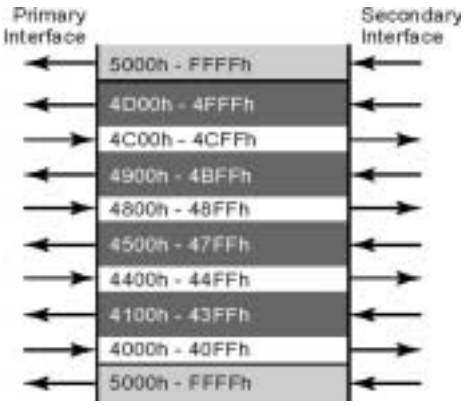
When the ISA enable bit is set, the 21152 does not forward downstream any I/O transactions addressing the top 768 bytes of each aligned 1 KB block. Only those transactions addressing the bottom 256 bytes of an aligned 1 KB block inside the base and limit I/O address range are forwarded downstream. Transactions above the 64 KB 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, the 21152 forwards upstream those I/O transactions addressing the top 768 bytes of each aligned 1 KB block within the first 64 KB 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 the 21152 can have I/O space mapped into the first 256 bytes of each 1 KB chunk below the 64 KB boundary, or anywhere in I/O space above the 64 KB boundary.

Figure 16 illustrates I/O forwarding when the ISA enable bit is set

Figure 16. I/O Transaction Forwarding in ISA Mode



Note:
In this example:
I/O Base Address = 0000 4000h
I/O Limit Address = 0000 4 FFFh
ISA Enable = 1

5.3 Memory Address Decoding

The 21152 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 5.4.1 describes VGA mode.

To enable downstream forwarding of memory transactions, the memory enable bit must be set in the command register in 21152 configuration space. To enable upstream forwarding of memory transactions, the master enable bit must be set in the command register. Setting the master enable bit also allows upstream forwarding of I/O transactions.

Caution: If any 21152 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, 21152 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.

5.3.1 Memory-Mapped I/O Base and Limit Address Registers

Memory-mapped I/O is also referred to as nonprefetchable memory. Memory addresses that cannot automatically be prefetched but that can conditionally prefetch based on command type should be mapped into this space. Read transactions to nonprefetchable space may exhibit side effects; this space may have non-memory-like behavior. The 21152 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 the 21152 uses to determine when to forward memory commands. The 21152 forwards a memory transaction from the primary to the secondary interface if the transaction address falls within the memory-mapped I/O address range. The 21152 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 1 MB. The maximum memory-mapped I/O address range is 4 GB.

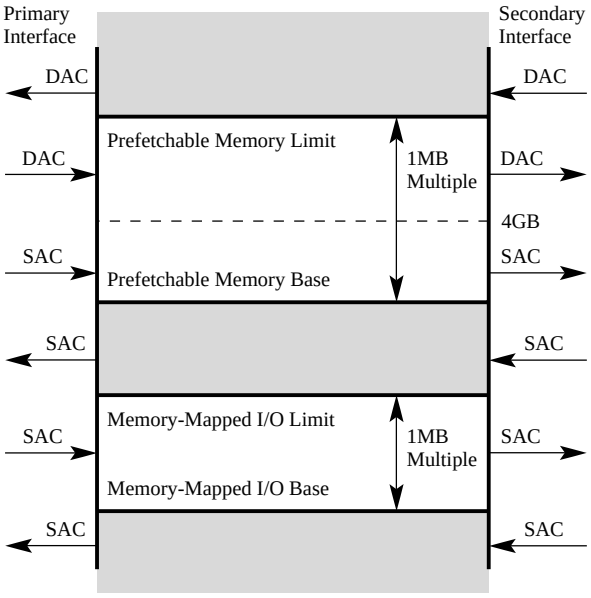
The memory-mapped I/O address range is defined by a 16-bit memory-mapped I/O base address register at configuration offset 20h and by a 16-bit memory-mapped I/O limit address register at offset 22h. The top 12 bits of each of these registers correspond to bits <31:20> of the memory address. The low 4 bits are hardwired to 0. The low 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 low 20 bits of the memory-mapped I/O limit address are assumed to be F FFFFh, 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.

Figure 17 shows how transactions are forwarded using both the memory-mapped I/O range and the prefetchable memory range.

Figure 17. Memory Transaction Forwarding Using Base and Limit Registers



Note:
DAC – Dual Address Cycle
SAC – Single Address Cycle

LJ-04639.AI4

5.3.2 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. The 21152 prefetches 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 the 21152 uses to determine when to forward memory commands. The 21152 forwards a memory transaction from the primary to the secondary interface if the transaction address falls within the prefetchable memory address range. The 21152 ignores memory transactions initiated on the secondary interface that fall into this address range. The 21152 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 in order to pass any single address cycle transactions downstream. Section 5.3.3 further describes 64-bit addressing support.

The prefetchable memory address range has a granularity and alignment of 1 MB. The maximum memory address range is 4 GB when 32-bit addressing is used, and 2^{24} bytes when 64-bit addressing is used.

The 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 28h. 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 1h, indicating 64-bit address support. The low 20 bits of the prefetchable memory base address are assumed to be 0 0000h, which results in a natural alignment to a 1 MB boundary. The low 20 bits of the prefetchable memory limit address are assumed to be F FFFFh, which results in an alignment to the top of a 1 MB block.

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 1 MB block of memory. Write these registers with their appropriate values before setting either the memory enable bit or the master enable bit in the command register in configuration space.

To turn off the prefetchable memory address range, write the 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.

5.3.3 Prefetchable Memory 64-Bit Addressing Registers

The 21152 supports 64-bit memory address decoding for forwarding of dual-address memory transactions. The dual-address cycle is used to support 64-bit addressing. The first address phase of a dual-address transaction contains the low 32 address bits, and the second address phase contains the high 32 address bits. During a dual-address cycle transaction, the upper 32 bits must never be 0—use the single address cycle commands for transactions addressing the first 4GB of memory space.

The 21152 implements the prefetchable memory base address upper 32 bits register and the prefetchable memory limit address upper 32 bits register to define a prefetchable memory address range greater than 4 GB. The prefetchable address space can then be defined in three different ways:

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

If the prefetchable memory space on the secondary interface resides entirely in the first 4GB of memory, both upper 32 bits registers must be set to 0. The 21152 ignores all dual-address cycle transactions initiated on the primary interface and forwards all dual-address transactions initiated on the secondary interface upstream.

If the secondary interface prefetchable memory space resides entirely above the first 4 GB of memory, both the prefetchable memory base address upper 32 bits register and the prefetchable memory limit address upper 32 bits register must be initialized to nonzero values. The 21152 ignores all single address memory transactions initiated on the primary interface and forwards all single address memory transactions initiated on the secondary interface upstream (unless they fall within the memory-mapped I/O or VGA memory range). A dual-address memory transaction is forwarded downstream from the primary interface if it falls within the address range defined by the prefetchable memory base address, prefetchable memory base address upper 32 bits, prefetchable memory limit address, and prefetchable memory limit address upper 32 bits registers. If the dual-address transaction initiated on the secondary interface falls outside this address range, it is forwarded upstream to the primary interface. The 21152 does not respond to a dual-address transaction initiated on the primary interface that falls outside this address range, or to a dual-address transaction initiated on the secondary interface that falls within the address range.

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

The prefetchable memory base address upper 32 bits register is located at configuration Dword offset 28h, and the prefetchable memory limit address upper 32 bits register is located at configuration Dword offset 2Ch. Both registers are reset to 0. Refer to Figure 17 for an illustration of how transactions are forwarded using both the memory-mapped I/O range and the prefetchable memory range.

5.4 VGA Support

The 21152 provides two modes for VGA support:

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

5.4.1 VGA Mode

When a VGA-compatible device exists downstream from the 21152, set the VGA mode bit in the bridge control register in configuration space to enable VGA mode. When the 21152 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 21152 base and limit address registers. The 21152 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 nonprefetchable. The 21152 requests only a single data transfer from the target, and read byte enable bits are forwarded to the target bus.

The VGA I/O addresses consist of the following I/O addresses:

- 3B0h–3BBh
- 3C0h–3DFh

These I/O addresses are aliased every 1 KB throughout the first 64 KB of I/O space. This means that address bits <15:10> are not decoded and can be any value, while address bits <31:16> must be all 0s.

VGA BIOS addresses starting at C0000h are not decoded in VGA mode.

5.4.2 VGA Snoop Mode

The 21152 provides VGA snoop mode, allowing for VGA palette write transactions to be forwarded downstream. This mode is used when a graphics device downstream from the 21152 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 the 21152 claims VGA palette write transactions by asserting DEVSEL# in VGA snoop mode.

When the VGA snoop bit is set, the 21152 forwards downstream transactions with the following I/O addresses:

- 3C6h
- 3C8h
- 3C9h

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 aliased every 1 KB throughout the first 64 KB of I/O space.

Note: If both the VGA mode bit and the VGA snoop bit are set, the 21152 behaves in the same way as if only the VGA mode bit were set.

§ §

6.0 Transaction Ordering

To maintain data coherency and consistency, the 21152 complies with the ordering rules set forth in the *PCI Local Bus Specification*, Revision 2.1, for transactions crossing the bridge.

This chapter describes the ordering rules that control transaction forwarding across the 21152. For a more detailed discussion of transaction ordering, refer to Appendix E of the *PCI Local Bus Specification, Revision 2.1*.

6.1 Transactions Governed by Ordering Rules

Ordering relationships are established for the following classes of transactions crossing the 21152:

- 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, also comprised of I/O write and configuration write transactions.

  Delayed write completion transactions have been completed on the target bus, and the target response is queued in the 21152 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, and configuration read transactions.

  Delayed read completion transactions have been completed on the target bus, and the read data has been queued in the 21152 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.

The 21152 does not combine or merge write transactions:

- The 21152 does not combine separate write transactions into a single write transaction—this optimization is best implemented in the originating master.
- The 21152 does not merge bytes on separate masked write transactions to the same Dword address—this optimization is also best implemented in the originating master.
- The 21152 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.

6.2 General Ordering Guidelines

Independent transactions on the primary and secondary buses have a relationship only when those transactions cross the 21152.

The following general ordering guidelines govern transactions crossing the 21152:

- 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 the delayed transaction requests, using some fairness algorithm. Repeating a delayed transaction cannot be contingent on completion of another delayed transaction; otherwise, a deadlock can occur.
- Write transactions flowing in one direction have no ordering requirements with respect to write transactions flowing in the other direction. The 21152 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 nonlocked, nonposted transaction as a master. This is true of the 21152 and must also be true of other bus agents; otherwise, a deadlock can occur.
- The 21152 accepts posted write transactions, regardless of the state of completion of any delayed transactions being forwarded across the 21152.

6.3 Ordering Rules

Table 24 shows the ordering relationships of all the transactions and refers by number to the ordering rules that follow.

Table 24. 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 21152'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 24. These ordering rules apply to posted write transactions, delayed write and read requests, and delayed write and read completion transactions crossing the 21152 in the same direction. Note that delayed completion transactions cross the 21152 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 the 21152 as the target of the write transaction. The posted write transaction must complete to the target before the read data is returned to the initiator.

   The read transaction can be to a status register of the initiator of the posted write data and therefore should not complete until the write transaction is complete.

4. Delayed write requests cannot pass previously queued posted write data.

   As in the case of posted memory write transactions, the delayed write transaction can be setting 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 bridges that support delayed transactions are used in the same system with bridges that do not support delayed transactions. A fairness algorithm is used to arbitrate between the posted write queue and the delayed transaction queue.

6.4 Data Synchronization

Data synchronization refers to the relationship between interrupt signaling and data delivery. The *PCI Local Bus Specification,* Revision 2.1, 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.

The 21152 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.

§ §

7.0 Error Handling

The 21152 checks, forwards, and generates parity on both the primary and secondary interfaces. To maintain transparency, the 21152 always tries to forward the existing parity condition on one bus to the other bus, along with address and data. The 21152 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, the 21152 implements the following:

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

This chapter provides detailed information about how the 21152 handles errors. It also describes error status reporting and error operation disabling.

7.1 Address Parity Errors

The 21152 checks address parity for all transactions on both buses (all address and all bus commands).

When the 21152 detects an address parity error on the primary interface, the following occurs:

- If the parity error response bit is set in the command register, the 21152 does not claim the transaction with **p_devsel_l**; this may allow the transaction to terminate in a master abort.

  If the parity error response bit is not set, the 21152 proceeds normally and accepts the transaction if it is directed to or across the 21152.
- The 21152 sets the detected parity error bit in the status register.
- The 21152 asserts **p_serr_l** and sets the signaled system error bit in the status register, if both of the following conditions are met:
  - — The SERR# enable bit is set in the command register.
  - — The parity error response bit is set in the command register.

When the 21152 detects an address parity error on the secondary interface, the following occurs:

- If the parity error response bit is set in the bridge control register, the 21152 does not claim the transaction with **s_devsel_l**; this may allow the transaction to terminate in a master abort.

  If the parity error response bit is not set, the 21152 proceeds normally and accepts the transaction if it is directed to or across the 21152.
- The 21152 sets the detected parity error bit in the secondary status register.
- The 21152 asserts **p_serr_l** and sets the signaled system error bit in the status register, if both of the following conditions are met:
  - — The SERR# enable bit is set in the command register.
  - — The parity error response bit is set in the bridge control register.

7.2 Data Parity Errors

When forwarding transactions, the 21152 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 the 21152.

7.2.1 Configuration Write Transactions to 21152 Configuration Space

When the 21152 detects a data parity error during a Type 0 configuration write transaction to 21152 configuration space, the following events occur:

- If the parity error response bit is set in the command register, the 21152 asserts **p_trdy_l** and writes the data to the configuration register. The 21152 also asserts **p_perr_l**.
- If the parity error response bit is not set, the 21152 does not assert **p_perr_l**.
- The 21152 sets the detected parity error bit in the status register, regardless of the state of the parity error response bit.

7.2.2 Read Transactions

When the 21152 detects a parity error during a read transaction, the target drives data and data parity, and the initiator checks parity and conditionally asserts PERR#.

For downstream transactions, when the 21152 detects a read data parity error on the secondary bus, the following events occur:

- The 21152 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.
- The 21152 sets the detected parity error bit in the secondary status register.
- The 21152 sets the data parity detected bit in the secondary status register, if the secondary interface parity error response bit is set in the bridge control register.
- The 21152 forwards the bad parity with the data back to the initiator on the primary bus.

  If the data with the bad parity is prefetched 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.
- The 21152 completes the transaction normally.

For upstream transactions, when the 21152 detects a read data parity error on the primary bus, the following events occur:

- The 21152 asserts **p_perr_l** two cycles following the data transfer, if the primary interface parity error response bit is set in the command register.
- The 21152 sets the detected parity error bit in the primary status register.
- The 21152 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.
- The 21152 forwards the bad parity with the data back to the initiator on the secondary bus.

  If the data with the bad parity is prefetched 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.
- The 21152 completes the transaction normally.

The 21152 returns to the initiator the data and parity that was received from the target. When the initiator detects a parity error on this read data and is enabled to report it, the initiator asserts PERR# two cycles after the data transfer occurs. It is assumed that the initiator takes responsibility for handling a parity error condition; therefore, when the 21152 detects PERR# asserted while returning read data to the initiator, the 21152 does not take any further action and completes the transaction normally.

7.2.3 Delayed Write Transactions

When the 21152 detects a data parity error during a delayed write transaction, the initiator drives data and data parity, and the target checks parity and conditionally asserts PERR#.

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

- During the original delayed write request transaction
- When the initiator repeats the delayed write request transaction
- When the 21152 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 the 21152 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, the 21152 asserts TRDY# to the initiator and the transaction is not queued. If multiple data phases are requested, STOP# is also asserted to cause a target disconnect. Two cycles after the data transfer, the 21152 also asserts PERR#.

  If the parity error response bit is not set, the 21152 returns a target retry and queues the transaction as usual. Signal PERR# is not asserted. In this case, the initiator repeats the transaction.
- The 21152 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 reattempts 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 the 21152 is delivering data to the target on the secondary bus and **s_perr_l** is asserted by the target, the following events occur:

- The 21152 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.
- The 21152 captures the parity error condition to forward it back to the initiator on the primary bus.

Similarly, for upstream transactions, when the 21152 is delivering data to the target on the primary bus and **p_perr_l** is asserted by the target, the following events occur:

- The 21152 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.
- The 21152 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 reattempt 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 the 21152 has write status to return, the following events occur:

- The 21152 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.
- The 21152 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 the 21152 has write status to return, the following events occur:

- The 21152 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.
- The 21152 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, in the case 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:

- The 21152 asserts **p_perr_l** two cycles after the data transfer, if both of the following are true:
  - The primary interface parity error response bit is set in the command register.
  - The secondary interface parity error response bit is set in the bridge control register.
- The 21152 completes the transaction normally.

For upstream transactions, in the case 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:

- The 21152 asserts **s_perr_l** two cycles after the data transfer, if both of the following are true:
  - The primary interface parity error response bit is set in the command register.
  - The secondary interface parity error response bit is set in the bridge control register.
- The 21152 completes the transaction normally.

7.2.4 Posted Write Transactions

During downstream posted write transactions, when the 21152, responding as a target, detects a data parity error on the initiator (primary) bus, the following events occur:

- The 21152 asserts **p_perr_l** two cycles after the data transfer, if the primary interface parity error response bit is set in the command register.
- The 21152 sets the primary interface parity error detected bit in the status register.
- The 21152 captures and forwards the bad parity condition to the secondary bus.
- The 21152 completes the transaction normally.

Similarly, during upstream posted write transactions, when the 21152, responding as a target, detects a data parity error on the initiator (secondary) bus, the following events occur:

- The 21152 asserts **s_perr_l** two cycles after the data transfer, if the secondary interface parity error response bit is set in the bridge control register.
- The 21152 sets the secondary interface parity error detected bit in the secondary status register.
- The 21152 captures and forwards the bad parity condition to the primary bus.
- The 21152 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:

- The 21152 sets the data parity detected bit in the secondary status register, if the secondary interface parity error response bit is set in the bridge control register.
- The 21152 asserts **p_serr_l** and sets the signaled system error bit in the status register, if all of the following conditions are met:
  - The SERR# enable bit is set in the command register.
  - The device-specific **p_serr_l** disable bit for posted write parity errors is not set.
  - The secondary interface parity error response bit is set in the bridge control register.
  - The primary interface parity error response bit is set in the command register.
  - The 21152 did not detect the parity error on the primary (initiator) bus; that is, the parity error was not forwarded from the primary 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:

- The 21152 sets the data parity detected bit in the status register, if the primary interface parity error response bit is set in the command register.
- The 21152 asserts **p_serr_l** and sets the signaled system error bit in the status register, if all of the following conditions are met:
  - The SERR# enable bit is set in the command register.
  - The secondary interface parity error response bit is set in the bridge control register.
  - The primary interface parity error response bit is set in the command register.
  - The 21152 did not detect the parity error on the secondary (initiator) bus; that is, the parity error was not forwarded from the secondary bus.

The assertion of **p_serr_l** is used to signal the parity error condition in the case where 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 was forwarded from the initiating bus to the target bus, **p_serr_l** is not asserted.

7.3 Data Parity Error Reporting Summary

In the previous sections, the 21152's responses to data parity errors are presented according to the type of transaction in progress. This section organizes the 21152's responses to data parity errors according to the status bits that the 21152 sets and the signals that it asserts.

Table 25 shows setting the detected parity error bit in the status register, corresponding to the primary interface. This bit is set when the 21152 detects a parity error on the primary interface.

Table 25. 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[1]
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

1. x = don't care

Table 26 shows setting the detected parity error bit in the secondary status register, corresponding to the secondary interface. This bit is set when the 21152 detects a parity error on the secondary interface.

Table 26. Setting the Secondary Interface Detected Parity Error Bit

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

1. x = don’t care

Table 27 shows setting the data parity detected bit in the status register, corresponding to the primary interface. This bit is set under the following conditions:

- The 21152 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 27. Setting the Primary Interface Data Parity Detected Bit

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

1. x = don’t care

Table 28 shows setting the data parity detected bit in the secondary status register, corresponding to the secondary interface. This bit is set under the following conditions:

- The 21152 must be a master on the secondary bus.
- The parity error response bit in the bridge control register, corresponding to the secondary interface, must be set.
- The **s_perr_l** signal is detected asserted or a parity error is detected on the secondary bus.

Table 28. Setting the Secondary Interface Data Parity Detected Bit

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

1. x = don’t care

Table 29 shows assertion of **p_perr_l**. This signal is set under the following conditions:

- The 21152 is either the target of a write transaction or the initiator of a read transaction on the primary bus.
- The parity error response bit in the command register, corresponding to the primary interface, must be set.
- The 21152 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 29. Assertion of p_perr_l

p_perr_l	Transaction Type	Direction	Bus Where Error Was Detected	Primary/Secondary Parity Error Response Bits
1 (deasserted)	Read	Downstream	Primary	x/x[1]
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

1. x = don’t care.
2. The parity error was detected on the target (secondary) bus but not on the initiator (primary) bus.

Table 30 shows assertion of **s_perr_l**. This signal is set under the following conditions:

- The 21152 is either the target of a write transaction or the initiator of a read transaction on the secondary bus.
- The parity error response bit in the bridge control register, corresponding to the secondary interface, must be set.
- The 21152 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 30. Assertion of s_perr_l

s_perr_l	Transaction Type	Direction	Bus Where Error Was Detected	Primary/Secondary Parity Error Response Bits
1 (deasserted)	Read	Downstream	Primary	x/x[1]
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

1. x = don't care.
2. The parity error was detected on the target (primary) bus but not on the initiator (secondary) bus.

Table 31 shows assertion of **p_serr_l**. This signal is set under the following conditions:

- The 21152 has detected **p_perr_l** asserted on an upstream posted write transaction or **s_perr_l** asserted on a downstream posted write transaction.
- The 21152 did not detect the parity error as a target of the posted write transaction.
- The parity error response bit on the command register and the parity error response bit on the bridge control register must both be set.
- The SERR# enable bit must be set in the command register.

Table 31. Assertion of p_serr_l for Data Parity Errors

p_serr_l	Transaction Type	Direction	Bus Where Error Was Detected	Primary/Secondary Parity Error Response Bits
1 (deasserted)	Read	Downstream	Primary	x/x[1]
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

1. 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.

7.4 System Error (SERR#) Reporting

The 21152 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 7.2.3.

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

- For the 21152 to assert **p_serr_l** for any reason, the SERR# enable bit must be set in the command register.
- Whenever the 21152 asserts **p_serr_l**, the 21152 must also set the signaled system error bit in the status register.

In compliance with the *PCI-to-PCI Bridge Architecture Specification*, the 21152 asserts **p_serr_l** when it detects the secondary SERR# input, **s_serr_l**, asserted and the SERR# forward enable bit is set in the bridge control register. In addition, the 21152 also sets the received system error bit in the secondary status register.

The 21152 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} attempts to deliver (2^{24} target retries received)
- Parity error reported on target bus during posted write transaction (refer to previous section)
- Delayed write data discarded after 2^{24} attempts to deliver (2^{24} target retries received)
- Delayed read data cannot be transferred from target after 2^{24} attempts (2^{24} target retries received)
- Master timeout on delayed transaction

The device-specific **p_serr_l** status register reports the reason for the 21152's 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# enable bit for that event in the bridge control register and therefore does not have a device-specific disable bit.

§ §

8.0 Exclusive Access

This chapter describes the use of the LOCK# signal to implement exclusive access to a target for transactions that cross the 21152.

8.1 Concurrent Locks

The primary and secondary bus lock mechanisms operate concurrently except when a locked transaction crosses the 21152. 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.

8.2 Acquiring Exclusive Access Across the 21152

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

- The PCI bus must be idle.
- The LOCK# signal must be deasserted.

The initiator leaves the LOCK# signal deasserted during the address phase (only the first address phase of a dual address transaction) and asserts LOCK# one clock cycle later. Once a data transfer is completed from the target, the target lock has been achieved.

Locked transactions can cross the 21152 only in the downstream direction, from the primary bus to the secondary bus.

When the target resides on another PCI bus, the master must acquire not only the lock on its own PCI bus but also the lock on every bus between its bus and the target's bus. When the 21152 detects, on the primary bus, an initial locked transaction intended for a target on the secondary bus, the 21152 samples the address, transaction type, byte enable bits, and parity, as described in Section 4.6.4. It also samples the lock signal. 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 read transaction. Subsequent locked transactions can be read or 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 prefetched.

When the locked delayed read request is queued, the 21152 does not queue any more transactions until the locked sequence is finished. The 21152 signals a target retry to all transactions initiated subsequent to the locked read transaction that are intended for targets on the other side of the 21152. The 21152 allows any transactions queued before the locked transaction to complete before initiating the locked transaction.

When the locked delayed read request transaction moves to the head of the delayed transaction queue, the 21152 initiates the transaction as a locked read transaction by deasserting **s_lock_l** on the secondary bus during the first address phase, and by asserting **s_lock_l** one cycle later. If **s_lock_l** is already asserted (used by another initiator), the 21152 waits to request access to the

secondary bus until **s_lock_l** is sampled deasserted when the secondary bus is idle. Note that the existing lock on the secondary bus could not have crossed the 21152; otherwise, the pending queued locked transaction would not have been queued. When the 21152 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, the 21152 transfers the read data back to the initiator, and the lock is then also established on the primary bus.

For the 21152 to recognize and respond to the initiator, the initiator's subsequent attempts of the read transaction must use the locked transaction sequence (deassert **p_lock_l** during address phase, and assert **p_lock_l** one cycle later). If the LOCK# sequence is not used in subsequent attempts, a master time-out condition may result. When a master time-out condition occurs, **p_serr_l** is conditionally asserted (refer to Section 7.4), the read data and queued read transaction are discarded, and the **s_lock_l** signal is deasserted on the secondary bus.

Once the intended target has been locked, any subsequent locked transactions initiated on the primary bus that are forwarded by the 21152 are driven as locked transactions on the secondary bus.

When the 21152 receives a target abort or a master abort in response to the delayed locked read transaction, this status is passed back to the initiator, and no locks are established on either the target or the initiator bus. The 21152 resumes forwarding unlocked transactions in both directions.

When the 21152 detects, on the secondary bus, a locked delayed transaction request intended for a target on the primary bus, the 21152 queues and forwards the transaction as an unlocked transaction. The 21152 ignores **s_lock_l** for upstream transactions and initiates all upstream transactions as unlocked transactions.

8.3 Ending Exclusive Access

After the lock has been acquired on both the primary and secondary buses, the 21152 must maintain the lock on the secondary (target) bus for any subsequent locked transactions until the initiator relinquishes the lock.

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

An established target lock is maintained until the initiator relinquishes the lock. The 21152 does not know whether the current transaction is the last one in a sequence of locked transactions until the initiator deasserts the **p_lock_l** signal at the end of the transaction.

When the last locked transaction is a delayed transaction, the 21152 has already completed the transaction on the secondary bus. In this case, as soon as the 21152 detects that the initiator has relinquished the **p_lock_l** signal by sampling it in the deasserted state while **p_frame_l** is deasserted, the 21152 deasserts the **s_lock_l** signal on the secondary bus as soon as possible. Because of this behavior, **s_lock_l** may not be deasserted until several cycles after the last locked transaction has been completed on the secondary bus. As soon as the 21152 has deasserted **s_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, the 21152 deasserts **s_lock_l** on the secondary bus at the end of the transaction because the lock was relinquished at the end of the write transaction on the primary bus.

When the 21152 receives a target abort or a master abort in response to a locked delayed transaction, the 21152 returns a target abort when the initiator repeats the locked transaction. The initiator must then deassert **p_lock_l** at the end of the transaction. The 21152 sets the appropriate status bits, flagging the abnormal target termination condition (refer to Section 4.8). Normal forwarding of unlock posted and delayed transactions is resumed.

When the 21152 receives a target abort or a master abort in response to a locked posted write transaction, the 21152 cannot pass back that status to the initiator. The 21152 asserts **p_serr_1** when a target abort or a master abort is received during a locked posted write transaction, if the SERR# enable bit is set in the command register. Signal **p_serr_1** is asserted for the master abort condition if the master abort mode bit is set in the bridge control register (refer to Section 7.4).

§ §

9.0 PCI Bus Arbitration

The 21152 must arbitrate for use of the primary bus when forwarding upstream transactions, and for use of the secondary bus when forwarding downstream transactions. The arbiter for the primary bus resides external to the 21152, typically on the motherboard. For the secondary PCI bus, the 21152 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.

9.1 Primary PCI Bus Arbitration

The 21152 implements a request output pin, **p_req_l**, and a grant input pin, **p_gnt_l**, for primary PCI bus arbitration. The 21152 asserts **p_req_l** when forwarding transactions upstream; that is, it acts as initiator on the primary PCI bus.

For posted write transactions (refer to Section 4.5.1), **p_req_l** is asserted one cycle after **s_devsel_l** is asserted. For delayed read and write requests, **p_req_l** is not asserted until the transaction request has been completely queued in the delayed transaction queue (target retry has been returned to the initiator) and is at the head of the delayed transaction queue.

When **p_gnt_l** is asserted low by the primary bus arbiter after the 21152 has asserted **p_req_l**, the 21152 initiates a transaction on the primary bus during the next PCI clock cycle. When **p_gnt_l** is asserted to the 21152 when **p_req_l** is not asserted, the 21152 parks **p_ad**, **p_cbe_l,** and **p_par** by driving them to valid logic levels. When the primary bus is parked at the 21152 and the 21152 then has a transaction to initiate on the primary bus, the 21152 starts the transaction if **p_gnt_l** was asserted during the previous cycle.

9.2 Secondary PCI Bus Arbitration

The 21152 implements an internal secondary PCI bus arbiter. This arbiter supports four external masters in addition to the 21152. The internal arbiter can be disabled, and an external arbiter can be used instead for secondary bus arbitration.

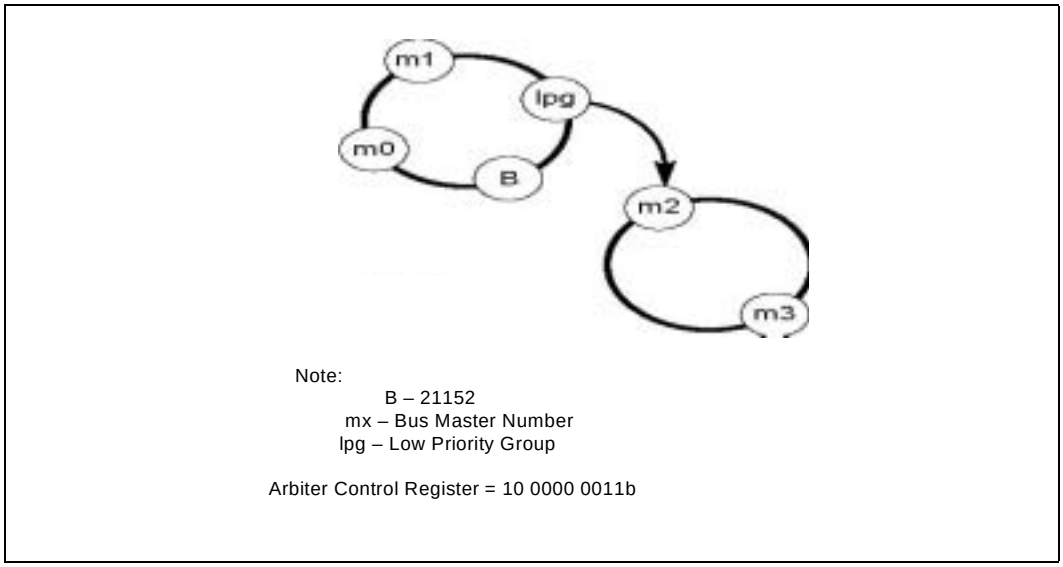
9.2.1 Secondary Bus Arbitration Using the Internal Arbiter

To use the internal arbiter, the secondary bus arbiter enable pin, **s_cfn_l,** must be tied low. The 21152 has four secondary bus request input pins, **s_req_l<3:0>**, and four secondary bus output grant pins, **s_gnt_l<3:0>**, to support external secondary bus masters. The 21152 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 programmable 2-level rotating algorithm. Two groups of masters are assigned, 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 18 shows an example of an internal arbiter where three masters, including the 21152, 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*, **m2**, *B*, *m0*, *m1*, **m3**, and so on.

Figure 18. Secondary Arbiter Example



Each bus master, including the 21152, 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 in device-specific configuration space. 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 the 21152 is assigned to the high priority group. The 21152 receives highest priority on the target bus every other transaction, and priority rotates evenly among the other masters.

Priorities are reevaluated every time **s_frame_l** is asserted, that is, 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 deasserts the asserted grant signal and asserts the grant corresponding to the new higher priority request on the next PCI clock cycle. When priorities are reevaluated, 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 the 21152 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 deasserts the grant. That master does not receive any more grants until it deasserts its request for at least one PCI clock cycle.

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 deasserts another. It deasserts one grant, and then asserts the next grant, no earlier than one PCI clock cycle later. If the secondary PCI bus is busy, that is, either **s_frame_l** or **s_irdy_l** is asserted, the arbiter can deassert one grant and assert another grant during the same PCI clock cycle.

9.2.2 Secondary Bus Arbitration Using an External Arbiter

The internal arbiter is disabled when the secondary bus central function control pin, **s_cfn_l**, is pulled high. An external arbiter must then be used.

When **s_cfn_l** is tied high, the 21152 reconfigures two pins to be external request and grant pins. The **s_gnt_l<0>** pin is reconfigured to be the 21152's external request pin because it is an output. The **s_req_l<0>** pin is reconfigured to be the external grant pin because it is an input. When an external arbiter is used, the 21152 uses the **s_gnt_l<0>** pin to request the secondary bus. When the reconfigured **s_req_l<0>** pin is asserted low after the 21152 has asserted **s_gnt_l<0>**, the 21152 initiates a transaction on the secondary bus one cycle later. If **s_req_l<0>** is asserted and the 21152 has not asserted **s_gnt_l<0>**, the 21152 parks the **s_ad**, **s_cbe_l**, and **s_par** pins by driving them to valid logic levels.

The unused secondary bus grant outputs, **s_gnt_l<3:1>**, are driven high. Unused secondary bus request inputs, **s_req_l<3:1>**, should be pulled high.

9.2.3 Bus Parking

Bus parking refers to driving the AD, C/BE#, 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 C/BE# signals should be driven first, with the PAR signal driven one cycle later.

The 21152 parks the primary bus only when **p_gnt_l** is asserted, **p_req_l** is deasserted, and the primary PCI bus is idle. When **p_gnt_l** is deasserted, the 21152 tristates the **p_ad**, **p_cbe_l**, and **p_par** signals on the next PCI clock cycle. If the 21152 is parking the primary PCI bus and wants to initiate a transaction on that bus, then the 21152 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, the 21152 keeps the secondary bus grant asserted to a particular master until a new secondary bus request comes along. After reset, the 21152 parks the

secondary bus at itself until transactions start occurring on the secondary bus. If the internal arbiter is disabled, the 21152 parks the secondary bus only when the reconfigured grant signal, **s_req_l<0>**, is asserted and the secondary bus is idle.

§ §

10.0 Clocks

This chapter provides information about the 21152 clocks.

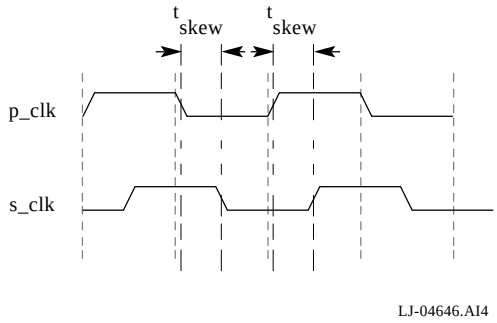
10.1 Primary and Secondary Clock Inputs

The 21152 implements a separate clock input for each PCI interface. The primary interface is synchronized to the primary clock input, **p_clk**, and the secondary interface is synchronized to the secondary clock input, **s_clk**.

The 21152 operates at a maximum frequency of 33 MHz, and **s_clk** always operates at the same frequency as **p_clk**.

The primary and secondary clock inputs must always maintain a synchronous relationship to each other; that is, their edge relationships to each other are well defined. The maximum skew between **p_clk** and **s_clk** rising edges is 7 ns, as is the maximum skew between **p_clk** and **s_clk** falling edges. The minimum skew between **p_clk** and **s_clk** edges is 0 ns. The secondary clock edge must never precede the primary clock edge. Figure 19 illustrates the timing relationship between the primary and the secondary clock inputs.

Figure 19. p_clk and s_clk Relative Timing



10.2 Secondary Clock Outputs

The 21152 has five secondary clock outputs, **s_clk_o<4:0>**, that can be used as clock inputs for up to four external secondary bus devices and for the 21152 secondary clock input.

The **s_clk_o** outputs are derived from **p_clk**. The **s_clk_o** edges are delayed from **p_clk** edges by a minimum of 0 ns and a maximum of 5 ns. The maximum skew between **s_clk_o** edges is 500 ps. Therefore, to meet the **p_clk** and **s_clk** requirements stated in Section 10.1, no more than 2 ns of delay is allowed for secondary clock etch returning to the device secondary clock inputs.

The rules for using secondary clocks are:

- Each secondary clock output is limited to one load.
- One of the secondary clock outputs must be used for the 21152 **s_clk** input.
- Intel recommends using an equivalent amount of etch on the board for all secondary clocks, to minimize skew between them, and a maximum delay of the etch of 2 ns.
- Intel recommends terminating or disabling unused secondary clock outputs to reduce power dissipation and noise in the system.

10.2.1 Disabling Unused Secondary Clock Outputs

When secondary clock outputs are not used, they can be individually disabled and driven high by writing the secondary clock control register in configuration space.

§ §

11.0 Reset

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

11.1 Primary Interface Reset

The 21152 has one reset input, **p_rst_l**. When **p_rst_l** is asserted, the following events occur:

- The 21152 immediately tristates all primary and secondary PCI interface signals.
- The 21152 performs a chip reset.
- Registers that have default values are reset.
  Appendix 17.0 lists the values of all configuration space registers after reset.

The **p_rst_l** asserting and deasserting edges can be asynchronous to **p_clk** and **s_clk**.

11.2 Secondary Interface Reset

The 21152 is responsible for driving the secondary bus reset signal, **s_rst_l**. The 21152 asserts **s_rst_l** when any of the following conditions is met:

- Signal **p_rst_l** is asserted.
  Signal **s_rst_l** remains asserted as long as **p_rst_l** is asserted.
- The secondary reset bit in the bridge control register is set.
  Signal **s_rst_l** remains asserted until a configuration write operation clears the secondary reset bit.
- The chip reset bit in the diagnostic control register is set.
  Signal **s_rst_l** remains asserted until a configuration write operation clears the secondary reset bit.

When **s_rst_l** is asserted, all secondary PCI interface control signals, including the secondary grant outputs, are immediately tristated. Signals **s_ad**, **s_cbe_l**, and **s_par** are driven low for the duration of **s_rst_l** assertion. All posted write and delayed transaction data buffers are reset; therefore, any transactions residing in 21152 buffers at the time of secondary reset are discarded.

When **s_rst_l** is asserted by means of the secondary reset bit, the 21152 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 21152 and secondary bus.

When the chip reset bit is set, all registers and chip state are reset and all signals are tristated. In addition, **s_rst_l** is asserted, and the secondary reset bit is automatically set. Signal **s_rst_l** remains asserted until a configuration write operation clears the secondary reset bit.

As soon as chip reset completes, within 20 PCI clock cycles after completion of the configuration write operation that sets the chip reset bit, the chip reset bit automatically clears and the chip is ready for configuration.

During chip reset, the 21152 is inaccessible.

§ §

12.0 PCI Power Management

The 21152–AB incorporates functionality that complies fully with the Advanced Configuration Power Interface (ACPI) and the *PCI Power Management Specification*. These features include:

- PCI Power Management registers using the enhanced capabilities port (ECP) address mechanism
- Support for D0, $D3_{hot}$, and $D3_{cold}$ power management states
- Support for D0, D1, D2, $D3_{hot}$, and $D3_{cold}$ power management states for devices behind the bridge
- Support of the B2 secondary bus power state when in the $D3_{hot}$ power management state

The 21152–AA *does not* include these features.

Table 32 shows the states and related actions that the 21152 performs during power management transitions. (No other transactions are permitted.)

Table 32. Power Management Transitions

Current State	Next State	Action
D0	$D3_{cold}$	Power has been removed from the 21152. A power-up reset must be performed to bring the 21152 to D0.
D0	$D3_{hot}$	If enabled to do so by the **bpcc** pin, the 21152 will disable the secondary clocks and drive them low.
D0	D2	Unimplemented power state. The 21152 will ignore the write to the power state bits (power state remains at D0).
D0	D1	Unimplemented power state. The 21152 will ignore the write to the power state bits (power state remains at D0).
$D3_{hot}$	D0	The 21152 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.
$D3_{hot}$	D3cold	Power has been removed from the 21152. A power-up reset must be performed to bring the 21152 to D0.
$D3_{cold}$	D0	Power-up reset. The 21152 performs the standard power-up reset functions as described in Chapter 11.0.

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

§ §

13.0 Configuration Space Registers

This chapter provides a detailed description of the 21152 configuration space registers. The chapter is divided into two sections: Section 13.1 describes the standard 21152 PCI-to-PCI bridge configuration registers, and Section 13.2 describes the 21152 device-specific configuration registers.

The 21152 configuration space uses the PCI-to-PCI bridge standard format specified in the *PCI-to-PCI Bridge Architecture Specification*. The header type at configuration address 0Eh reads as 01h, indicating that this device uses the PCI-to-PCI bridge format.

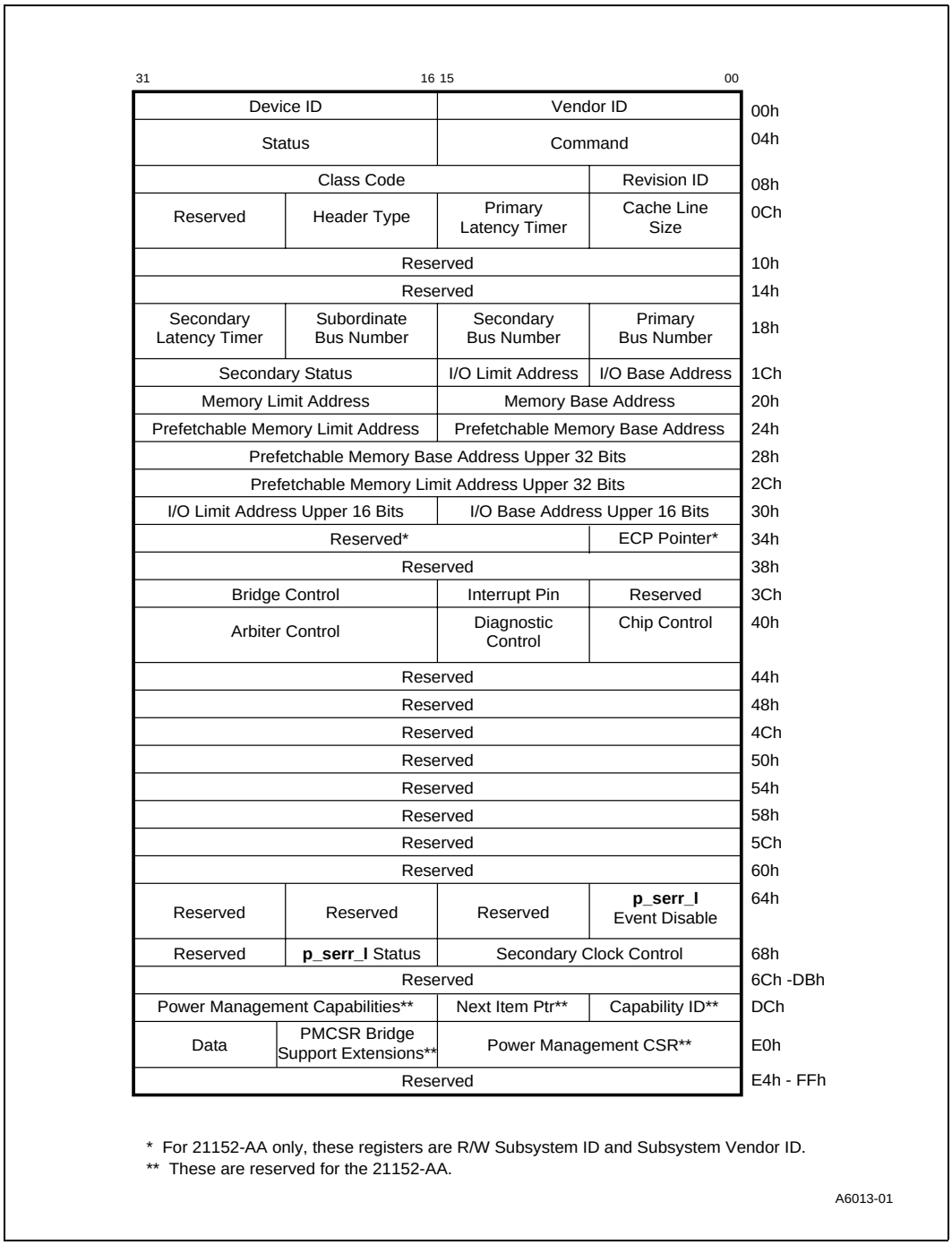
The 21152 also contains device-specific registers, starting at address 40h. Use of these registers is not required for standard PCI-to-PCI bridge implementations.

The configuration space registers can be accessed only from the primary PCI bus. To access a register, perform a Type 0 format configuration read or write operation to that register. During the Type 0 address phase, **p_ad<7:2>** indicates the Dword offset of the register. During the data phase, **p_cbe_l<3:0>** selects the bytes in the Dword that is being accessed.

Caution: Software changes the configuration register values that affect 21152 behavior only during initialization. Change these values subsequently only when both the primary and secondary PCI buses are idle, and the data buffers are empty; otherwise, the behavior of the 21152 is unpredictable.

Figure 20 shows a summary of configuration space.

Figure 20. 21152 Configuration Space



31	16	15	00	
Device ID		Vendor ID		00h
Status		Command		04h
Class Code			Revision ID	08h
Reserved	Header Type	Primary Latency Timer	Cache Line Size	0Ch
Reserved				10h
Reserved				14h
Secondary Latency Timer	Subordinate Bus Number	Secondary Bus Number	Primary Bus Number	18h
Secondary Status		I/O Limit Address	I/O Base Address	1Ch
Memory Limit Address		Memory Base Address		20h
Prefetchable Memory Limit Address		Prefetchable Memory Base Address		24h
Prefetchable Memory Base Address Upper 32 Bits				28h
Prefetchable Memory Limit Address Upper 32 Bits				2Ch
I/O Limit Address Upper 16 Bits		I/O Base Address Upper 16 Bits		30h
Reserved*			ECP Pointer*	34h
Reserved				38h
Bridge Control		Interrupt Pin	Reserved	3Ch
Arbiter Control		Diagnostic Control	Chip Control	40h
Reserved				44h
Reserved				48h
Reserved				4Ch
Reserved				50h
Reserved				54h
Reserved				58h
Reserved				5Ch
Reserved				60h
Reserved	Reserved	Reserved	**p_serr_l** Event Disable	64h
Reserved	**p_serr_l** Status	Secondary Clock Control		68h
Reserved				6Ch -DBh
Power Management Capabilities**		Next Item Ptr**	Capability ID**	DCh
Data	PMCSR Bridge Support Extensions**	Power Management CSR**		E0h
Reserved				E4h - FFh

* For 21152-AA only, these registers are R/W Subsystem ID and Subsystem Vendor ID.
** These are reserved for the 21152-AA.

A6013-01

13.1 PCI-to-PCI Bridge Standard Configuration Registers

This section provides a detailed description of the PCI-to-PCI bridge standard configuration registers.

Each field has a separate description.

Fields that have the same configuration Dword address are selectable by turning on (driving low) the appropriate byte enable bits on **p_cbe_l** during the data phase. To select all fields of a configuration address, drive all byte enable bits low.

All reserved fields and registers are read only and always return 0.

13.1.1 Vendor ID Register — Offset 00h

This section describes the vendor ID register.

Dword address = 00h

Byte enable **p_cbe_l<3:0>** = xx00b

Dword Bit	Name	R/W	Description
15:0	Vendor ID	R	Identifies Intel Corporation as the vendor of this device. Internally hardwired to be 1011h.

13.1.2 Device ID Register — Offset 02h

This section describes the device ID register.

Dword address = 00h

Byte enable **p_cbe_l<3:0>** = 00xxb

Dword Bit	Name	R/W	Description
31:16	Device ID	R	Identifies this device as the 21152. Internally hardwired to be 24h.

13.1.3 Command Register — Offset 04h

This section describes the command register.

These bits affect the behavior of the 21152 primary interface, except where noted. Some of the bits are repeated in the bridge control register, to act on the secondary interface.

This register must be initialized by configuration software.

Dword address = 04h

Byte enable **p_cbe_l<3:0>** = xx00b

Dword Bit	Name	R/W	Description
0	I/O space enable	R/W	Controls the 21152's response to I/O transactions on the primary interface. When 0: The 21152 does not respond to I/O transactions initiated on the primary bus. When 1: The 21152's response to I/O transactions initiated on the secondary bus is enabled. Reset value: 0.
1	Memory space enable	R/W	Controls the 21152's response to memory transactions on the21152 primary interface. When 0: The 21152 does not respond to memory transactions initiated on the primary bus. When 1: The 21152's response to memory transactions initiated on the primary bus is enabled. Reset value: 0.
2	Master enable	R/W	Controls the 21152's ability to initiate memory and I/O transactions on the primary bus on behalf of an initiator on the secondary bus. Forwarding of configuration transactions is not affected. When 0: The 21152 does not respond to I/O or memory transactions on the secondary interface and does not initiate I/O or memory transactions on the primary interface. When 1: The 21152 is enabled to operate as an initiator on the primary bus and responds to I/O and memory transactions initiated on the secondary bus. Reset value: 0.
3	Special cycle enable	R	The 21152 ignores special cycle transactions, so this bit is read only and returns 0.
4	Memory write and invalidate enable	R	The 21152 generates memory write and invalidate transactions only when operating on behalf of another master whose memory write and invalidate transaction is crossing the 21152. This bit is read only and returns 0.

Dword Bit	Name	R/W	Description
5	VGA snoop enable	R/W	Controls the 21152's response to VGA-compatible palette write transactions. VGA palette write transactions correspond to I/O transactions whose address bits are as follows: • **p_ad<9:0>** are equal to 3C6h, 3C8h, and 3C9h. • **p_ad<15:10>** are not decoded. • **p_ad<31:16>** must be 0. When 0: VGA palette write transactions on the primary interface are ignored unless they fall inside the 21152's I/O address range. When 1: VGA palette write transactions on the primary interface are positively decoded and forwarded to the secondary interface. Reset value: 0.
6	Parity error response	R/W	Controls the 21152's response when a parity error is detected on the primary interface. When 0: The 21152 does not assert **p_perr_l**, nor does it set the data parity reported bit in the status register. The 21152 does not report address parity errors by asserting **p_serr_l**. When 1: The 21152 drives **p_perr_l** and conditionally sets the data parity reported bit in the status register when a data parity error is detected (Refer to Chapter 7.0). The 21152 allows **p_serr_l** assertion when address parity errors are detected on the primary interface. Reset value: 0.
7	Wait cycle control	R	Reads as 0 to indicate that the 21152 does not perform address or data stepping.
8	SERR# enable	R/W	Controls the enable for **p_serr_l** on the primary interface. When 0: Signal **p_serr_l** cannot be driven by the 21152. When 1: Signal **p_serr_l** can be driven low by the 21152 under the conditions described in Section 7.4. Reset value: 0.
9	Fast back-to-back enable	R/W	Controls the ability of the 21152 to generate fast back-to-back transactions on the primary bus. When 0: The 21152 does not generate back-to-back transactions on the primary bus. When 1: The 21152 is enabled to generate back-to-back transactions on the primary bus. Reset value: 0.
15:10	Reserved	R	Reserved. Returns 0 when read.

13.1.4 Status Register — Offset 06h

This section describes the status register.

These bits affect the status of the 21152 primary interface. Bits reflecting the status of the secondary interface are found in the secondary status register. W1TC indicates that writing 1 to a bit sets that bit to 0. Writing 0 has no effect.

Dword address = 04h

Byte enable **p_cbe_l<3:0>** = 00xxb

Dword Bit	Name	R/W	Description
19:16	Reserved	R	Reserved. Returns 0 when read.
20	ECP	R	Enhanced Capabilities Port (ECP) enable. Reads as 1 in the 21152–AB and later revisions to indicate that the 21152–AB supports an enhanced capabilities list. The 21152–AA reads as 0 to show that this capability is not supported.
21	66-MHz capable	R	Indicates whether the primary interface is 66 MHz capable. Reads as 0 to indicate that the primary interface operates at a maximum frequency of 33 MHz.
22	Reserved	R	Reserved. Returns 0 when read.
23	Fast back-to-back capable	R	Reads as 1 to indicate that the 21152 is able to respond to fast back-to-back transactions on the primary interface.
24	Data parity detected	R/W1TC	This bit is set to 1 when all of the following are true: The 21152 is a master on the primary bus. Signal **p_perr_l** is detected asserted, or a parity error is detected on the primary bus. The parity error response bit is set in the command register. Reset value: 0.
26:25	DEVSEL# timing	R	Indicates slowest response to a nonconfiguration command on the primary interface. Reads as 01b to indicate that the 21152 responds no slower than with medium timing.
27	Signaled target abort	R/W1TC	This bit is set to 1 when the 21152 is acting as a target on the primary bus and returns a target abort to the primary master. Reset value: 0.
28	Received target abort	R/W1TC	This bit is set to 1 when the 21152 is acting as a master on the primary bus and receives a target abort from the primary target. Reset value: 0.
29	Received master abort	R/W1TC	This bit is set to 1 when the 21152 is acting as a master on the primary bus and receives a master abort.
30	Signaled system error	R/W1TC	This bit is set to 1 when the 21152 has asserted **p_serr_l**. Reset value: 0.
31	Detected parity error	R/W1TC	This bit is set to 1 when the 21152 detects an address or data parity error on the primary interface. Reset value: 0.

13.1.5 Revision ID Register — Offset 08h

This section describes the revision ID register.

Dword address = 08h

Byte enable **p_cbe_l<3:0>** = xxx0b

Dword Bit	Name	R/W	Description
7:0	Revision ID	R	Indicates the revision number of this device. Revisions 00h through 02h indicate 21152–AA. The 21152–AB starts at Rev. ID 03h.

13.1.6 Programming Interface Register — Offset 09h

This section describes the programming interface register.

Dword address = 08h

Byte enable **p_cbe_l<3:0>** = xx0xb

Dword Bit	Name	R/W	Description
15:8	Programming interface	R	No programming interfaces have been defined for PCI-to-PCI bridges. Reads as 0.

13.1.7 Subclass Code Register — Offset 0Ah

This section describes the subclass code register.

Dword address = 08h

Byte enable **p_cbe_l<3:0>** = x0xxb

Dword Bit	Name	R/W	Description
23:16	Subclass code	R	Reads as 04h to indicate that this bridge device is a PCI-to-PCI bridge.

13.1.8 Base Class Code Register — Offset 0Bh

This section describes the base class code register.

Dword address = 08h

Byte enable **p_cbe_l<3:0>** = 0xxxb

Dword Bit	Name	R/W	Description
31:24	Base class code	R	Reads as 06h to indicate that this device is a bridge device.

13.1.9 Cache Line Size Register — Offset 0Ch

This section describes the cache line size register.

Dword address = 0Ch

Byte enable **p_cbe_l<3:0>** = xxx0b

Dword Bit	Name	R/W	Description
7:0	Cache line size	R/W	Designates the cache line size for the system in units of 32-bit Dwords. Used for prefetching memory read transactions and for terminating memory write and invalidate transactions. The cache line size should be written as a power of 2. If the value is not a power of 2 or is greater than 16, the 21152 behaves as if the cache line size were 0. Reset value: 0.

13.1.10 Primary Latency Timer Register — Offset 0Dh

This section describes the primary latency timer register.

Dword address = 0Ch

Byte enable **p_cbe_l<3:0>** = xx0xb

Dword Bit	Name	R/W	Description
15:8	Master latency timer	R/W	Master latency timer for the primary interface. Indicates the number of PCI clock cycles from the assertion of **p_frame_l** to the expiration of the timer when the 21152 is acting as a master on the primary interface. All bits are writable, resulting in a granularity of one PCI clock cycle. When 0: The 21152 relinquishes the bus after the first data transfer when the 21152's primary bus grant has been deasserted, with the exception of memory write and invalidate transactions. Reset value: 0.

13.1.11 Header Type Register — Offset 0Eh

This section describes the header type register.

Dword address = 0Ch

Byte enable **p_cbe_l<3:0>** = x0xxb

Dword Bit	Name	R/W	Description
23:16	Header type	R	Defines the layout of addresses 10h through 3Fh in configuration space. Reads as 01h to indicate that the register layout conforms to the standard PCI-to-PCI bridge layout.

13.1.12 Primary Bus Number Register — Offset 18h

This section describes the primary bus number register.

This register must be initialized by configuration software.

Dword address = 18h

Byte enable **p_cbe_l<3:0>** = xxx0b

Dword Bit	Name	R/W	Description
7:0	Primary bus number	R/W	Indicates the number of the PCI bus to which the primary interface is connected. The 21152 uses this register to decode Type 1 configuration transactions on the secondary interface that should either be converted to special cycle transactions on the primary interface or passed upstream unaltered. Reset value: 0.

13.1.13 Secondary Bus Number Register — Offset 19h

This section describes the secondary bus number register.

This register must be initialized by configuration software.

Dword address = 18h

Byte enable **p_cbe_l<3:0>** = xx0xb

Dword Bit	Name	R/W	Description
15:8	Secondary bus number	R/W	Indicates the number of the PCI bus to which the secondary interface is connected. The 21152 uses this register to determine when to respond to and forward Type 1 configuration transactions on the primary interface, and to determine when to convert them to Type 0 or special cycle transactions on the secondary interface. Reset value: 0.

13.1.14 Subordinate Bus Number Register — Offset 1Ah

This section describes the subordinate bus number register.

This register must be initialized by configuration software.

Dword address = 18h

Byte enable **p_cbe_l<3:0>** = x0xxb

Dword Bit	Name	R/W	Description
23:16	Subordinate bus number	R/W	Indicates the number of the highest numbered PCI bus that is behind (or subordinate to) the 21152. Used in conjunction with the secondary bus number to determine when to respond to Type 1 configuration transactions on the primary interface and pass them to the secondary interface as a Type 1 configuration transaction. Reset value: 0.

13.1.15 Secondary Latency Timer Register — Offset 1Bh

This section describes the secondary latency timer register.

Dword address = 18h

Byte enable **p_cbe_l<3:0>** = 0xxxb

Dword Bit	Name	R/W	Description
31:24	Secondary latency timer	R/W	Master latency timer for the secondary interface. Indicates the number of PCI clock cycles from the assertion of **s_frame_l** to the expiration of the timer when the 21152 is acting as a master on the secondary interface. All bits are writable, resulting in a granularity of one PCI clock cycle. When 0: The 21152 ends the transaction after the first data transfer when the 21152's secondary bus grant has been deasserted, with the exception of memory write and invalidate transactions. Reset value: 0.

13.1.16 I/O Base Address Register — Offset 1Ch

This section describes the I/O base address register.

This register must be initialized by configuration software.

Dword address = 1Ch

Byte enable **p_cbe_l<3:0>** = xxx0b

Dword Bit	Name	R/W	Description
3:0	32-bit indicator	R	The low 4 bits of this register read as 1h to indicate that the 21152 supports 32-bit I/O address decoding.
7:4	I/O base address <15:12>	R/W	Defines the bottom address of an address range used by the 21152 to determine when to forward I/O transactions from one interface to the other. The upper 4 bits are writable and correspond to address bits <15:12>. The lower 12 bits of the address 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 register. The I/O address range adheres to 4 KB alignment and granularity. Reset value: 0.

13.1.17 I/O Limit Address Register — Offset 1Dh

This section describes the I/O limit address register.

This register must be initialized by configuration software.

Dword address = 1Ch

Byte enable **p_cbe_l<3:0>** = xx0xb

Dword Bit	Name	R/W	Description
11:8	32-bit indicator	R/W	The low 4 bits of this register read as 1h to indicate that the 21152 supports 32-bit I/O address decoding.
15:12	I/O limit address <15:12>	R/W	Defines the top address of an address range used by the 21152 to determine when to forward I/O transactions from one interface to the other. The upper 4 bits are writable and correspond to address bits <15:12>. The lower 12 bits of the address 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 register. The I/O address range adheres to 4 KB alignment and granularity. Reset value: 0.

13.1.18 Secondary Status Register — Offset 1Eh

This section describes the secondary status register.

These bits reflect the status of the 21152 secondary interface. W1TC indicates that writing 1 to that bit sets the bit to 0. Writing 0 has no effect.

Dword address = 1Ch

Byte enable **p_cbe_l<3:0>** = 00xxb

Dword Bit	Name	R/W	Description
20:16	Reserved	R	Reserved. Returns 0 when read.
21	66 MHz capable	R	Indicates whether the secondary interface is 66 MHz capable. Reads as 0 to indicate that the secondary interface operates at a maximum frequency of 33 MHz.
22	Reserved	R	Reserved. Returns 0 when read.
23	Fast back-to-back capable	R	Reads as 1 to indicate that the 21152 is able to respond to fast back-to-back transactions on the secondary interface.
24	Data parity detected	R/W1TC	This bit is set to 1 when all of the following are true: • The 21152 is a master on the secondary bus. • Signal **s_perr_l** is detected asserted, or a parity error is detected on the secondary bus. • The parity error response bit is set in the bridge control register. Reset value: 0.
26:25	DEVSEL# timing	R	Indicates slowest response to a command on the secondary interface. Reads as 01b to indicate that the 21152 responds no slower than with medium timing.
27	Signaled target abort	R/W1TC	This bit is set to 1 when the 21152 is acting as a target on the secondary bus and returns a target abort to the secondary bus master. Reset value: 0.
28	Received target abort	R/W1TC	This bit is set to 1 when the 21152 is acting as a master on the secondary bus and receives a target abort from the secondary bus target. Reset value: 0.
29	Received master abort	R/W1TC	This bit is set to 1 when the 21152 is acting as an initiator on the secondary bus and receives a master abort. Reset value: 0.
30	Received system error	R/W1TC	This bit is set to 1 when the 21152 detects the assertion of **s_serr_l** on the secondary interface. Reset value: 0.
31	Detected parity error	R/W1TC	This bit is set to 1 when the 21152 detects an address or data parity error on the secondary interface. Reset value: 0.

13.1.19 Memory Base Address Register — Offset 20h

This section describes the memory base address register.

This register must be initialized by configuration software.

Dword address = 20h

Byte enable **p_cbe_l<3:0>** = xx00b

Dword Bit	Name	R/W	Description
3:0	Reserved	R	The low 4 bits of this register are read only and return 0.
15:4	Memory base address <31:20>	R/W	Defines the bottom address of an address range used by the 21152 to determine when to forward memory transactions from one interface to the other. The upper 12 bits are writable and correspond to address bits <31:20>. The lower 20 bits of the address are assumed to be 0. The memory address range adheres to 1 MB alignment and granularity. Reset value: 0.

13.1.20 Memory Limit Address Register — Offset 22h

This section describes the memory limit address register.

This register must be initialized by configuration software.

Dword address = 20h

Byte enable **p_cbe_l<3:0>** = 00xxb

Dword Bit	Name	R/W	Description
19:16	Reserved	R	The low 4 bits of this register are read only and return 0.
31:20	Memory limit address <31:20>	R/W	Defines the top address of an address range used by the 21152 to determine when to forward memory transactions from one interface to the other. The upper 12 bits are writable and correspond to address bits <31:20>. The lower 20 bits of the address are assumed to be FFFFFh. The memory address range adheres to 1 MB alignment and granularity. Reset value: 0.

13.1.21 Prefetchable Memory Base Address Register — Offset 24h

This section describes the prefetchable memory base address register.

This register must be initialized by configuration software.

Dword address = 24h

Byte enable **p_cbe_l<3:0>** = xx00b

Dword Bit	Name	R/W	Description
3:0	64-bit indicator	R	The low 4 bits of this register are read only and return 1h to indicate that this range supports 64-bit addressing.
15:4	Prefetchable memory base address <31:20>	R/W	Defines the bottom address of an address range used by the 21152 to determine when to forward memory read and write transactions from one interface to the other. The upper 12 bits are writable and correspond to address bits <31:20>. The lower 20 bits of the address are assumed to be 0. The memory base register upper 32 bits contains the upper half of the base address. The memory address range adheres to 1 MB alignment and granularity. Reset value: 0.

13.1.22 Prefetchable Memory Limit Address Register — Offset 26h

This section describes the prefetchable memory limit address register.

This register must be initialized by configuration software.

Dword address = 24h

Byte enable **p_cbe_l<3:0>** = 00xxb

Dword Bit	Name	R/W	Description
19:16	64-bit indicator	R	The low 4 bits of this register are read only and return 1h to indicate that this range supports 64-bit addressing.
31:20	Prefetchable memory limit address <31:20>	R/W	Defines the top address of an address range used by the 21152 to determine when to forward memory read and write transactions from one interface to the other. The upper 12 bits are writable and correspond to address bits <31:20>. The lower 20 bits of the address are assumed to be FFFFFh. The memory limit upper 32 bits register contains the upper half of the limit address. The memory address range adheres to 1 MB alignment and granularity. Reset value: 0.

13.1.23 Prefetchable Memory Base Address Upper 32 Bits Register — Offset 28h

This section describes the prefetchable memory base address upper 32 bits register.

This register must be initialized by configuration software.

Dword address = 28h

Byte enable **p_cbe_l<3:0>** = 0000b

Dword Bit	Name	R/W	Description
31:0	Upper 32 prefetchable memory base address <63:32>	R/W	Defines the upper 32 bits of a 64-bit bottom address of an address range used by the 21152 to determine when to forward memory read and write transactions from one interface to the other. The memory address range adheres to 1 MB alignment and granularity. Reset value: 0.

13.1.24 Prefetchable Memory Limit Address Upper 32 Bits Register — Offset 2Ch

This section describes the prefetchable memory limit address upper 32 bits register.

This register must be initialized by configuration software.

Dword address = 2Ch

Byte enable **p_cbe_l<3:0>** = 0000b

Dword Bit	Name	R/W	Description
31:0	Upper 32 prefetchable memory limit address <63:32>	R/W	Defines the upper 32 bits of a 64-bit top address of an address range used by the 21152 to determine when to forward memory read and write transactions from one interface to the other. Extra read transactions should have no side effects. The memory address range adheres to 1 MB alignment and granularity. Reset value: 0.

13.1.25 I/O Base Address Upper 16 Bits Register — Offset 30h

This section describes the I/O base address upper 16 bits register.

This register must be initialized by configuration software.

Dword address = 30h

Byte enable **p_cbe_l<3:0>** = xx00b

Dword Bit	Name	R/W	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 used by the 21152 to determine when to forward I/O transactions from one interface to the other. The I/O address range adheres to 4 KB alignment and granularity. Reset value: 0.

13.1.26 I/O Limit Address Upper 16 Bits Register — Offset 32h

This section describes the I/O limit address upper 16 bits register.

This register must be initialized by configuration software.

Dword address = 32h

Byte enable **p_cbe_l<3:0>** = 00xxb

Dword Bit	Name	R/W	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 used by the 21152 to determine when to forward I/O transactions from one interface to the other. The I/O address range adheres to 4 KB alignment and granularity. Reset value: 0.

13.1.27 Capabilities Pointer Register — Offset 34h

This section describes the capabilities pointer register.

Dword address = 34h

Byte enable **p_cbe_l<3:0>** = 0000b

Dword Bit	Name	R/W	Description
7:0	ECP_PTR	R	Enhanced Capabilities Port (ECP) offset pointer. Reads as DCh in the 21152–AB and later revisions to indicate that the first item, which corresponds to the power management registers, resides at that configuration offset. This is a R/W register with no side effects in the 21152–AA.
31:8	Reserved	R	Reserved. The 21152–AB and later revisions return 0 when read. This is a R/W register with no side effects in the 21152–AA.

13.1.28 Interrupt Pin — Offset 3Dh

This section describes the interrupt pin register.

Dword address = 3Ch

Byte enable **p_cbe_l<3:0>** = xx0xb

Dword Bit	Name	R/W	Description
15:8	Interrupt pin	R	Reads as 0 to indicate that the 21152 does not have an interrupt pin.

13.1.29 Bridge Control — Offset 3Eh

This section describes the bridge control register.

This register must be initialized by configuration software.

Dword address = 3Eh

Byte enable **p_cbe_l<3:0>** = 00xxb

Dword Bit	Name	R/W	Description
16	Parity error response	R/W	Controls the 21152's response when a parity error is detected on the secondary interface. When 0: The 21152 does not assert **s_perr_l,** nor does it set the data parity reported bit in the secondary status register. The 21152 does not report address parity errors by asserting **p_serr_l**. When 1: The 21152 drives **s_perr_l** and conditionally sets the data parity reported bit in the secondary status register when a data parity error is detected on the secondary interface (Refer to Chapter 7.0). Also must be set to 1 to allow **p_serr_l** assertion when address parity errors are detected on the secondary interface. Reset value: 0.
17	SERR# forward enable	R/W	Controls whether the 21152 asserts **p_serr_l** when it detects **s_serr_l** asserted. When 0: The 21152 does not drive **p_serr_l** when it detects **s_serr_l** asserted. When 1: The 21152 asserts **p_serr_l** when **s_serr_l** is detected asserted (the primary SERR# driver enable bit must also be set). Reset value: 0.
18	ISA enable	R/W	Modifies the 21152's response to ISA I/O addresses. Applies only to those addresses falling within the I/O base and limit address registers and within the first 64KB of PCI I/O space. When 0: The 21152 forwards all I/O transactions downstream that fall within the I/O base and limit address registers. When 1: The 21152 ignores primary bus I/O transactions within the I/O base and limit address registers and within the first 64 KB of PCI I/O space that address the last 768 bytes in each 1 KB block. Secondary bus I/O transactions are forwarded upstream if the address falls within the last 768 bytes in each 1 KB block. Reset value: 0.

Dword Bit	Name	R/W	Description
19	VGA enable	R/W	Modifies the 21152's response to VGA-compatible addresses. When 0: VGA transactions are ignored on the primary bus unless they fall within the I/O base and limit address registers and the ISA mode is 0. When 1: The 21152 positively decodes and forwards the following transactions downstream, regardless of the values of the I/O base and limit registers, ISA mode bit, or VGA snoop bit: • Memory transactions addressing 000A0000h–000BFFFFh • I/O transaction addressing: — **p_ad<9:0>** = 3B0h–3BBh and 3C0h–3DFh — **p_ad<15:10>** are not decoded. — **p_ad<31:16>** = 0000h. I/O and memory space enable bits must be set in the command register. The transactions listed here are ignored by the 21152 on the secondary bus. Reset value: 0.
20	Reserved	R	Reserved. Returns 0 when read.
21	Master abort mode	R/W	Controls the 21152's behavior when a master abort termination occurs in response to a transaction initiated by the 21152 on either the primary or secondary PCI interface. When 0: The 21152 asserts TRDY# on the initiator bus for delayed transactions, and FFFF FFFFh for read transactions. For posted write transactions, **p_serr_l** is not asserted. When 1: The 21152 returns a target abort on the initiator bus for delayed transactions. For posted write transactions, the 21152 asserts **p_serr_l** if the SERR# enable bit is set in the command register. Reset value: 0.
22	Secondary bus reset	R/W	Controls **s_rst_l** on the secondary interface. When 0: The 21152 deasserts **s_rst_l**. When 1: The 21152 asserts **s_rst_l**. When **s_rst_l** is asserted, the data buffers and the secondary interface are initialized back to reset conditions. The primary interface and configuration registers are not affected by the assertion of **s_rst_l**. Reset value: 0.
23	Fast back-to-back enable		Controls the ability of the 21152 to generate fast back-to-back transactions on the secondary interface. When 0: The 21152 does not generate fast back-to-back transactions on the secondary PCI bus. When 1: The 21152 is enabled to generate fast back-to-back transactions on the secondary PCI bus. Reset value: 0.
24	Primary master time-out	R/W	Sets the maximum number of PCI clock cycles that the 21152 waits for an initiator on the primary bus to repeat a delayed transaction request. The counter starts once the delayed transaction completion is at the head of the queue. If the master has not repeated the transaction at least once before the counter expires, the 21152 discards the transaction from its queues. When 0: The primary master time-out value is 2^{15} PCI clock cycles, or 0.983 ms for a 33-MHz bus. When 1: The value is 2^{10} PCI clock cycles, or 30.7 µs for a 33-MHz bus. Reset value: 0.

Dword Bit	Name	R/W	Description
25	Secondary master timeout	R/W	Sets the maximum number of PCI clock cycles that the 21152 waits for an initiator on the secondary bus to repeat a delayed transaction request. The counter starts once the delayed transaction completion is at the head of the queue. If the master has not repeated the transaction at least once before the counter expires, the 21152 discards the transaction from its queues. When 0: The primary master time-out value is 2^{15} PCI clock cycles, or 0.983 ms for a 33 MHz bus. When 1: The value is 2^{10} PCI clock cycles, or 30.7 µs for a 33 MHz bus. Reset value: 0.
26	Master timeout status	R/W1TC	This bit is set to 1 when either the primary master time-out counter or the secondary master time-out counter expires and a delayed transaction is discarded from the 21152's queues. Write 1 to clear. Reset value: 0.
27	Master timeout SERR# enable	R/W	Controls assertion of **p_serr_l** during a master timeout. When 0: Signal **p_serr_l** is not asserted as a result of a master time-out. When 1: Signal **p_serr_l** is asserted when either the primary master time-out counter or the secondary master time-out counter expires and a delayed transaction is discarded from the 21152's queues. The SERR# enable bit in the command register must also be set. Reset value: 0.
31:28	Reserved	R	Reserved. Returns 0 when read.

13.1.30 Capability ID Register — Offset DCh

This section describes the capability ID register. (Implemented in the 21152-AB and later revisions only. In the 21152-AA, this register is reserved.)

Dword address = DCh

Byte enable **p_cbe_l<3:0>** = xxx0b

Dword Bits	Name	R/W	Description
7:0	CAP_ID	R	Enhanced capabilities ID. Reads only as 01h to indicate that these are power management enhanced capability registers.

13.1.31 Next Item Register — Offset DDh

This section describes the next item register. (Implemented in the 21152-AB and later revisions only. In the 21152-AA, this register is reserved.)

Dword address = DCh

Byte enable **p_cbe_l<3:0>** = xx0x

Dword Bit	Name	R/W	Description
15:8	NEXT_ITEM	R	Next Item Pointer. Reads as 0 to indicate that there are not other ECP registers.

13.1.32 Power Management Capabilities Registers — Offset DEh

This section describes the power management capabilities registers. (Implemented in the 21152-AB and later revisions only. In the 21152-AA, this register is reserved.)

Dword address = DCh

Byte enable **p_cbe_l<3:0>** = 00xx

Dword Bit	Name	R/W	Description
18:16	PM_VER	R	Power Management Revision. Reads as 001 to indicate that this device is compliant to Revision 1.0 of the *PCI Power Management Interface Specification*.
19	PME#Clock	R	PME# Clock Required. Reads as 0 because this device does not support the PME# pin.
20	AUX	R	Auxiliary Power Support. Reads as 0 because this device does not have PME# support or an auxiliary power source.
21	DSI	R	Device-Specific Initialization. Reads as 0 to indicate that this device does not have device-specific initialization requirements.
24:22	Reserved	R	Reserved. Read as 000b.
25	D1	R	D1 Power State Support. Reads as 0 to indicate that this device does not support the D1 power management state.
26	D2	R	D2 Power State Support. Reads as 0 to indicate that this device does not support the D2 power management state.
31:27	PME_SUP	R	PME# Support. Reads as 0 to indicate that this device does not support the PME# pin.

13.1.33 Power Management Control and Status Registers — Offset E0h

This section describes the power management control and status registers. (Implemented in the 21152-AB and later revisions only. In the 21152-AA, this register is reserved.)

Dword address = E0h

Byte enable **p_cbe_l** = xx00

Dword Bit	Name	R/W	Description
1:0	PWR_STATE	R/W	Power State. Reflects the current power state of this device. If an unimplemented power state is written to this register, the 21152 completes the write transaction, ignores the write data, and does not change the value of this field. Writing a value of D0 when the previous state was D3 will cause a chip reset to occur (without asserting **s_rst_l**). 00b: D0 01b: D1 (not implemented) 10b: D2 (not implemented) 11b: D3 Reset value: 00b
7:2	Reserved	R	Reserved. Reads as 00000b.
8	PME_EN	R	PME# Enable. Reads as 0 because the PME# pin is not implemented.
12:9	DATA_SEL	R	Data Select. Reads as 0000b because the data register is not implemented.
14:13	DATA_ SCALE	R	Data Scale. Reads as 00b because the data register is not implemented.
15	PME_STAT	R	PME Status. Reads as 0 because the PME# pin is not implemented.

13.1.34 PPB Support Extensions Registers — Offset E2h

This section describes the PPB support extensions registers. (Implemented in the 21152-AB and later revisions only. In the 21152-AA, this register is reserved.)

Dword address = E0h

Byte enable **p_cbe_l<3:0>** = x0xx

Dword Bit	Name	R/W	Description
21:16	Reserved	R	Reserved. Read only as 000000b.
22	B2_B3	R	B2_B3 Support for $D3_{hot}$. When the BPCC_En bit (bit 24) reads as 1, this bit reads as 1 to indicate that the secondary bus clock outputs will be stopped and driven low when this device is placed in $D3_{hot}$. This bit is not defined when the BPCC_En bit reads as 0.
23	BPCC_En	R	Bus Power/Clock Control Enable. When the **bpcc** pin is tied high, this bit reads as a 1 to indicate that the bus power/clock control mechanism is enabled, as described in B2_B3 (bit 23). When the **bpcc** pin is tied low, this bit reads as a 0 to indicate that the bus power/clock control mechanism is disabled (secondary clocks are not disabled when this device is placed in $D3_{hot}$).

13.1.35 Data Register — Offset E3h

This section describes the data register. (Implemented in the 21152-AB and later revisions only. In the 21152-AA, this register is reserved.)

Dword address = E0h

Byte enable **p_cbe_l<3:0>** = 0xxx

Dword Bit	Name	R/W	Description
31:24	Data	R	Data register. This register is not implemented and reads 00h.

13.2 Device-Specific Configuration Registers

This section provides a detailed description of the 21152 device-specific configuration registers.

Each field has a separate description.

Fields that have the same configuration address are selectable by turning on (driving low) the appropriate byte enable bits on **p_cbe_l** during the data phase. To select all fields of a configuration address, drive all byte enable bits low.

All reserved fields and registers are read only and always return 0.

13.2.1 Chip Control Register — Offset 40h

This section describes the chip control register.

Dword address = 40h

Byte enable **p_cbe_l<3:0>** = xxx0b

Dword Bit	Name	R/W	Description
0	Reserved	R	Reserved. Returns 0 when read.
1	Memory write disconnect control	R/W	Controls when the 21152, as a target, disconnects memory write transactions. When 0: The 21152 disconnects on queue full or on a 4KB boundary. When 1: The 21152 disconnects on a cache line boundary, as well as when the queue fills or on a 4KB boundary. Reset value: 0.
3:2	Reserved	R	Reserved. Returns 0 when read.
4	Secondary bus prefetch disable	R/W	Controls the 21152's ability to prefetch during upstream memory read transactions. When 0: The 21152 prefetches and does not forward byte enable bits during memory read transactions. When 1: The 21152 requests only one Dword from the target during memory read transactions and forwards read byte enable bits. The 21152 returns a target disconnect to the requesting master on the first data transfer. Memory read line and memory read multiple transactions are still prefetchable. Reset value: 0.
7:5	Reserved	R	Reserved. Returns 0 when read.

13.2.2 Diagnostic Control Register — Offset 41h

This section describes the diagnostic control register.

W1TR indicates that writing 1 in this bit position causes a chip reset to occur. Writing 0 has no effect.

Dword address = 40h

Byte enable **p_cbe_l<3:0>** = xx0xb

Dword Bit	Name	R/W	Description
8	Chip reset	R/W1TR	Chip and secondary bus reset control. When 1: Causes the 21152 to perform a chip reset. Data buffers, configuration registers, and both the primary and secondary interfaces are reset to their initial state. The 21152 clears this bit once chip reset is complete. The 21152 can then be reconfigured. Secondary bus reset **s_rst_l** is asserted and the secondary reset bit in the bridge control register is set when this bit is set. The secondary reset bit in the bridge control register must be cleared in order to deassert **s_rst_l**.
10:9	Test mode	R/W	Controls the testability of the 21152's internal counters. These bits are used for chip test only. The value of these bits controls which bytes of the counters are exercised: • 00b = Normal functionality — all bits are exercised. • 01b = Byte 1 is exercised. • 10b = Byte 2 is exercised. • 11b = Byte 0 is exercised. Reset value: 00b.
15:11	Reserved	R	Reserved. Returns 0 when read.

13.2.3 Arbiter Control Register — Offset 42h

This section describes the arbiter control register.

Dword address = 40h

Byte enable **p_cbe_l<3:0>** = 00xxb

Dword Bit	Name	R/W	Description
25:16	Arbiter control	R/W	Each bit controls whether a secondary bus master is assigned to the high priority arbiter group or the low priority arbiter group. Bits <19:16> correspond to request inputs **s_req_l<3:0>**, respectively. Bit <25> corresponds to the 21152 as a secondary bus master. When 0: Indicates that the master belongs to the low priority group. When 1: Indicates that the master belongs to the high priority group. Reset value: 10 0000 0000b.
31:26	Reserved	R	Reserved. Returns 0 when read.

13.2.4 p_serr_l Event Disable Register -— Offset 64h

This section describes the **p_serr_l** event disable register.

Dword address = 64h

Byte enable **p_cbe_l<3:0>** = xxx0b

Dword Bit	Name	R/W	Description
0	Reserved	R	Reserved. Returns 0 when read.
1	Posted write parity error	R/W	Controls the 21152's ability to assert **p_serr_l** when a data parity error is detected on the target bus during a posted write transaction. When 0: Signal **p_serr_l** is asserted if this event occurs and the SERR# enable bit in the command register is set. When 1: Signal **p_serr_l** is not asserted if this event occurs. Reset value: 0.
2	Posted write nondelivery	R/W	Controls the 21152's ability to assert **p_serr_l** when it is unable to deliver posted write data after 2^{24} attempts. When 0: Signal **p_serr_l** is asserted if this event occurs and the SERR# enable bit in the command register is set. When 1: Signal **p_serr_l** is not asserted if this event occurs. Reset value: 0.
3	Target abort during posted write	R/W	Controls the 21152's ability to assert **p_serr_l** when it receives a target abort when attempting to deliver posted write data. When 0: Signal **p_serr_l** is asserted if this event occurs and the SERR# enable bit in the command register is set. When 1: Signal **p_serr_l** is not asserted if this event occurs. Reset value: 0.
4	Master abort on posted write	R/W	Controls the 21152's ability to assert **p_serr_l** when it receives a master abort when attempting to deliver posted write data. When 0: Signal **p_serr_l** is asserted if this event occurs and the SERR# enable bit in the command register is set. When 1: Signal **p_serr_l** is not asserted if this event occurs. Reset value: 0.
5	Delayed write nondelivery	R/W	Controls the 21152's ability to assert **p_serr_l** when it is unable to deliver delayed write data after 2^{24} attempts. When 0: Signal **p_serr_l** is asserted if this event occurs and the SERR# enable bit in the command register is set. When 1: Signal **p_serr_l** is not asserted if this event occurs. Reset value: 0.
6	Delayed read—no data from target	R/W	Controls the 21152's ability to assert **p_serr_l** when it is unable to transfer any read data from the target after 2^{24} attempts. When 0: Signal **p_serr_l** is asserted if this event occurs and the SERR# enable bit in the command register is set. When 1: Signal **p_serr_l** is not asserted if this event occurs. Reset value: 0.
7	Reserved	R	Reserved. Returns 0 when read.

13.2.5 Secondary Clock Control Register — Offset 68h

This section describes the secondary clock control register.

Dword address = 68h

Byte enable **p_cbe_l<3:0>** = xx00b

Dword Bit	Name	R/W	Description
1:0	Clock 0 disable	R/W	If either bit is 0: Signal **s_clk_o<0>** is enabled. When both bits are 1: Signal **s_clk_o<0>** is disabled and driven high. Upon secondary bus reset, these bits are initialized to 0.
3:2	Clock 1 disable	R/W	If either bit is 0: Signal **s_clk_o<1>** is enabled. When both bits are 1: Signal **s_clk_o<1>** is disabled and driven high. Upon secondary bus reset, these bits are initialized to 0.
5:4	Clock 2 disable	R/W	If either bit is 0: Signal **s_clk_o<2>** is enabled. When both bits are 1: Signal **s_clk_o<2>** is disabled and driven high. Upon secondary bus reset, these bits are initialized to 0.
7:6	Clock 3 disable	R/W	If either bit is 0: Signal **s_clk_o<3>** is enabled. When both bits are 1: Signal **s_clk_o<3>** is disabled and driven high. Upon secondary bus reset, these bits are initialized to 0.
8	Clock 4 disable	R/W	When 0: Signal **s_clk_o<4>** is enabled. When 1: Signal **s_clk_o<4>** is disabled and driven high. Upon secondary bus reset, this bit is initialized to 0.
9:13	Reserved	R	Reserved. Returns 1 when read.

13.2.6 p_serr_l Status Register — Offset 6Ah

This section describes the **p_serr_l** status register.

This status register indicates the reason for the 21152's assertion of **p_serr_l**.

Dword address = 68h

Byte enable **p_cbe_l<3:0>** = x0xxb

Dword Bit	Name	R/W	Description
0	Address parity error	R/W1TC	When 1: Signal **p_serr_l** was asserted because an address parity error was detected on either the primary or secondary PCI bus. Reset value: 0.
1	Posted write data parity error	R/W1TC	When 1: Signal **p_serr_l** was asserted because a posted write data parity error was detected on the target bus. Reset value: 0.
2	Posted write nondelivery	R/W1TC	When 1: Signal **p_serr_l** was asserted because the 21152 was unable to deliver posted write data to the target after 2^{24} attempts. Reset value: 0.
3	Target abort during posted write	R/W1TC	When 1: Signal **p_serr_l** was asserted because the 21152 received a target abort when delivering posted write data. Reset value: 0.
4	Master abort during posted write	R/W1TC	When 1: Signal **p_serr_l** was asserted because the 21152 received a master abort when attempting to deliver posted write data. Reset value: 0.
5	Delayed write nondelivery	R/W1TC	When 1: Signal **p_serr_l** was asserted because the 21152 was unable to deliver delayed write data after 2^{24} attempts. Reset value: 0.
6	Delayed read—no data from target	R/W1TC	When 1: Signal **p_serr_l** was asserted because the 21152 was unable to read any data from the target after 2^{24} attempts. Reset value: 0.
7	Delayed transaction master time-out	R/W1TC	When 1: Signal **p_serr_l** was asserted because a master did not repeat a read or write transaction before the master time-out counter expired on the initiator's PCI bus. Reset to 0.

§ §

14.0 Diagnostic Nand Tree

The 21152 implements two pins for testing purposes:

- The **goz_l** pin, when asserted, tristates all bidirectional pins.
- The **nand_out** pin is the output of a serial Nand tree connecting all chip inputs except **p_clk** and **s_clk**. A pattern can be applied to chip inputs, and the **nand_out** pin verifies input pin interconnect.

Before the Nand tree test mechanism is used, all bidirectional signals must first be tristated by assertion of **goz_l**. Signal **goz_l** should remain asserted for the duration of the test.

Note: Any inputs tied high or low (for example, **s_cfn_l**), should be connected to power or ground through a resistive device to allow the Nand tree test mechanism to be used.

The Nand tree begins at the **s_cfn_l** input, runs clockwise to **p_rst_l**, and then is output at **nand_out**.

Intel recommends the following Nand tree test sequence:

1. Drive **goz_l** low.
2. Drive each input and bidirectional pin high, with the possible exception of **p_clk** and **s_clk.** (Signals **p_clk** and **s_clk** are not included in the Nand tree.)
3. Starting with pin 49 (**s_cfn_l**), proceed clockwise and individually drive each pin low; **nand_out** should toggle with each pin.
4. Turn off tester drivers.
5. Drive **goz_l** high.
6. Reset chip before proceeding with further testing.

§ §

15.0 Electrical Specifications

This chapter specifies the following electrical behavior of the 21152:

- PCI electrical conformance
- Absolute maximum ratings
- dc specifications
- ac timing specifications

15.1 PCI Electrical Specification Conformance

The 21152 PCI pins conform to the basic set of PCI electrical specifications in the *PCI Local Bus Specification, Revision 2.1*. Refer to that document for a complete description of the PCI I/O protocol and pin ac specifications.

15.2 Absolute Maximum Ratings

The 21152 is specified to operate at a maximum frequency of 33 MHz at a junction temperature (T_j) not to exceed 125°C. Table 33 lists the absolute maximum ratings for the 21152. Stressing the device beyond the absolute maximum ratings may cause permanent damage. These are stress ratings only. Operating beyond the functional operating range is not recommended, and extended exposure beyond the functional operating range may affect the reliability.

Table 33. Absolute Maximum Ratings

Parameter	Minimum	Maximum
T_j	—	125°C
Supply voltage, V_{cc}	—	3.9 V
Maximum voltage applied to signal pins	—	5.5 V
Maximum power, P_{WC}	—	1.2 W @ 33 MHz
Storage temperature range, T_{stg}	–55°C	125°C

Table 34 lists the functional operating range.

Table 34. Functional Operating Range

Parameter	Minimum	Maximum
Supply voltage, V_{cc}	3.0 V	3.6 V
Operating ambient temperature, T_a	0°C	70°C

15.3 DC Specifications

Table 35 defines the dc parameters met by all 21152 signals under the conditions of the functional operating range.

Table 35. DC Parameters

Symbol	Parameter	Condition	Minimum	Maximum	Unit
V_{cc}	Supply voltage	—	3.0	3.6	V
V_{il}	Low-level input voltage[1]	—	–0.5	0.3 V_{cc}	V
V_{ih}	High-level input voltage[1]	—	0.5 V_{cc}	V_{cc} + 0.5 V	V
V_{ol}	Low-level output voltage[2]	I_{out} = 1500 µA	—	0.1 V_{cc}	V
V_{ol5V}	Low-level output voltage[3]	I_{out} = 6 mA	—	0.55	V
V_{oh}	High-level output voltage[2]	I_{out} = –500 µA	0.9 V_{cc}	—	V
V_{oh5V}	High-level output voltage[3]	I_{out} = –2 mA	2.4	—	V
I_{il}	Low-level input leakage current[1,4]	0 < V_{in} < V_{cc}	—	±10	µA
C_{in}	Input pin capacitance	—	—	10.0	pF
C_{IDSEL}	**p_idsel** pin capacitance	—	—	8.0	pF
C_{clk}	**p_clk, s_clk** pin capacitance	—	5.0	12.0	pF

1. Guarantees meeting the specification for the 5-V signaling environment.
2. For 3.3-V signaling environment.
3. For 5-V signaling environment.
4. Input leakage currents include high-Z output leakage for all bidirectional buffers with tristate outputs.

Note: In Table 35, currents into the chip (chip sinking) are denoted as positive (+) current. Currents from the chip (chip sourcing) are denoted as negative (–) current.

15.4 AC Timing Specifications

The next sections specify the following:

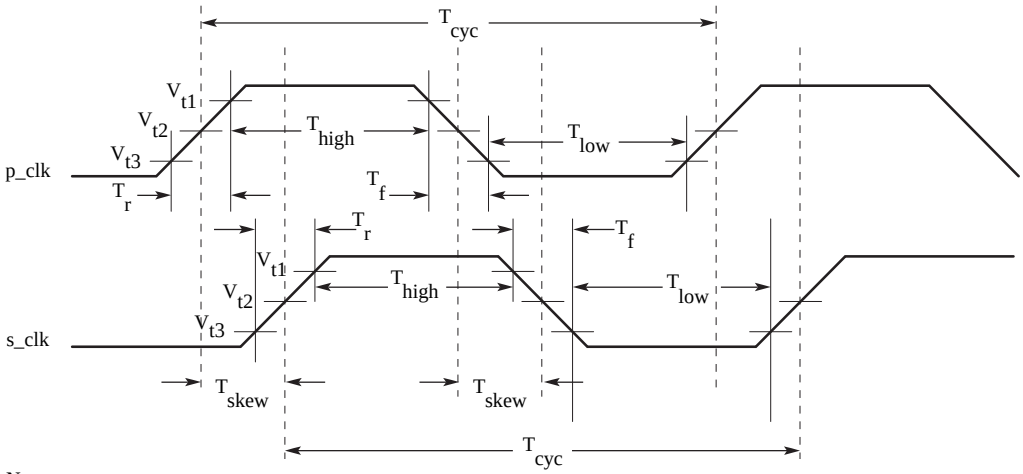
- Clock timing specifications
- PCI signal timing specifications
- Reset timing specifications

15.4.1 Clock Timing Specifications

The ac specifications consist of input requirements and output responses. The input requirements consist of setup and hold times, pulse widths, and high and low times. Output responses are delays from clock to signal. The ac specifications are defined separately for each clock domain within the 21152.

Figure 21 shows the ac parameter measurements for the **p_clk** and **s_clk** signals, and Table 36 specifies **p_clk** and **s_clk** parameter values for clock signal ac timing. See also Figure 22 for a further illustration of signal timing. Unless otherwise indicated, all ac parameters are guaranteed when tested within the functional operating range of Table 34.

Figure 21. PCI Clock Signal AC Parameter Measurements



Note:

V_{t1} – 2.0 V for 5-V clocks; 0.5 V_{cc} for 3.3-V clocks

V_{t2} – 1.5 V for 5-V clocks; 0.4 V_{cc} for 3.3-V clocks

V_{t3} – 0.8 V for 5-V clocks; 0.3 V_{cc} for 3.3-V clocks

LJ-04738.AI4

Table 36. PCI Clock Signal AC Parameters

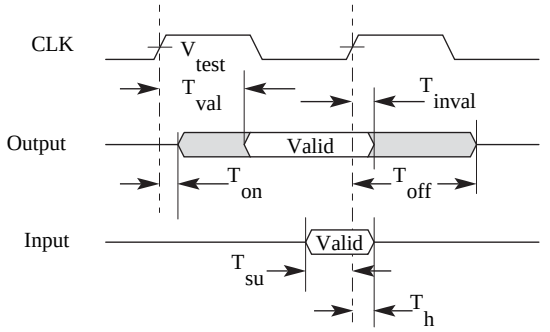
Symbol	Parameter	Minimum	Maximum	Unit
T_{cyc}	**p_clk**,**s_clk** cycle time	30	Infinity	ns
T_{high}	**p_clk**, **s_clk** high time	11	—	ns
T_{low}	**p_clk**, **s_clk** low time	11	—	ns
	p_clk, **s_clk** slew rate[1]	1	4	V/ns
T_{sclk}	Delay from **p_clk** to **s_clk**	0	7	ns
T_{sclkr}	**p_clk** rising to **s_clk_o** rising	0	5	ns
T_{sclkf}	**p_clk** falling to **s_clk_o** falling[2]	0	5	ns
T_{dskew}	**s_clk_0** duty cycle skew from **p_clk** duty cycle[2]	—	0.750	ns
T_{skew}	**s_clk_0<x>** to **s_clk_0<y>**	—	0.500	ns

1. 0.2 V_{cc} to 0.6 V_{cc}.
2. Measured with 30-pF lumped load.

15.4.2 PCI Signal Timing Specifications

Figure 22 and Table 37 show the PCI signal timing specifications.

Figure 22. PCI Signal Timing Measurement Conditions



Note:
V_{test} – 1.5 V for 5-V signals; 0.4 V_{cc} for 3.3-V signals

LJ-04739.AI4

Table 37. PCI Signal Timing

Symbol	Parameter	Minimum	Maximum	Unit
T_{val}	CLK to signal valid delay — bused signals[1,2,3]	2	11	ns
$T_{val(ptp)}$	CLK to signal valid delay — point-to-point	2	12	ns
T_{on}	Float to active delay	2	—	ns
T_{off}	Active to float delay	—	28	ns
T_{su}	Input setup time to CLK — bused signals	7	—	ns
$T_{su(ptp)}$	Input setup time to CLK—point-to-point	10, 12	—	ns
T_{h}	Input signal hold time from CLK	0	—	ns

1. See Figure 22.
2. All primary interface signals are synchronized to **p_clk**. All secondary interface signals are synchronized to **s_clk**.
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, s_stop_l.**

15.4.3 Reset Timing Specifications

Table 38 shows the reset timing specifications for **p_rst_l** and **s_rst_l**.

Table 38. Reset Timing Specifications

Symbol	Parameter	Minimum	Maximum	Unit
T_{rst}	**p_rst_l** active time after power stable	1	—	µs
$T_{rst-clk}$	**p_rst_l** active time after **p_clk** stable	100	—	µs
$T_{rst-off}$	**p_rst_l** active-to-output float delay	—	40	ns
T_{srst}	**s_rst_l** active after **p_rst_l** assertion	—	40	ns
$T_{srst-on}$	**s_rst_l** active time after **s_clk** stable	100	—	µs
T_{dsrst}	**s_rst_l** deassertion after **p_rst_l** deassertion	—	3	Cycles
	p_rst_l slew rate[1]	50	—	mV/ns

1. Applies to rising (deasserting) edge only.

§ §

16.0 Mechanical Specifications

The 21152 is contained in an industry-standard 160-pin plastic quad flat pack (PQFP) package, shown in Figure 23.

Figure 23. 160-Pin PQFP Package

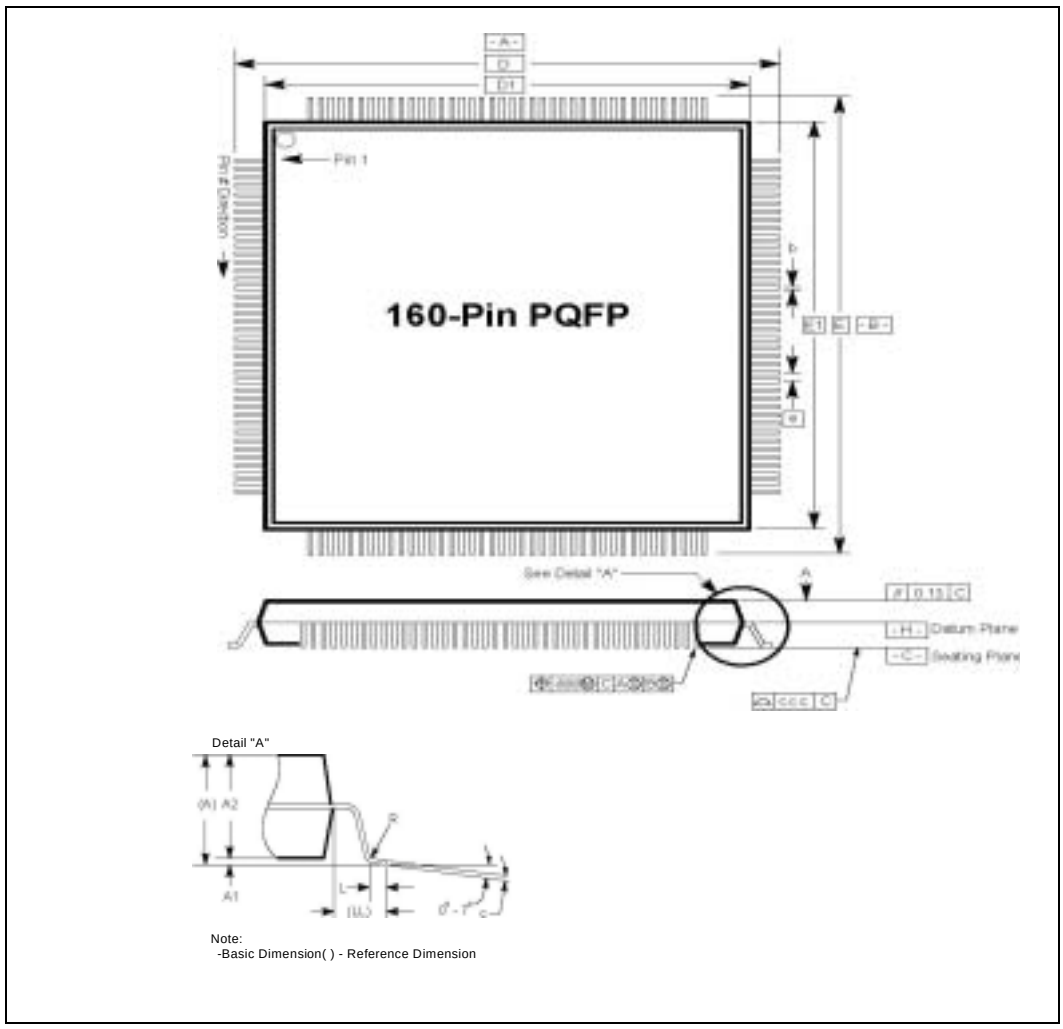


Table 39 lists the 160-pin package dimensions in millimeters.

Table 39. 160-Pin PQFP Package Dimensions

Symbol	Dimension	Value (mm)
LL	Lead length	1.30 reference[1]
e	Lead pitch	0.65 BSC[2]
L	Foot length	0.65 minimum to 1.03 maximum
A	Package overall height	4.50
A1	Package standoff height	0.25 minimum
A2	Package thickness	3.17 minimum to 3.67 maximum
b	Lead width	0.22 minimum to 0.38 maximum
c	Lead thickness	0.12 minimum to 0.23 maximum
ccc	Coplanarity	0.10
ddd	Lead skew	0.13
D	Package overall width	31.20 BSC[2]
D1	Package width	28.00 BSC[2]
E	Package overall length	31.20 BSC[2]
E1	Package length	28.00 BSC[2]
R	Ankle radius	0.13 minimum to 0.30 maximum

1. The value for this measurement is for reference only.
2. ANSI Y14.5M-1982 American National Standard Dimensioning and Tolerancing, Section 1.3.2, defines Basic Dimension (BSC) as: A numerical value used to describe the theoretically exact size, profile, orientation, or location of a feature or datum target. It is the basis from which permissible variations are established by tolerances on other dimensions, in notes, or in feature control frames.

§ §

17.0 Configuration Register Values After Reset

Table 40 lists the value of the 21152 configuration registers after reset. Reserved registers are not listed and are always read only as 0.

Table 40. Configuration Register Values After Reset *(Sheet 1 of 2)*

Byte Address	Register Name	Reset Value
00–01h	Vendor ID	1011h
02–03h	Device ID	0024h
04–05h	Command	0000h
06–07h	Status	0280h[1] 0290h[2]
08h	Revision ID	Initially 00h[3]
09–0Bh	Class code	060400h
0Ch	Cache line	00h
0Dh	Primary master latency timer	00h
0Eh	Header type	01h
18h	Primary bus number	00h
19h	Secondary bus number	00h
1Ah	Subordinate bus number	00h
1Bh	Secondary master latency timer	00h
1Ch	I/O base	01h
1Dh	I/O limit	01h
1E–1Fh	Secondary status	0280h
20–21h	Memory-mapped I/O base	0000h
22–23h	Memory-mapped I/O limit	0000h
24–25h	Prefetchable memory base	0001h
26–27h	Prefetchable memory limit	0001h
28–2Bh	Prefetchable memory base upper 32 bits	00000000h
2C–2Fh	Prefetchable memory limit upper 32 bits	00000000h
30–31h	I/O base upper 16 bits	0000h
32–33h	I/O limit upper 16 bits	0000h
34–35h	Subsystem vendor ID ECP pointer	0000h[4] 00DCh[2]
36–37h	Subsystem ID	0000h
3Dh	Interrupt pin	00h
3E–3Fh	Bridge control	0000h
40h	Chip control	00h
41h	Diagnostic control	00h
42–43h	Arbiter control	0200h

Table 40. Configuration Register Values After Reset *(Sheet 2 of 2)*

Byte Address	Register Name	Reset Value
64h	**p_serr_l** event disable	00h
68–69h	Secondary clock control	00h
6Ah	**p_serr_l** status	00h
DCh	Power management capability ID	01h[5]
DDh	Next item	00h[5]
DE–DFh	Power management capabilities	0001h[5]
E0–E1h	Power management CSR	0000h[5]
E2h	PPB support extensions	C0h (**bpcc** = 1)[5] 00h (**bpcc** = 0)
E3h	Data register	00h[5]

1. 21152–AA.
2. 21152–AB and later revisions.
3. Dependent on revision of device. The first revision is read as 00h; subsequent revisions increment by 1.
4. 21152–AA only. In the 21152–AB, these registers are reserved.
5. 21152–AB and later revisions only. In the 21152–AA, these registers are reserved.

§ §