PCI Express™ to PCI/PCI-X Bridge Specification Revision 1.0

July 14, 2003

REVISION	REVISION HISTORY	DATE
1.0	Initial release	7/14/03

PCI-SIG disclaims all warranties and liability for the use of this document and the information contained herein and assumes no responsibility for any errors that may appear in this document, nor does PCI-SIG make a commitment to update the information contained herein.

Contact the PCI-SIG office to obtain the latest revision of the specification.

Questions regarding the PCI Express to PCI/PCI-X Bridge Specification or membership in PCI-SIG may be forwarded to:

Membership Services
www.pcisig.com
E-mail: administration@pcisig.com
Phone: 1-800-433-5177 (Domestic Only)
503-291-2569
Fax: 503-297-1090

Technical Support
techsupp@pcisig.com

DISCLAIMER

This PCI Express to PCI/PCI-X Bridge Specification is provided "as is" with no warranties whatsoever, including any warranty of merchantability, noninfringement, fitness for any particular purpose, or any warranty otherwise arising out of any proposal, specification, or sample. PCI-SIG disclaims all liability for infringement of proprietary rights, relating to use of information in this specification. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted herein.

Contents

Figures

Tables

Objective of This Specification

As PCI Express technology deploys into computer marketplaces, a means of preserving and extending existing industry investments in conventional PCI and PCI-X is required. Additionally, some add-in card manufacturers may desire a means of developing PCI Express Endpoint application devices that plug into platforms that exclusively support conventional PCI and PCI-X system slots.

This specification describes the PCI Express to PCI/PCI-X bridge (also referred to herein as PCI Express bridge) architecture, interface requirements, and the programming model. A device compliant with this specification supports an industry-standard method to bridge from a PCI Express Base 1.0a-compliant primary interface to a PCI 3.0- or, additionally, a PCI-X 2.0a-compliant secondary interface. Bridge implementations that meet the requirements presented in this specification will possess compatibility with system software developed to work with the Type 01h configuration headers described in PCI 3.0 and PCI-X PT 2.0a. Additionally, the Capabilities List items that apply to PCI bridges and PCI-X bridges may be used in conjunction with PCI Express bridges without modification.

In addition, this document provides guidelines for PCI Express reverse bridges (i.e., a bridge from PCI/PCI-X primary to PCI Express secondary interface). These guidelines are presented in Appendix A for the purpose of aiding those interested in developing such components but should not be considered as architectural requirements or as being exhaustive in scope.

The highlights of the PCI Express bridge architecture include:

- Support for a PCI Express 1.0a-compliant primary interface
- Support for one or more PCI 3.0-compliant interfaces, with optional PCI-X 2.0a support (both Mode 1 and Mode 2)
- Error detection and forwarding between interfaces
- Optional message forwarding
- Support for extended configuration space transaction forwarding

Reference Documents

PCI Express Base Specification, Rev. 1.0a (PCI Express Base 1.0a)

PCI Express Card Electromechanical Specification, Rev. 1.0 (PCI Express CEM 1.0)

PCI Local Bus Specification, Rev. 3.0 (PCI 3.0)

PCI-X Protocol Addendum to the PCI Local Bus Specification, Rev. 2.0a (PCI-X PT 2.0a)

PCI-X Electrical and Mechanical Addendum to the PCI Local Bus Specification, Rev. 2.0a (PCI-X EM 2.0a)

(PCI-X PT 2.0a and PCI-X EM 2.0a are collectively referred to as PCI-X 2.0a)

PCI Hot-Plug Specification, Rev. 1.1 (PCI HP 1.1)

PCI Standard Hot-Plug Controller and Subsystem Specification, Rev. 1.0 (SHPC 1.0)

PCI-to-PCI Bridge Architecture Specification, Rev. 1.2 (PCI Bridge 1.2)

PCI Power Management Interface Specification, Rev. 1.1 (PCI PM 1.1)

PCI BIOS Specification, Rev. 2.1 (PCI BIOS 2.1)

Documentation Conventions

Capitalization

Some terms are capitalized to distinguish their definition in the context of this document from their common English meaning. Words not capitalized have their common English meaning. When terms such as "memory write" or "memory read" appear completely in lower case, they include all transactions of that type.

Register names and the names of fields and bits in registers and headers are presented with the first letter capitalized and the remainder in lower case.

Numbers and Number Bases

Hexadecimal numbers are written with a lower case "h" suffix, e.g., FFFh and 80h. Hexadecimal numbers larger than four digits are represented with a space dividing each group of four digits, as in 1E FFFF FFFFh. Binary numbers are written with a lower case "b" suffix, e.g., 1001b and 10b. Binary numbers larger than four digits are written with a space dividing each group of four digits, as in 1000 0101 0010b.

All other numbers are decimal.

Implementation Notes

Implementation Notes should not be considered to be part of this specification. They are included for clarification and illustration only.

Terms and Acronyms

address order	Incrementing sequentially, beginning with the starting address of the Sequence. For example, Split Completions in the same Sequence (that is, resulting from a single Split Request) must be returned in address order.
advertise (Credits)	For PCI Express, used in the context of Flow Control, the act of a Receiver sending information regarding its Flow Control Credit availability.
ADB Delimited Quantum, ADQ	A portion (or all) of a PCI-X transaction or a buffer that fits between two adjacent ADBs. For example, if a transaction starts between two ADBs, crosses one ADB, and ends before reaching the next ADB, the transaction includes two ADQs of data. Such a transaction fits in two buffers inside a device that divides its buffers on ADBs.
Allowable Disconnect Boundary, ADB	For PCI-X, a naturally aligned 128-byte boundary. PCI-X Initiators and targets are permitted to disconnect burst transactions only on ADBs.
asserted	The active logical state of a conceptual or actual signal.
attribute	Transaction handling preferences indicated in PCI Express by specified Packet header bits and fields (for example, No Snoop). For PCI-X, attribute refers to the 36-bit field driven on the bus during the attribute phase(s) of a PCI-X transaction.
attribute phase	The clock after the address phase(s) in a PCI-X transaction.
application bridge	A PCI-X device that implements internal posting of memory write transactions that the device must initiate on the PCI-X interface but uses a Type 00h Configuration Space header and the Class Code of the application it performs rather than that of a bridge. This specification and PCI Express Base 1.0a do not include provisions for application bridges.
bridge	A device that virtually or actually connects a conventional PCI/PCI-X segment or PCI Express Port with an internal component interconnect or with another conventional PCI/PCI-X segment or PCI Express Port.
burst push transaction	A burst transaction on PCI-X for which the initiator is the source of the data. Burst push transactions use one of the following commands: Memory Write Memory Write Block Alias to Memory Write Block Split Completion Device ID Message In PCI-X Mode 2, all burst push transactions except Memory Write transactions use source-synchronous clocking. No other transactions use source-synchronous clocking.

burst transaction

A transaction on PCI-X that uses one of the following commands:

Memory Read Block
Memory Write Block
Memory Write
Alias to Memory Read Block
Alias to Memory Write Block
Split Completion
Device ID Message

Burst transactions can generally be of any length, from 1 to 4096 bytes. (Note that if the byte count is small enough, a burst transaction contains only a single data phase.) On 64-bit buses, they are permitted to be initiated both as 64-bit and 32-bit transactions.

byte count — The number of bytes to be included in a Sequence on PCI-X. It appears in the attribute phase of all burst transactions and indicates the number of bytes affected by the transaction.

cold reset — For PCI Express, a Fundamental Reset following the application of power.

Completer — The logical device addressed by a Request.

Completer ID — The combination of a Completer's Bus Number, Device Number, and Function Number that uniquely identifies the Completer of the Request.

Completion — A Packet used to terminate, or to partially terminate, a transaction sequence. A *Completion* always corresponds to a preceding Request, and, in some cases, includes data.

component — A physical device (a single package).

Configuration Space — One of the four address spaces within the PCI architecture family. Packets with a *Configuration Space* address are used to configure a device.

conventional PCI — Behavior or features that conform to PCI 3.0.

correctable ECC error — For PCI-X, an error interpreted by the ECC mechanism as the result of the inversion of a single bit.

Data Link Layer — For PCI Express, the intermediate Layer that is between the Transaction Layer and the Physical Layer.

Data Link Layer Packet, DLLP — A PCI Express Packet generated in the Data Link Layer to support Link management functions.

data payload — Information following the header in some PCI Express packets that is destined for consumption by the logical device receiving the Packet (for example, Write Requests or Read Completions).

data phase — For conventional PCI and PCI-X, each clock in which the target signals some kind of data transfer or terminates the transaction. Clocks in which the target signals Wait State one or more times and then signals something else are part of the same data phase.

deasserted — The inactive logical state of a conceptual or actual signal.

device — A logical device corresponding to a PCI device configuration space. May be either a single or multi-function device.

device boundary	The first address of a device range or the first address beyond the end of a device range. Address ranges for devices with Type 00h Configuration Space headers are established with Base Address registers, as described in PCI 3.0. Address ranges for devices with Type 01h Configuration Space headers (bridges) are established with Base Address, Memory Base, I/O Base, and Prefetchable Memory Base registers, as described in PCI Bridge 1.2. In the terms of PCI-X, a device boundary is always an ADB. To “disconnect at a device boundary” means to disconnect a transaction in such a way that the last address of the transaction corresponds to the last address of the device. To “cross a device boundary” means that the transaction includes one or more addresses of the devices on both sides of the boundary.
Device ID Message	A Sequence on PCI-X that uses the Device ID Message command. Device ID message transactions are burst transactions that use explicit routing (i.e., the Completer ID – the completer Bus Number, completer Device Number, and completer Function Number) or implicit routing (described in PCI-X PT 2.0a) to address the completer. They are intended for direct peer-to-peer communication between devices.
Downstream	Downstream refers either to the relative position of an interconnect/system element (Link/device) as something that is farther from the Root Complex or to a direction of information flow; i.e., when information is flowing away from the Root Complex. The Ports on a Switch which are not the Upstream Port are Downstream Ports. All Ports on a Root Complex are Downstream Ports. The secondary interface of a bridge is Downstream of the primary interface. The Downstream component on a Link is the component farther from the Root Complex.
DWORD, DW	Four bytes. Used in the context of a data payload, the four bytes of data must be on a naturally aligned 4-byte boundary (the least significant two bits of the byte address are 00b).
DWORD transaction	A transaction on PCI-X that uses one of the following commands: Interrupt Acknowledge Special Cycle I/O Read I/O Write Configuration Read Configuration Write Memory Read DWORD DWORD transactions address no more than a single DWORD and on 64- and 32-bit buses are permitted to be initiated only as 32-bit transactions (REQ64# must be deasserted). During the attribute phase, the Requester Attributes contain valid byte enables. During the data phase, theC/BE# bus is reserved and driven high by the initiator.
ECC	For PCI-X, an error correcting code. In ECC mode, each address, attribute, common-clock data phase, and source-synchronous data subphase include additional bits that are used to correct, if enabled, any single bit errors and detect any double bit errors.
ECC mode	For PCI-X, the mode of operation of the bus that uses ECC error protection. A bus operating in PCI-X Mode 1 operates either in parity mode or ECC mode, as determined by a bit in the ECC Control and Status register. A bus operating in PCI-X Mode 2 always operates in ECC mode.
Endpoint	A PCI Express device with a Type 00h Configuration Space header.

ending address	For PCI-X, the last address included in the Sequence. For burst transactions, it is calculated by adding the starting address and the byte count and subtracting one and is permitted to be aligned to any byte. For DWORD transactions, it is the last byte (AD[1:0] = 11b) of the DWORD addressed by the starting address.
Error Recovery, Error Detection	The mechanisms that ensure integrity of data transfer on PCI Express, including the management of the transmit side retry buffer(s).
float	When a PCI device has finished driving a bus or a control signal and it places the output buffers in the high-impedance state, the device is said to float the bus or the control signal.
Flow Control	The method for communicating PCI Express receive buffer status from a Receiver to a Transmitter to prevent receive buffer overflow and allow Transmitter compliance with ordering rules.
FCP or Flow Control Packet	A PCI Express DLLP used to send Flow Control information from the Transaction Layer in one component to the Transaction Layer in another component.
function	A logical function corresponding to a PCI function configuration space. May be used to refer to one function of a multi-function device or to the only function in a single-function device.
header	A set of fields that appear at the front of a Packet that contain the information required to determine the characteristics and purpose of the Packet.
Hierarchy	The tree structured PCI Express I/O interconnect topology.
hierarchy domain	The part of a Hierarchy originating from a single Root Port.
Host Bridge	The part of a Root Complex that connects a host CPU or CPUs to a Hierarchy.
hot reset	A reset propagated in-band across a Link using a Physical Layer mechanism.
Immediate Transaction	A PCI-X transaction that terminates in a way that includes transferring data or that terminates with an error that completes the Sequence. Transactions in which the target signals Data Transfer, Single Data Phase Disconnect, Disconnect at Next ADB, Master-Abort, or Target-Abort are Immediate Transactions. Transactions that terminate with Retry or Split Response are not Immediate Transactions.
I/O Space	One of the four address spaces of the PCI Express architecture. Identical to the I/O space defined in PCI 3.0.
isochronous	Data associated with time-sensitive applications, such as audio or video applications.
Layer	A unit of distinction applied to this specification to help clarify the behavior of key elements. The use of the term *Layer* does not imply a specific implementation.
Link	The collection of two Ports and their interconnecting Lanes. A *Link* is a dual-simplex communications path between two components.
Logical Bus	The logical connection among a collection of devices that have the same Bus Number in Configuration Space.
logical device	An element of a PCI Express system that responds to a unique device number in Configuration Space. *Logical devices* are either a single function or multi-function devices. *Logical device* requirements apply to both single function logical devices as well as to each function individually of a multi-function *logical device*.

Malformed Packet	A TLP that violates specific TLP formation rules as defined in this specification and in PCI Express Base 1.0a.
Message	A TLP used to communicate information outside of the Memory, I/O, and Configuration spaces.
Message Signaled Interrupt, MSI/MSI-X	Two similar but separate mechanisms that enable a device to request service by writing a system-specified DWORD of data to a system-specified address using a Memory Write Request. MSI is the original mechanism, first defined in PCI 2.2. MSI-X is a separate extension mechanism included in PCI 3.0. Compared to MSI, MSI-X supports a larger maximum number of vectors and independent message address/data for each vector.
Message Space	One of the four address spaces of the PCI Express architecture.
naturally aligned	A data payload with a starting address equal to an integer multiple of a power of two, usually a specific power of two. For example, 64 bytes *naturally aligned* means the least significant 6 bits of the byte address are 00 0000b.
Packet	A fundamental unit of information transfer on PCI Express consisting of a header that, in some cases, is followed by a data payload.
parity mode	The mode of operation of a PCI bus that uses parity error protection. A bus operating in PCI-X Mode 1 operates either in parity mode or ECC mode, as determined by a bit in the ECC Control and Status register. A bus operating in PCI-X Mode 2 never operates in parity mode.
PCI bus	The PCI Local Bus, as specified in the PCI 3.0 and PCI-X 2.0a specifications.
PCI Software Model	The software model necessary to initialize, discover, configure, and use a PCI device, as specified in PCI 3.0, PCI-X 2.0a, and PCI BIOS 2.1 specifications.
PCI-X initialization pattern	A combination of bus control signals that is used to place PCI-X devices in PCI-X Mode 1 (with parity or ECC protection) or Mode 2 at the rising edge of RST#. Also indicates the range of frequency of the clock.
PCI-X Mode 1	A mode of operation compliant with revisions of PCI-X prior to PCI-X 2.0a. It includes: PCI-X 66 and PCI-X 133 modes Data transfers always use common clock 3.3V signaling levels Optional ECC (added in PCI-X 2.0a) Devices compliant with the *PCI-X Addendum to the PCI Local Bus Specification, Rev. 1.0,* are automatically compliant with PCI-X 2.0a in PCI-X Mode 1. Sometimes abbreviated as “Mode 1” when the meaning is clear from the context. When used as an adjective, (for example, “Mode 1 device,” “Mode 1 add-in card,” “Mode 1 system,” or “Mode 1 slot”) it defines an object whose highest operating mode is either PCI-X 66 or PCI-X 133, whether or not that object is actually operating in PCI-X Mode 1.

PCI-X Mode 2	A mode of operation compliant with PCI-X 2.0a that includes: PCI-X 266 and PCI-X 533 modes: Burst push transactions other than Memory Write use source-synchronous clocking 1.5V signaling levels for source-synchronous signals ECC All features not defined in revisions of PCI-X prior to PCI-X 2.0a apply in PCI-X Mode 2. (Some features are optionally available in Mode 1 also.) Sometimes abbreviated as “Mode 2” when the meaning is clear from the context. When used as an adjective, (for example, “Mode 2 device,” “Mode 2 add-in card,” “Mode 2 system,” or “Mode 2 slot”) it defines an object whose highest operating mode is either PCI-X 266 or PCI-X 533, whether or not that object is actually operating in PCI-X Mode 2.
Physical Layer	The Layer of the PCI Express architecture that directly interacts with the communication medium between two components.
Port	Logically, an interface between a component and a PCI Express Link. Physically, a group of transmitters and receivers located on the same chip that define a Link.
QWORD, QW	Eight bytes. Used in the context of a data payload, the 8 bytes of data must be on a naturally aligned 8-byte boundary (the least significant three bits of the address are 000b).
read side effects	Changes to the state of a device that occur if a location within the device is read, for example, the clearing of a status bit or advancing to the next data value in a sequential buffer.
Receiver	The component that receives Packet information across a Link.
Receiving Port	In the context of a specific TLP or DLLP, the Port that receives the Packet on a given Link.
reserved	The contents, states, or information are not defined at this time. Using any *reserved* area (for example, Packet header bit-fields, configuration register bits) is not permitted. Reserved register fields must be read-only and must return 0 when read. Reserved encodings for register and packet fields must not be used. Any implementation dependence on a *reserved* field value or encoding will result in an implementation that is not PCI Express-compliant. The functionality of such an implementation cannot be guaranteed in this or any future revision of this specification.
Request	A Packet used to initiate a transaction sequence. A *Request* includes operation code and, in some cases, address and length, data, or other information.
Requester	A logical device (an initiator) that first introduces a transaction sequence into the PCI Express or PCI-X domain.
Requester ID	The combination of a Requester's Bus Number, Device Number, and Function Number that uniquely identifies the Requester.
Root Complex	An entity that includes a Host Bridge and one or more Root Ports.
Root Port	A PCI Express Port on a Root Complex that maps a portion of the Hierarchy through an associated virtual PCI-to-PCI bridge.
Sequence	One or more PCI-X transactions associated with carrying out a single logical transfer by a requester.

Sequence ID	For PCI-X: The combination of the Requester ID (requester Bus Number, Device Number, and Function number) and Tag attributes. This combination uniquely identifies transactions that are part of the same Sequence and is used in buffer-management algorithms and some ordering rules.
Split Completion	When used in the context of the PCI-X bus protocol, this term refers to a transaction using the Split Completion command. It is used by the completer to send the requested data (for read transactions completed without error) or a completion message back to the requester. When used in the context of transaction ordering and the transaction queues inside the requester, completer, and bridges, the term refers to a queue entry corresponding to a Split Completion transaction on the bus.
Split Completion Message	In the context of PCI-X bus transactions, a Split Completion Message is a Split Completion transaction that notifies the requester that a request (either read or write) encountered an error, or that a write request completed without an error. In the context of the Split Completion address, the term refers to the attribute bit that indicates the Split Completion is a message rather than data for a read request.
Split Request	When used in the context of the PCI-X bus protocol, this term refers to a transaction terminated with Split Response. When used in the context of transaction ordering and the transaction queues inside the requester, completer, and bridges, the term refers to a queue entry corresponding to a Split Request transaction on the bus. When the completer executes the Split Request, it becomes a Split Completion.
Split Response	The PCI-X protocol for terminating a transaction, whereby the target indicates that it will complete the transaction as a Split Transaction. The target may optionally terminate any DWORD transaction (except Special Cycle) and any read transaction with Split Response.
Split Transaction	A single logical transfer on PCI-X containing an initial transaction (the Split Request) that the target (the completer or a bridge) terminates with Split Response, followed by one or more transactions (the Split Completions) initiated by the completer (or bridge) to send the read data (if a read) or a completion message back to the requester.
Standard Hot-Plug Controller, SHPC	A PCI hot-plug controller compliant with SHPC 1.0.
starting address	Address indicated in the address phase of all PCI transactions except Split Completions, Interrupt Acknowledges, Special Cycles, and Device ID Messages. For conventional PCI, this is the first DWORD affected by the transaction and all addresses are DWORD-aligned. For PCI-X, this is the first byte affected by the transaction. The starting address of all PCI-X transactions except configuration transactions is permitted to be aligned to any byte (i.e., it uses the full address bus). The starting address of configuration transactions must be aligned to a DWORD boundary (i.e., AD[1:0] must be 00b).
Switch	A system element that connects two or more Ports to allow Packets to be routed from one Port to another. To configuration software, a *Switch* appears as a collection of virtual PCI-to-PCI Bridges.

Tag	For PCI Express, a 5-bit or 8-bit number assigned to a given non-posted Request to distinguish Completions for that Request from other Requests. For PCI-X, a 5-bit number assigned by the initiator of a Sequence to distinguish it from other Sequences. It appears in the attribute field and is part of the Sequence ID. An initiator must not reuse a Tag for a new Sequence until the original Sequence is complete (i.e., byte count satisfied, error condition encountered, etc.). See Chapter 2 of PCI-X PT 2.0a for details.
Transaction Descriptor	An element of a Packet header that, in addition to Address, Length, and Type, describes the properties of the Transaction.
target	A PCI device that responds to bus transactions. A bridge forwarding a transaction is the target on the originating bus.
Transaction Layer	The Layer that operates at the level of transactions (for example, read, write).
Transaction Layer Packet, TLP	A Packet generated in the Transaction Layer to convey a Request or Completion.
transaction sequence	A single Request and zero or more Completions associated with carrying out a single logical transfer by a Requester.
Transmitter	The component sending Packet information across a Link.
Transmitting Port	In the context of a specific TLP or DLLP, the Port that transmits the Packet on a given Link.
uncorrectable error	A parity error in PCI-X parity mode (and conventional PCI) or a detected multiple-bit ECC error in ECC mode. (Correctable ECC errors are also treated as uncorrectable if error correction is disabled.)
Unsupported Request (UR)	A Request Packet that specifies some action or access to some space that is not supported by the Target.

Upstream Upstream refers either to the relative position of an interconnect/system element (Link/device) as something that is closer to the Root Complex, or to a direction of information flow; i.e., when information is flowing towards the Root Complex. The Port on a Switch which is closest topologically to the Root Complex is the Upstream Port. The primary interface of a bridge is Upstream of the secondary interface. The Upstream component on a Link is the component closer to the Root Complex.

warm reset	A Fundamental Reset performed without cycling the supplied power.

1

1. Introduction

This chapter presents an overview of the key requirements of the PCI Express bridge architecture. It also highlights bridge topologies addressed by this document and introduces some important concepts.

Figure 1-1 illustrates an example of PCI Express-based system that includes a PCI Express bridge. Note that multiple PCI Express bridges can be integrated into the system to satisfy specific PCI slot support and fan-out requirements.

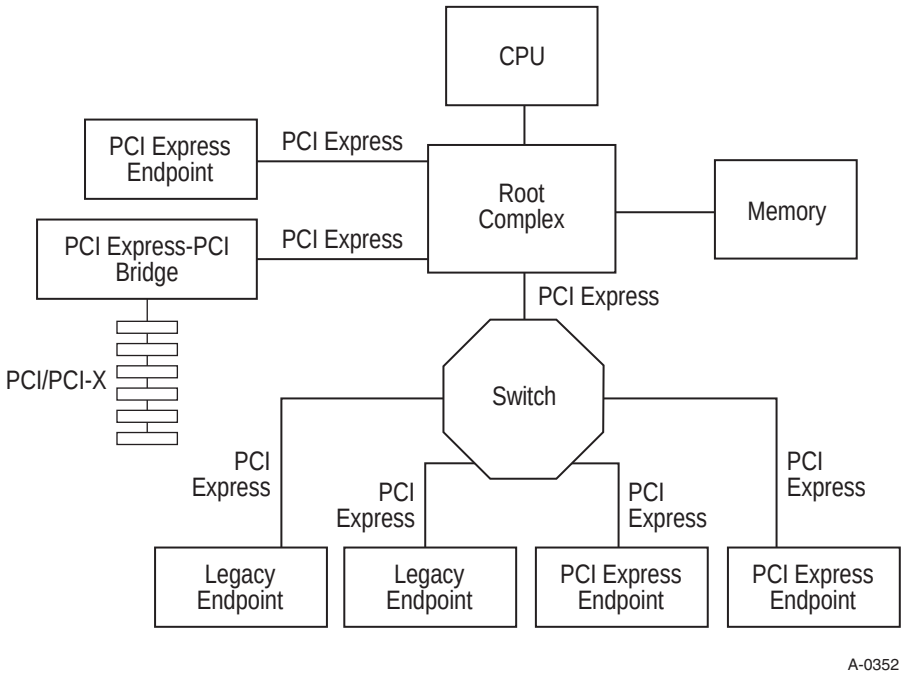

Figure 1-1: An Example System Topology

1.1. Evolution of the PCI-to-PCI Bridge Architecture

Originally, the PCI-to-PCI bridge architecture was developed to provide an I/O fan-out solution for multi-drop bus (i.e., conventional PCI) environments. This architecture was further extended with PCI-X to accommodate higher bandwidth applications. PCI Express was developed as a highly scaleable, point-to-point interconnect that enables component-level designers to provide multiple PCI Express Links out of the single component package as illustrated in Figure 1-2. By defining a standard PCI Express to PCI/PCI-X bridge architecture, system designers are provided with the

capability to preserve and extend existing conventional PCI and PCI-X fan-out requirements while simultaneously enabling the introduction of PCI Express interconnect technology into the system.

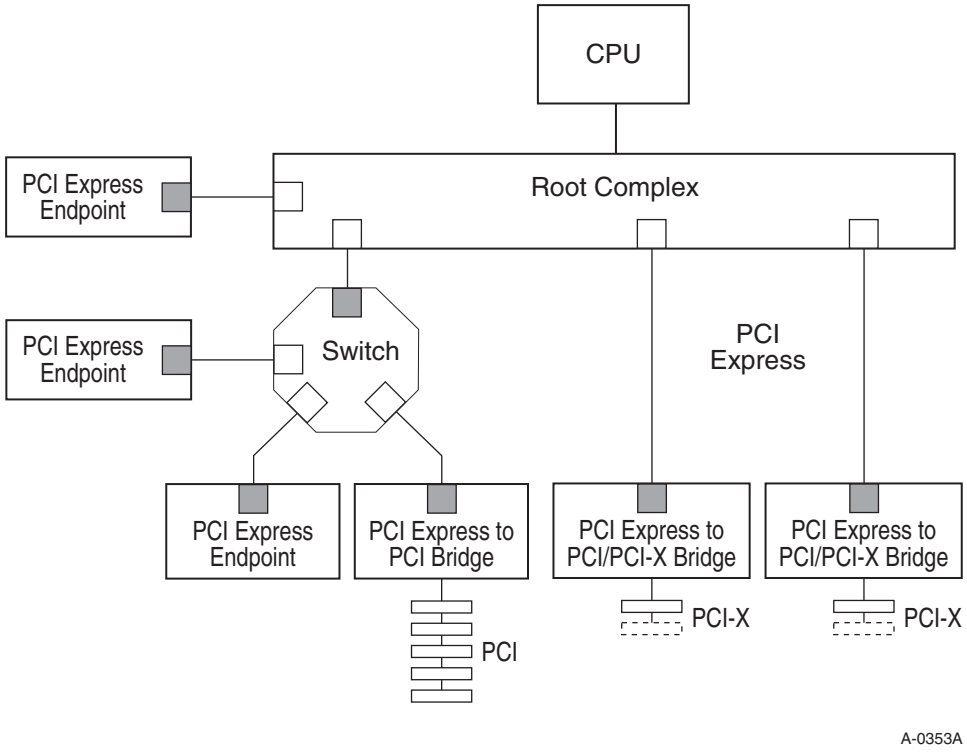

Figure 1-2: An Example of PCI/PCI-X Fan-Out Using PCI Express Bridges

The *PCI-PCI Bridge Architecture Specification, Revision 1.2*, was written to describe the requirements for bridging between two conventional PCI interfaces. PCI-X 1.0 introduced a set of additional bridge requirements that added rules governing bridging to PCI-X interfaces. The PCI-X bridge requirements were updated to include PCI-X Mode 2 in PCI-X 2.0a. The unique challenges present in facilitating translation between a PCI Express and conventional PCI or PCI-X interface call for a new bridge architecture specification, one that is more than a basic extension of the existing PCI-to-PCI bridge specification document. The PCI Express bridge architecture described in this document meets these challenges while maintaining full backward compatibility with the PCI-PCI bridge programming model. To accomplish software backward compatibility and support for forward-looking technologies, the PCI Express bridge architecture uses familiar PCI capability structures to add new functionality.

The programming interface of a PCI Express bridge:

- Maintains basic software backward-compatibility with the PCI bridge programming model (see PCI Bridge 1.2).
- Reports a Base Class Code encoding of 06h (Bridge Device), Sub Class Code of 04h (PCI-PCI Bridge), and Programming Interface of 00h (Non-Subtractive PCI-PCI Bridge) in the Device ID register of the Type 01h configuration header. Further information can be found in Appendix D of PCI 3.0.
- Includes the PCI Express Capabilities list item, the details of which can be found in Chapter 7 of PCI Express Base 1.0a.
- Reports a Device/Port Type of 0111b (PCI Express to PCI/PCI-X Bridge) in the PCI Express Capabilities register (see PCI Express 1.0, Chapter 7).

1.2. Bridge Topologies Considered

This specification describes the architectural requirements for bridges that possess a PCI Express primary interface and one or more conventional PCI/PCI-X secondary interfaces. Such a bridge will be referred to as a "PCI Express bridge," or simply as a "bridge." Architectural guidelines for a bridge that possesses a conventional PCI/PCI-X primary interface and one or more PCI Express secondary interfaces are presented in Appendix A. Such a bridge architecture will be referred to as a "PCI/PCI-X to PCI Express bridge," or simply as a "reverse bridge." The purpose of describing guidelines for reverse bridges is to facilitate rapid development of simple interface devices that permit native PCI Express devices on conventional PCI/PCI-X add-in cards to communicate with a system that supports conventional PCI/PCI-X compliant system interfaces.

If a bridge supports PCI-X, it may optionally support PCI-X Mode 2. When operating in any PCI-X mode, the secondary interface's behavior is governed by PCI-X 2.0a. Since any bus capable of operating in PCI-X mode must operate in conventional PCI mode when a conventional device is installed on that bus, a bridge's secondary interface must be capable of operating in conventional PCI mode. When operating in conventional PCI mode, the secondary interface's behavior is governed by PCI 3.0.

Unless otherwise specified in this document, a bridge must meet all the requirements specified in PCI 3.0 for conventional PCI devices and, if it supports PCI-X, in PCI-X PT 2.0a and PCI-X EM 2.0a for PCI-X devices on its secondary interface. As with PCI bridges and PCI-X bridges, a PCI Express bridge creates a new bus segment in the PCI configuration hierarchy. The PCI Express bridge is the source bridge for this segment and must meet all the requirements specified in this document for a source bridge. For example, a bridge that supports PCI-X must drive the PCI-X initialization pattern on the secondary bus before deasserting the secondary RST# to place secondary bus devices in the proper mode (conventional PCI or PCI-X) and to indicate the secondary bus clock frequency (in PCI-X mode).

A multi-ported PCI Express bridge component is a component with a PCI Express primary interface and more than one conventional PCI/PCI-X secondary bus interface. Such a component can be constructed using a number of logical bridge device architectures, as described in Section 1.6. In all cases, the device connected to the upstream interface will be a PCI Express bridge. More details on the multi-ported bridge architecture can be found in Section 1.6.

1.3. Bridge Requirements

1.3.1. Summary of Key Requirements

The following list is a summary of some of the key requirements of a PCI Express bridge. This list is compiled for the convenience of the reader and should not be considered as complete specification requirements:

- The bridge includes one PCI Express primary interface and one or more conventional PCI/PCI-X secondary interfaces.
- The bridge includes configuration registers accessible through the PCI-compatible configuration mechanism. The bridge programming model supports binary-code compatibility with PCI 3.0 (or later) aware operating systems and their corresponding bus enumeration and configuration software.
- The bridge is compliant with the electrical specifications described in PCI Express Base 1.0a and PCI 3.0 (PCI-X 2.0a if PCI-X is supported) for its respective interfaces.
- The secondary interface must support all PCI transactions required of devices as specified in PCI 3.0.
- The secondary interface must operate in conventional PCI mode if a conventional PCI device is installed on the corresponding bus segment (PCIXCAP connected to ground). The PCI Express bridge must sense the state of PCIXCAP and initialize the secondary bus devices properly (see Chapter 7).
- Memory mapped I/O address space for transaction forwarding (see Section 3.1.1).
- Like all PCI Express and PCI-X devices, the bridge must support 64-bit addressing on both the primary and secondary interfaces. To prevent address aliasing, the bridge must fully decode address fields.
- The bridge must handle all non-posted transactions that originate from the secondary interface as Delayed Transactions if the transaction crosses the bridge (i.e., the requester and completer are on different interfaces) and the originating interface is in conventional PCI mode.
- The bridge must complete all DWORD and burst memory read transactions that originate from the secondary interface as Split Transactions if the transaction crosses the bridge (i.e., the requester and completer are on different interfaces) and the originating interface is in a PCI-X mode.
- Transactions that originate from PCI Express and address locations internal to the bridge have the same requirements as described for PCI Express Endpoints.
- Hierarchical configuration transaction support, including pass through of extended configuration addresses when the secondary interface operates in PCI-X Mode 2.
- As with PCI bridges and PCI-X bridges, PCI Express bridges use a Type 01h Configuration Space header.

- PCI Express bridges must not propagate exclusive accesses from the secondary interface to the primary and are never allowed to initiate an exclusive access of their own.
- The PCI Express interface must comply with the definition of the flow control mechanism described in PCI Express Base 1.0a.
- The PCI Express interface must comply with the Link-level data integrity mechanism described in PCI Express Base 1.0a.

1.3.2. Central Resource Functions

The term central resource is used in PCI 3.0 to describe bus support features supplied by the host system, typically in a PCI-compliant bridge or standard chipset. Many of the central resource features may be integrated into a PCI Express bridge, especially those that directly support secondary bus operation. Examples of these functions include:

- Secondary bus Central Arbitration (REQ# as an input and GNT# as an output). If a bridge provides the arbiter for the secondary bus, it is required to implement a fairness algorithm to avoid potential deadlocks.
- Required secondary bus signal pull-ups or "keepers."
- Generation of the individual IDSEL signals to each device for system configuration.
- Driving REQ64# and PCI-X Initialization Pattern during reset.
- Bus parking functionality.

1.3.3. Optional Capabilities

Listed below are capabilities that a PCI Express bridge is not required to support but for which provisions have been made by this specification. These capabilities may be optionally supported by a bridge provided that the architecture adheres to the requirements and guidelines established in this specification. In addition, other specifications may apply.

- Generation and checking of 32-bit ECRC on PCI Express. Similar to PCI Express Endpoint devices that support ECRC, the bridge must support Advanced Error Reporting if it reports the ability to check ECRC (see Section 5.2.3, Section 10.1.1.2, and PCI Express Base 1.0a).
- Hot plug support for the PCI Express primary interface (see PCI Express Base 1.0a).
- PCI/PCI-X AD bus widths, other than 32 bits (see PCI 3.0 and PCI-X 2.0a).
- 66 MHz conventional PCI mode (see PCI 3.0).
- Arbitration support for secondary bus devices (see PCI 3.0).
- Support for optional address ranges (see Chapter 3):
  - I/O address range
  - Prefetchable memory address range
  - VGA addressing

- Propagation of exclusive accesses (i.e., Memory Read Request-Locked) from the primary interface to the secondary interface.
- Expansion ROM (see PCI 3.0).

Options specific to PCI-X:

- PCI-X Mode 1 or Mode 2 support. If PCI-X is supported by a bridge, it must support all PCI-X transactions required of devices as specified in PCI-X 2.0a (see Section 1.3.3.1 and Section 1.3.3.2 for the next level of detail).
- Vendor-Specific (Routed by ID or Implicit) PCI Express Message translation to/from PCI-X Device ID Messages (see Section 2.8.6).

1.3.3.1. PCI-X Mode 1 Support

Support of PCI-X Mode 1 is optional for PCI Express bridges. Additional requirements for bridges that support PCI-X Mode 1 support can be found throughout this document. The list below is a partial summary of the additional bridge-specific requirements (relative to the requirements for conventional PCI support):

- Implement version 0, 1, or 2 of the PCI-X Bridge Capabilities List item. Version 2 is required if Mode 1 ECC is supported. Bridges that implement either version 1 or 2 support ECC registers as well as updates to the PCI-X Bridge Status register and PCI-X Secondary Status register.
- Optional Device ID Message translation to/from PCI Express Messages.
- Other requirements as listed in PCI-X 2.0a.

1.3.3.2. PCI-X Mode 2 Support

Support of PCI-X Mode 2 is optional for PCI Express bridges. Additional requirements for bridges that support PCI-X Mode 2 support can be found throughout this document. The list below is a partial summary of the additional bridge-specific requirements (relative to the requirements for PCI-X Mode 1 support):

- Extended configuration address forwarding when the conventional PCI/PCI-X interface is operating in PCI-X Mode 2.
- Optional support of 16-bit Mode 2 operation.
- Additional error forwarding cases.
- Implement version 1 or 2 of the PCI-X Bridge Capabilities List item. Version 2 is required if Mode 1 ECC is supported. Bridges that implement either version 1 or 2 support ECC registers as well as updates to the PCI-X Bridge Status register and PCI-X Secondary Status register.
- Expanded PCI-X initialization patterns for PCI/PCI-X interface mode and frequency selection.
- Additional PCIXCAP encodings.
- PCI-X Mode 2 clock specification.
- Selectable V I/O (3.3V in PCI-X Mode 1 and 1.5V in PCI-X Mode 2).

- Other requirements as listed in PCI-X 2.0a.

1.3.4. Capabilities That Are Not Applicable or Are Outside The Scope of This Specification

The following list is a summary of capabilities that are considered to be either not applicable or outside the scope of this specification:

- Not applicable or not supported:
  - Forwarding and regeneration of ECC or parity, as defined in PCI-X PT 2.0a, between the secondary and primary interfaces.
  - Forwarding of end-to-end 32-bit CRC (ECRC), as defined in PCI Express Base 1.0a, between the primary and secondary interfaces.
  - Internal registers mapped into an I/O Base Address register.
  - Subtractive decode on primary interface.
  - Forwarding of Interrupt Acknowledge and Special Cycle transactions across the bridge. (Special Cycle transactions are only supported through Configuration Type 1 transactions.)
  - Support of VGA palette snooping.
- Outside of the scope of this specification:
  - Support for PCI Express extended Virtual Channels.
  - Support of PCI devices that require mapping to the first 1 MB of memory space.
  - ISA compatibility addressing for PCI devices other than VGA.
  - Primary boot ROM on secondary interface. (The bridge must be configured before access of downstream devices can occur.)
  - Support of downstream bridges to non-PCI buses.

1.4. Application Bridge Devices

There are certain classes of devices that exhibit some characteristics of bridges and some characteristics of peripheral components. PCI-X PT 2.0a defines a provision for such devices to claim a set of exceptions from the rules that apply to standard PCI-X bridges by declaring themselves as an application bridge (using the Device Complexity bit in the PCI-X Status register). Neither PCI Express Base 1.0a nor this specification provide a provision analogous to the PCI-X Device Complexity bit. Complex peripheral devices and bridge-like devices that connect directly to PCI Express have a choice of declaring themselves either as a PCI Express Endpoint, a PCI Express Legacy Endpoint, or a PCI Express bridge.

1.5. Software Compatibility

PCI Express bridge configuration space is accessible through the PCI-compatible configuration mechanism. For the baseline bridging capability (equivalent to the capabilities of PCI and PCI-X bridges), the programming model of the bridge and register defaults support full backward compatibility with PCI 3.0 or later aware operating systems and their corresponding bus enumeration and configuration software. Additional system software and/or firmware is required in order to take advantage of additional bridge capabilities, including optional features.

1.6. Multi-Ported Bridges

As physical layer bus technologies supporting multi-drop bus topologies give way to point-to-point interconnects, the bridge device becomes an essential element in determining system and add-in card fan-out capacity. For PCI Express bridge technology, one obvious way to achieve increased fan-out per level is to use bridge implementations that provide more than one conventional PCI/PCI-X bus interface (port) per physical component as illustrated in Figure 1-3. Such bridge implementations are referred to within this document as multi-ported bridges.

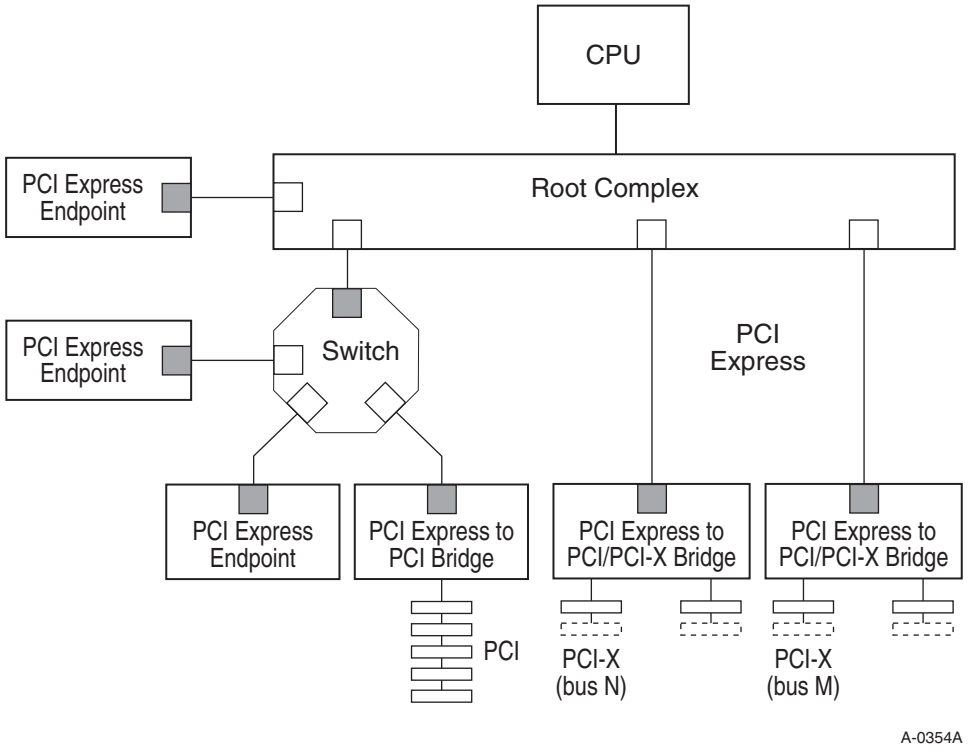

Figure 1-3: An Example of a Multi-Ported PCI Express Bridge Application

From the logical point of view, i.e., how these multi-ported bridges are seen in the configuration space, there are multiple implementation options. Figure 1-4 illustrates two basic methods of combining multiple individual logical bridges into a single physical multi-ported bridge component.

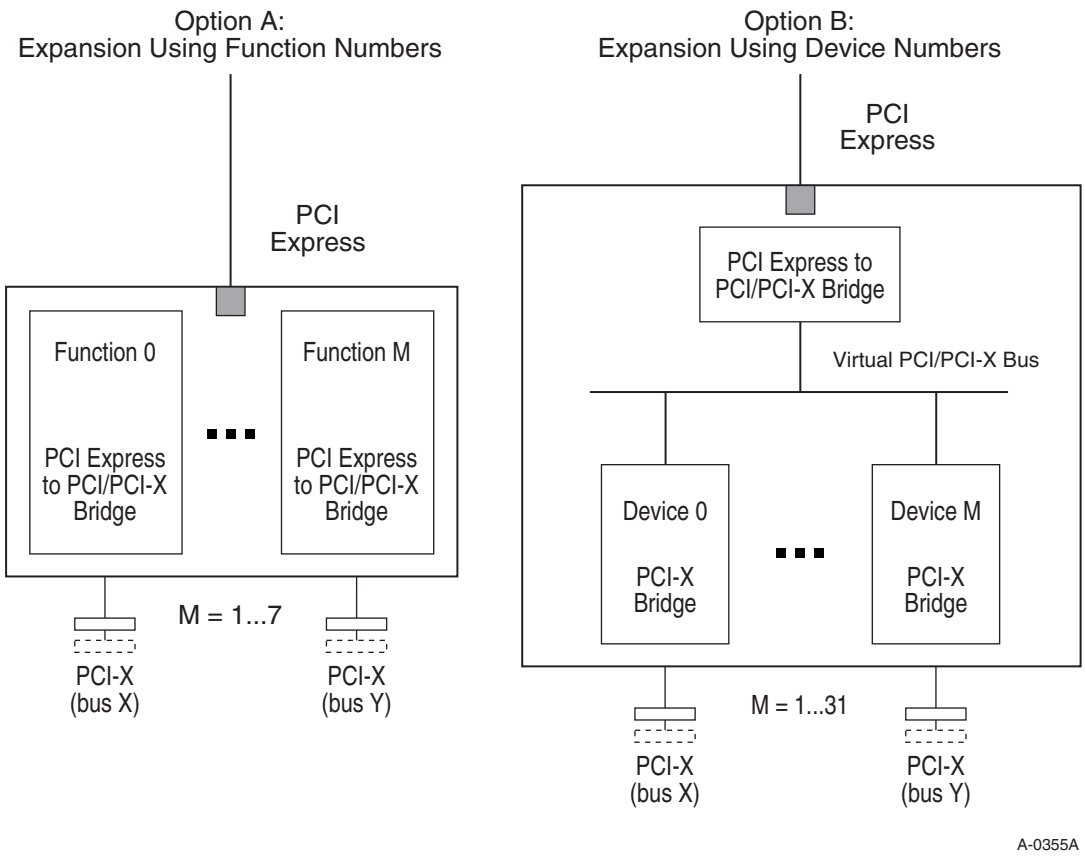

Figure 1-4: Expansion Approaches for Multi-Ported PCI Express Bridges

Figure 1-4 is presented for the purpose of illustration and is not intended as a guideline for implementing multi-ported bridge architectures. It is outside of the scope of this document to discuss in detail the tradeoffs associated with the illustrated approaches. It should be noted, however, that expansion using multiple functions (Option A) will permit preservation of all of the features/capabilities of the two-ported PCI Express bridges that are discussed in the remaining sections of this document, with one exception. The exception relates to the mapping of PCI INTx interrupts forwarded from the secondary interface and is discussed in detail in Section 8.1.

It is possible to use combinations of the two methods illustrated, where function-based expansion is used at the first level (i.e., closer to PCI Express interface) of the internal hierarchy and device-based expansion is used at the second level. This method can theoretically result in a fan-out capability that reaches and exceeds the architectural maximum number of PCI buses (i.e., 8 x 32) in a single hierarchy. Note, however, that practical implementations will be restricted by the package pin-out as well as by the printed-circuit-board layout requirements to a much smaller number of ports.

2

2. Bridge Operation

2.1. Minimum Flow Control Advertisements

PCI Express bridges must support the flow control mechanism described in PCI Express Base 1.0a and are subject to the minimum flow requirements shown in Table 2-1. During FC initialization of the default VC performed as a part of Link initialization, bridges must initially advertise credit values equal to or greater than those shown in Table 2-1. Note that support for extended VCs (i.e., other than the default VC) are outside the scope of this specification.

Table 2-1: Minimum PCI Express Flow Control Advertisements

Credit Type	Minimum Advertisement
PH	1 FC Unit – credit value of 01h
PD	Largest possible setting of the Max_Payload_Size for the component divided by FC Unit Size. Example: If the largest Max_Payload_Size value supported is 1024B, the smallest permitted initial credit value would be 040h.
NPH	1 FC Unit – credit value of 01h
NPD	1 FC Unit – credit value of 01h
CPLH	1 FC Unit – credit value of 01h
CPLD	Largest possible setting of the Max_Payload_Size for the component divided by FC Unit Size, or the size of the largest Read Request the component ever generates divided by the FC Unit Size, whichever is smaller

2.1.1. Conventional PCI Considerations

PCI Express bridges are permitted to advertise the minimum FC Unit advertisements for Completion Header and Completion Data as shown in Table 2-1. However, a PCI Express bridge implementation that only supports conventional PCI on its secondary interface(s) is recommended to advertise infinite credits (i.e., initial credit value of 0h) for Completion Header and Completion Data. Bridges making requests on PCI Express on behalf of a secondary interface operating in conventional PCI mode must reserve buffer space for the resulting completions.

2.1.2. PCI-X Considerations

PCI Express bridges must advertise finite FC credits for Completion Header and Completion Data when any secondary interface is operating or may be configured to operate in PCI-X mode (e.g., hot plug slots are implemented) with one exception (see below). Bridges are permitted to advertise the minimum FC Unit advertisements for Completion Header and Completion Data as shown in Table 2-1. However, for performance purposes, bridge implementations are strongly recommended to advertise much larger values than shown. When finite credit values are advertised, the bridge must provide buffer space commensurate with the Completion Header and Completion Data flow credits. Additionally, the bridge must advertise Completion Data Flow Credits sufficient to permit reception of TLPs with data payloads as large as the size the bridge reports via Max_Payload_Size Supported (see Chapter 7 of PCI Express Base 1.0a).

In the exception case, a bridge may advertise infinite FC credits for Completion Header and Completion Data when, in the implementation, it takes ownership (see Section 2.3) of all transactions initiated on the secondary interface. In this scenario, the bridge acts as the requester for the associated PCI Express transaction and is therefore subject to the rules for PCI Express Endpoints.

2.2. Buffer Size and Management Requirements

The following sections describe the minimum data buffer size requirements for PCI Express bridges. Section 2.2.1 covers the minimum buffering requirements associated with the PCI Express primary interface. Section 2.2.2 covers the minimum buffering requirements associated with the secondary interface. The buffering requirements related to the secondary interface are dependent upon whether the interface is operating in conventional PCI or one of the optional PCI-X modes, as noted.

2.2.1. PCI Express Considerations

The following bridge data buffer sizing requirements ensure proper PCI Express interoperability irrespective of the operating mode of the secondary interface:

- Follow the guidelines listed in PCI Express Base 1.0a regarding Retry Buffer sizing requirements.
- In the downstream direction, provide PCI Express data buffering consistent with the Flow Control credits advertised for Posted Data and Completion Data. The minimum data buffering requirement for both read completion data and posted write data is the amount specified by the Max_Payload_Size Supported field of the bridge's Device Capabilities register. Bridges are encouraged to implement much larger buffers to enable the posting of multiple and/or longer burst memory write transactions and Split Completions.

The PCI Express Root Complex may report its RCB capability as either 64 bytes or 128 bytes (see PCI Express Base 1.0a, Chapter 2). Bridges may optionally implement the Read Completion Boundary (RCB) bit of the Link Control register as read/write (see PCI Express Base 1.0a, Chapter 7) to permit this field to be programmed with the Root Complex's RCB parameter value. If the bridge does not implement this bit as read/write or if the bit is written to indicate an RCB of 64 bytes, at least one of the following rules must be observed (in addition to the other relevant rules

listed in this section and in Section 2.2.2) when the secondary interface is operating in a PCI-X mode to avoid potential deadlock:

- The bridge must provide adequate buffer space for all completion data associated with requests forwarded to PCI Express (i.e., the bridge is not permitted to overcommit its buffers).
- The bridge must have a minimum of 128 bytes of available read completion data buffer space for every read request issued on PCI Express that has a length equal to or greater than 128 bytes. For each read request issued with a length of less than 128 bytes, the bridge must have buffer space available sufficient to accommodate the entire request.

Failure to meet these minimum buffer requirements could result in the bridge holding less read data for a Split Completion than it can forward to PCI-X and leave it incapable of accepting additional read data from PCI Express.

2.2.2. Conventional PCI and PCI-X Considerations

PCI Express bridges operating with the secondary interface in conventional PCI mode have the following data buffering requirement:

- For upstream requests, the bridge must reserve buffer space for the resultant completions prior to forwarding the transaction to PCI Express.

PCI Express bridges operating with the secondary interface in a PCI-X mode have the following data buffering requirements:

- The bridge must have available a minimum buffer space of two ADQs in the upstream direction for holding immediate read data before initiating a read request on the PCI-X interface.
- The bridge must have available a minimum buffer space of one ADQ in the upstream direction for holding write data before signaling Data Transfer in response to a burst write transaction on the PCI-X interface. PCI Express bridges do not encounter the Disconnect at Next ADB corner cases on the destination interface that is applicable to PCI-X bridges.

IMPLEMENTATION NOTE

Buffer Space for PCI-X Immediate Read Data

If a bridge forwards a long memory read request to the PCI-X interface and the target responds immediately with data, the bridge must accept the data at least to the first ADB. If the starting address of the transaction is less than four data phases from an ADB, the bridge is not able to disconnect on that ADB and must proceed to the next. In this case, the bridge must have two ADQ buffers, one for the data between the starting address and the first ADB and the other for the data between the first and second ADBs.

If the Split Transaction Commitment Limit field in the bridge's Split Transaction Control register is set less than or equal to the Split Transaction Capacity field, the bridge always has buffer space available for the entire Sequence. In this case, no additional action is required to guarantee that two ADQ buffers are available before initiating the transaction.

2.3. Assignment of Requester ID and Tag by the Bridge

In some instances, the bridge must generate a new Requester ID and Tag combination for transactions forwarded to either the bridge's PCI Express interface or the secondary interface if it is operating in a PCI-X mode. The action of replacing the original transaction's Requester ID and/or Tag with one assigned by the bridge will be referred to as taking "ownership" of the transaction. In some cases, bridges are required to take ownership of either posted or non-posted transactions. In the case of non-posted transactions, the bridge must also maintain state information on a particular transaction in order to generate the appropriate completion.

The Tag is reassigned by the bridge according to the rules outlined in the following sections. If the bridge generates a new Requester ID for a transaction forwarded from the secondary interface to the primary interface, the bridge assigns the PCI Express Requester ID using its secondary interface's Bus Number and sets both the Device Number and Function Number fields to zero. If the bridge generates a new Requester ID for a transaction forwarded from PCI Express to a secondary interface operating in a PCI-X mode, the bridge assigns the PCI-X Requester ID using the bridge's primary Bus Number, Device Number, and Function Number.

The following summarizes the uniqueness requirements for the assignment of Requester ID and Tag for PCI Express and PCI-X transactions:

- All outstanding non-posted transactions (i.e., transactions requiring a completion) forwarded to PCI Express must possess a unique Transaction ID, even if multiple PCI Express transactions are generated for a single conventional PCI or PCI-X transaction.
- All posted write Sequences initiated but not completed on the PCI-X interface and outstanding non-posted transactions forwarded to PCI-X must possess a unique Sequence ID, even if multiple PCI-X transactions are generated for a single PCI Express transaction.

The following circumstances may result in a situation in which the bridge is required to take ownership of forwarded transactions (or, in case of posted writes, reassign the Tag of forwarded transactions) in order to avoid Sequence ID aliasing:

- A transaction forwarded to PCI-X is non-posted and the PCI Express Tag[7:5] is non-zero (see Section 2.4). The truncation of the 8-bit PCI Express Tag to a 5-bit PCI-X Tag may result in the unique Transaction IDs of two or more PCI Express transactions aliasing to the same PCI-X Sequence ID. The bridge is required to take ownership and provide a new 5-bit Tag for the PCI-X transaction.
- A posted write forwarded to PCI-X that uses the same Sequence ID as another posted write transaction (that was initiated, but is not completed, on PCI-X) or outstanding non-posted PCI-X transaction. If the previous sequence is a posted write, the bridge is permitted to either wait for the previous Sequence with the same Sequence ID to complete on the PCI-X interface or, alternatively, take ownership and assign a unique Sequence ID.[1] A PCI-X posted write Sequence is complete when its byte count has been satisfied or the target on the destination bus ends the transaction or Sequence in some other way. If the previous sequence is an outstanding

[1] Note that the Tag[7:0] field for is undefined for posted requests on PCI Express and may contain any value.

non-posted transaction, the bridge must take ownership of the posted write and assign a unique Sequence ID. In this case, the bridge must not make the forwarding of the write dependent upon the completion of the non-posted transaction.

- A conventional PCI or PCI-X non-posted transaction forwarded to PCI Express would cross a naturally-aligned 4 KB address boundary or a read request would exceed the PCI Express Max_Read_Request_Size. In either case, the bridge is required to ensure that all non-posted PCI Express transactions forwarded by the bridge possess unique Transaction IDs.
- All PCI-X Memory Read DWORD transactions forwarded to PCI Express. In this case, ownership is required because the bridge needs a mechanism to identify the PCI Express completion so that it can generate the proper completion transaction on PCI-X.

The bridge is required to return Split Completions to PCI-X with the Sequence ID supplied by the original request even if the bridge took ownership of the associated transaction(s) on PCI Express. The bridge is likewise required (under all circumstances) to return read completions to PCI Express with the Transaction ID supplied by the original request.

Note that, unlike PCI-X bridges defined in PCI-X PT 2.0a, PCI Express bridges may forward writes on the destination interface with a Sequence ID or Transaction ID that does not match the Transaction ID or Sequence ID received on the originating interface, respectively.

2.4. Forwarding PCI Express Transactions to PCI/PCI-X

The requirements for downstream forwarding of PCI Express transactions to conventional PCI and PCI-X are detailed in the following sections. In general, bridges forward requests to the secondary interface using a one-to-one mapping between a packet on the primary interface and a transaction on the secondary interface. Requests are mapped using the field translation rules that are defined in the following sections.

Due to protocol differences between PCI Express and the secondary interface protocols, one-to-one mapping between packets and transactions is not always possible. In these cases, the bridge is required to decompose PCI Express read requests and fragment PCI Express write requests into two or more requests on the secondary interface. Decomposing read requests requires the bridge to take ownership of these transactions. Taking ownership of read transactions requires a bridge to store relevant transaction state information (in order to generate the appropriate completion on the PCI Express interface) and bridge implementations will possess a finite amount of transaction buffer space to do so. Therefore, system performance may be adversely affected by this practice.

The decomposing of read requests is not the only situation in which the bridge is required to take ownership of transactions or store transaction-related state information. Bridges must also store transaction state information for all non-posted transactions that are forwarded to the secondary interface when it is operating in conventional PCI mode. Furthermore, bridges operating with the secondary interface in a PCI-X mode are required to take ownership of PCI Express non-posted requests (including reads) when the Tag[7:5] field is non-zero (as may be generated by PCI Express devices that support the Extended Tag field) since the corresponding field for PCI-X transactions, also called the Tag, is a 5-bit number. Failure to take ownership could result in Sequence ID aliasing on PCI-X side of the bridge.

The bridge is permitted to assign the PCI-X Tag to transactions that it takes ownership of using any algorithm. However, the Tag must be assigned such that the new Sequence ID is unique among all PCI-X Sequences (i.e., reads, non-posted writes, and posted writes) that have not ended. In any case, the bridge must return the original PCI Express request Tag[7:0] value with the completion.

For PCI Express transactions requiring a completion, bridges must return completion headers with the same values for the Requester ID, Tag, Attribute, and Traffic Class as were supplied in the header of the corresponding PCI Express request. Bridges that do not implement the VC capability must map all TCs to VC0 and forward all transactions regardless of the TC label.

IMPLEMENTATION NOTE

Returning TC values in Completions to PCI Express

Because the TC value is not applicable to PCI-X, the bridge must retain the TC contained in the original request in order to properly generate a subsequent completion on PCI Express. This requirement may be met by only storing information for TCs other than zero.

Table 2-2 summarizes the command (i.e., the Type field for PCI Express and the Command field for conventional PCI/PCI-X) translation requirements when forwarding a PCI Express transaction to conventional PCI or PCI-X. Commands that are not listed in this table are not forwarded by bridges in the downstream direction.

Table 2-2: PCI Express to Conventional PCI/PCI-X Command Translation

Primary Interface	Secondary Interface	
PCI Express Command	**Conventional PCI Command**	**PCI-X Command**
Memory Read Request	Memory Read or Memory Read Line or Memory Read Multiple	Memory Read DWORD or Memory Read Block
Memory Read Request - Locked	Memory Read or Memory Read Line or Memory Read Multiple[2]	Memory Read DWORD or Memory Read Block[2]
Memory Write Request	Memory Write Memory Write and Invalidate[3]	Memory Write Memory Write Block[3]
I/O Read Request	I/O Read	I/O Read
I/O Write Request	I/O Write	I/O Write

[2] LOCK# is asserted if supported (independent of the PCI/PCI-X read command used).

[3] The bridge must not use MWI or MWB if PCI Express Byte Enables are discontiguous.

Primary Interface	Secondary Interface	
Configuration Read Request – Type 1	Configuration Read Type 0 or Type 1	Configuration Read Type 0 or Type 1
Configuration Write Request – Type 1	Configuration Write Type 0 or Type 1, or Special Cycle	Configuration Write Type 0 or Type 1, or Special Cycle
Message Request – Vendor-Defined	N/A	N/A
Message Request with Data Payload – Vendor-Defined	N/A	Device ID Message[4]
Completion or Completion with Data	N/A	Split Completion

The following list summarizes the PCI header formation rules for transactions forwarded to conventional PCI from PCI Express:

- AD[63:0] or AD[31:0] – The start address of the transaction. Derived, in part, from the PCI Express Address [63:2] or Address [31:2], and 1st DW BE fields. PCI Express uses linear addressing for all transactions.
- C/BE#[7:0] – Command/Byte Enable signals. C/BE#[3:0] is populated with the PCI command during the address phase in accordance with the translations presented in Table 2-2.

The following list summarizes the additional rules for PCI-X attribute formation rules:

- Upper Byte Count, Lower Byte Count – A 12-bit value relevant to burst transactions that is the byte length of the remainder of a Sequence. It may be derived initially from PCI Express Length[9:0], the 1st DW BE, and the Last DW BE fields and later modified by the bridge.
- No Snoop – Forwarded unaltered by the bridge.
- Relaxed Ordering – Forwarded unaltered by the bridge.
- Tag – Either assigned by the bridge (if it takes ownership of transaction) or taken from PCI Express Tag[4:0].
- Requester Bus Number, Requester Device Number, and Requester Function Number – Either assigned by the bridge or taken from the PCI Express Requester ID.

See PCI 3.0 and PCI-X PT 2.0a for additional field requirements.

2.4.1. PCI Express Memory Write Request

When forwarding memory write transactions from PCI Express to PCI-X, bridges are required to ensure that the Sequence ID is unique with respect to all other posted write Sequences in progress (i.e., initiated but not completed) and outstanding non-posted transactions. It is recommended that a bridge take ownership of the PCI-X write transaction and assign a unique Sequence ID. Alternatively, in order to retain the Requester ID supplied with the PCI Express transaction, a bridge may choose to track all posted transactions in progress (i.e., initiated but not completed) and

[4] The translation and forwarding of DIMs and Vendor-Defined Messages with Data Payload across a bridge is optional.

outstanding non-posted transactions so that it may assign an appropriate Tag value that makes the Sequence ID unique.

Bridges are permitted to combine multiple PCI Express Memory Write Request transactions from the same requester (i.e., transactions that share the same Bus Number, Device Number, and Function Number) into a single PCI-X transaction to improve PCI-X bus efficiency. See Chapter 3 of PCI 3.0 for the definition of combining as it applies to writes.

See Chapter 2 of PCI-X PT 2.0a for more information on Sequence ID requirements.

2.4.1.1. Translation to PCI Memory Write and Invalidate

When forwarding a PCI Express Memory Write Request, the bridge can optionally translate the request (or even combine requests) into a PCI transaction using the Memory Write and Invalidate (MWI) command if it meets the MWI usage rules specified in PCI 3.0. If translation to MWIs is supported, the bridge must implement the Memory Write and Invalidate bit in the Command register (see Section 5.1.1.1) and the CacheLine Size register (see Section 5.1.1.3). In the respective registers, the Memory Write and Invalidate bit must be set and the cacheline size must be set to a value that the bridge supports or else MWI must not be generated.

A bridge is permitted to meet the MWI usage rules by promoting only a subset of a Memory Write Request. For example, consider a Memory Write Request that begins and ends on address boundaries that are not cacheline aligned but that contain one or more cache lines within the payload. When forwarding the Memory Write Request, the bridge is permitted to meet the MWI usage requirements by segmenting the original Memory Write Request into multiple (three in this example) separate transactions on the destination bus. The first transaction uses a MW transaction to transfer the memory write data from the starting address (which is not cacheline aligned) up to the next aligned cacheline boundary. The bridge then uses a MWI transaction to transfer the memory write data beginning on the aligned cacheline boundary including all subsequent complete cache lines up to the final aligned cacheline boundary contained in the original Memory Write Request. The bridge then uses a MW transaction to transfer the remainder of the data contained in the original Memory Write Request. Note that the bridge cannot alter the ordering of the original Memory Write Request and all bytes must be forwarded in the order they were received. Also note that the segmenting of writes into multiple separate transactions for the purpose of promotion may actually degrade system performance.

2.4.2. PCI Express Memory Read Request

In translating a PCI Express Memory Read Request into a transaction on conventional PCI, a bridge must use its prefetchable and non-prefetchable memory windows (as defined by base and limit registers) to determine, in part, the proper PCI read command to use. Memory Read Requests that fall into the non-prefetchable range must be translated into PCI transactions that use the Memory Read command. If a transaction from the primary interface of a bridge accesses a location mapped in the prefetchable memory address range for a secondary interface operating in conventional PCI mode, the bridge is permitted to ignore byte enable information when completing the transaction on the secondary interface. Furthermore, the bridge is permitted to modify the PCI bus command to Memory Read Line or Memory Read Multiple. The bridge is not permitted to extend the length of the transaction beyond the length requested on PCI Express.

Bridges operating with the secondary interface in a PCI-X mode must translate PCI Express Memory Read Requests into Memory Read DWORD transactions in the scenario described presently. Memory Read Requests with a length of two DWORDs (QWORD aligned) or less may have any combination of byte enable bits set for the first and/or second DWORDs. Memory Read Block transactions on PCI-X do not support byte enable encoding and all data bytes transferred from the starting byte address through the byte count are considered valid data. Therefore, a PCI Express bridge must decompose any Memory Read Request with a length of two DWORDs that targets non-prefetchable address space and that contains byte enables for the first or second DWORD that are non-contiguous into two Memory Read DWORD transactions on PCI-X. Failure to do so may result in the reading of address locations that were not intended to be read.

Read completions returned by the bridge for PCI Express read requests must contain the same Transaction ID as supplied with the original read request. This requirement persists even when a read request is forwarded to a secondary interface operating in conventional PCI mode, or when the bridge has taken ownership of a read request forwarded to a secondary interface operating in a PCI-X mode. Additionally, a bridge's Read Completion Boundary parameter (described in Chapter 2 of PCI Express Base 1.0a) is 128 B and it must be observed when forwarding read completions to PCI Express.

2.4.3. Configuration Reads and I/O Reads

The PCI Express Configuration Read (Type 0 and Type 1) and I/O Read Request transactions are non-prefetchable. Therefore, the bridge must translate Configuration Read Type 1 and I/O Read Request transactions such that the byte enable information is preserved and such that no additional bytes are requested. Since these transactions are limited to a single DWORD in length, this can be accomplished through appropriate use of the conventional PCI/PCI-X Byte Enable fields.

2.4.4. Forwarding Split Completions to PCI-X

If the Root Complex's RCB parameter is set to 64 bytes, the PCI Express bridge may receive read completions that are not of sufficient byte length to forward downstream as a Split Completion on PCI-X. This may occur when the PCI Express completions are associated with a PCI-X read request that spans one or more ADBs but the data received from PCI Express would not permit the bridge to transfer Split Completion data up to the ADB. In such a case, the bridge must buffer the received data until subsequent PCI Express completions associated with the read request are received and a Split Completion can be forwarded.

If the initial read completion associated with a PCI-X memory read request would require a bridge to disconnect (as a master) the resultant PCI-X completion at the first ADB and the number of data phases transferred would be less than four, the bridge is permitted to modify the byte count and set the BCM bit of the resultant PCI-X completion as described in Chapter 8 of PCI-X PT 2.0a. This case is analogous to a PCI-X bridge receiving an Immediate Response to a PCI-X read request.

See Chapter 8 of PCI-X PT 2.0a for additional requirements related to the forwarding of Split Completions.

2.5. Forwarding Conventional PCI/PCI-X Transactions to PCI Express

The requirements for upstream forwarding of conventional PCI and PCI-X transactions to PCI Express are detailed in the following sections. In general, bridges forward requests to the primary interface using a one-to-one mapping between a transaction on the secondary interface and a packet on the primary interface. Requests are mapped using the field translation rules that are defined in the following sections.

Bridges are encouraged to forward transactions without decomposing read requests or fragmenting write requests destined for the PCI Express interface. Decomposing read requests requires the bridge to take ownership of these transactions and maintain transaction state information needed to construct the appropriate completion. This practice may degrade system performance. Due to protocol differences between the secondary interface protocols and PCI Express, one-to-one mapping betweens transactions and packets is not always possible. In some cases, the bridge is required to decompose read requests and fragment write requests received from the secondary interface into two or more requests on the primary interface, as described presently and in the following sections. The requirements depend, in part, on the bus mode (conventional PCI or PCI-X) that the secondary interface is operating in.

PCI Express does not permit read requests to cross naturally-aligned 4-KB address boundaries. Therefore, the bridge must decompose Memory Read Block requests from PCI-X that possess a starting address and length that indicate that the read would cross one of these boundaries. Bridges must also not prefetch across these boundaries on behalf of memory read requests from conventional PCI. Additionally, the bridge must observe the Max_Read_Request_Size when forwarding read requests from the secondary interface (and decompose accordingly).

Bridges are sometimes required to fragment conventional PCI and PCI-X write requests containing discontiguous bytes into multiples PCI Express requests. This requirement is due to the fact that conventional PCI and PCI-X Memory Write transactions transfer byte enables with every data phase while PCI Express only supports byte enables on the first and last DWORDs of a TLP. Furthermore, discontiguous byte enables are only permitted on TLPs of one DWORD or two DWORDs (QWORD aligned) in length and TLPs with no byte enables set must be one DWORD in length. Note that PCI MWI and PCI-X Memory Write Block transactions cannot possess discontiguous byte enables and will therefore never be fragmented for this reason.

Bridges are required to fragment write requests that originate from the secondary interface whose starting address and length indicate that the write request would cross a naturally-aligned 4 KB address boundary when forwarded to the primary interface, since PCI Express does not permit this. Additionally, writes must be fragmented when the quantity of data transferred on the secondary interface exceeds the Max_Payload_Size parameter set for PCI Express on the primary interface.

A bridge's Read Completion Boundary parameter (described in Chapter 2 of PCI Express Base 1.0a) is 128 bytes and it must be observed when forwarding Split Completions to PCI Express.

When forwarding transactions to PCI Express, the bridge must ensure that all outstanding non-posted transactions that the bridge takes ownership of possess a unique Transaction ID. Unique Transaction IDs permit all completions to be properly identified for return to the requester.

The bridge must not set the Relaxed Ordering or No Snoop Attribute bits on transactions forwarded upstream from a bus operating in conventional PCI mode.

Table 2-3 summarizes the command (i.e., the Type field for PCI Express and the Command field for conventional PCI/PCI-X) translation requirements when forwarding a conventional PCI or PCI-X transaction to PCI Express. Commands that are not listed in this table are not supported by bridges in the upstream direction.

Table 2-3: Conventional PCI/PCI-X to PCI Express Command Translation

Secondary Interface		Primary Interface
Conventional PCI Command	**PCI-X Command**	**PCI Express Command**
Memory Read or Memory Read Line or Memory Read Multiple	Memory Read DWORD or Memory Read Block	Memory Read Request
Memory Write Memory Write and Invalidate	Memory Write Memory Write Block	Memory Write Request
I/O Read	I/O Read	I/O Read Request
I/O Write	I/O Write	I/O Write Request
N/A	Device ID Message[5]	Message Request with Data Payload – Vendor-Defined[5]
N/A	Split Completion	Completion or Completion with Data

The following list summarizes the PCI Express Header Field formation rules that are common to both conventional PCI and PCI-X transactions forwarded to PCI Express:

- Fmt[1:0] – Conventional PCI and PCI-X transactions with addresses below the 4 GB boundary use a 3 DW header while those with addresses at or above the 4 GB boundary use a 4 DW header. Write Requests, DIMs, and Split Completions use a TLP format with data.
- Type[4:0] – Populated based on command translations listed in Table 2-3 and requirements listed in the following sections. The treatment of this field with respect to DIMs is described in Section 2.8.
- TC[2:0] – For requests, this field must be zero. For completions, the field must contain the value supplied in the corresponding request.
- Attr[1:0] – Includes Relaxed Ordering and No Snoop attributes. The bridge must forward these bits unmodified from PCI-X. The bridge may optionally implement bypass logic to take advantage of relaxed ordering rules (see Section 2.9.2 for details). These bits must be set to zero for transactions forwarded from conventional PCI.
- TD – Set to 1b by the bridge when the TLP Digest contains ECRC. If the bridge implements ECRC, it is required to follow the rules for Endpoints listed in Chapter 2 of PCI Express Base 1.0a.

[5] The translation and forwarding of DIMs and Vendor-Defined Messages with Data Payload across a bridge is optional.

- EP – Set to 1b by the bridge when forwarding an uncorrectable data error from conventional PCI/PCI-X (see Chapter 10 for details).
- Length[9:0] – Length of conventional PCI/PCI-X write or read request rounded up to nearest DW-aligned boundary (1 to 1024 DWORDs). Used in conjunction with the First/Last DW Byte Enable fields to support the conventional PCI/PCI-X transaction length granularity of a single byte.
- Requester ID[15:0] – Either assigned by the bridge or taken directly from the PCI-X Requester ID field.
- Tag[7:0] – Either assigned by the bridge or Tag[4:0] taken directly from PCI-X Tag[4:0] field with Tag[7:5] set to zero.
- 1st DW BE[3:0] and Last DW BE[3:0] – Used for Memory, I/O, and Configuration Requests. Settings are determined by the conventional PCI/PCI-X transaction starting byte address and implicit (conventional PCI and configuration transactions) or explicit (PCI-X) byte length.
- Address[63:2] (4 DW header) or Address[31:2] (3 DW header) – DWORD starting address for the transaction. The byte address of the conventional PCI/PCI-X transaction is rounded down to the nearest DW-aligned boundary to derive this field.

See PCI Express Base 1.0a for additional field requirements.

2.5.1. Conventional PCI Requester

If a bridge forwards any transaction from conventional PCI to PCI Express, the bridge must create a Requester ID and Tag to form the PCI Express Transaction ID. The bridge uses the Bus Number for its conventional PCI interface (from the Secondary Bus Number register) and sets both the Device Number and Function Number fields to zero. In the case of a non-posted transaction, the bridge will match the resulting PCI Express completion(s) to the Delayed Request state information based on the Bus Number in the Requester ID field and the Tag(s) associated with the completion header(s) (the Device Number and Function Number fields in the completion header are ignored in this case).

The bridge is permitted to assign Tag values to transactions forwarded from conventional PCI to PCI Express using any algorithm. For example, if the bridge enqueues multiple Delayed Transactions on the conventional PCI interface, the Tag could be assigned according to the Delayed Transaction with which it is associated. However, the Tag must be assigned such that the Transaction IDs for all outstanding transactions on PCI Express requiring a completion (i.e., reads and non-posted writes) are unique.

Bridges must never set the PCI Express Relaxed Ordering or No Snoop Attribute bits on transactions forwarded from a conventional PCI bus.

2.5.1.1. Write Transactions

Bridges translate PCI Memory Write and Memory Write and Invalidate transactions into PCI Express Memory Write Requests as shown in Table 2-3. Internal posting of write data is required for PCI transactions that use the Memory Write or Memory Write and Invalidate commands when

the bridge has posting buffer space available. Posting of I/O Write and Configuration Write transactions by a bridge is not permitted since these are non-posted transactions that require a completion.

PCI Express does not permit write requests to cross naturally-aligned 4-KB address boundaries. Therefore, the bridge must fragment memory write transactions that originate from PCI such that they do not cross these boundaries on PCI Express. Additionally, the bridge must observe the Max_Payload_Size parameter when forwarding PCI memory write transactions to PCI Express and fragment these transactions accordingly.

A bridge is generally allowed to terminate with Retry/Disconnect a transaction that uses the Memory Write or Memory Write and Invalidate commands only when its buffers are filled with previously received memory write data or to Retry the transaction if the bridge is locked from the PCI Express side. Terminating a Memory Write or Memory Write and Invalidate transaction with Retry for other reasons can lead to deadlocks. Refer to Chapter 5 of PCI Bridge 1.2 for more details on deadlocks.

2.5.1.2. Delayed Transactions

Only non-posted conventional PCI transactions may be completed as Delayed Transactions by a bridge. These include I/O Read, I/O Write, Configuration Read, Configuration Write, Memory Read, Memory Read Line, and Memory Read Multiple. Memory Write and Memory Write and Invalidate transactions are postable and, therefore, cannot be completed as Delayed Transactions. A Delayed Transaction occurs when a bridge responds to a non-posted transaction with Retry and subsequently forwards the request onto the PCI Express interface. When the associated completion returns from PCI Express, the bridge buffers the completion (with the exceptions noted in Section 2.5.1.2.2) until the PCI requester reissues the transaction and the Delayed Transaction sequence terminates.

To complete a transaction using Delayed Transaction termination, a bridge must latch the following information:

- Address
- Command
- Byte enables
- Address and data parity
- REQ64# (if a 64-bit transfer)

For write transactions completed using Delayed Transaction termination, a bridge must also latch data from byte lanes for which the byte enable is asserted and may optionally latch data from byte lanes for which the byte enable is deasserted.

After latching the required information, the bridge terminates the transaction on the originating (secondary) interface with Retry. Once upstream ordering requirements have been satisfied (see Section 2.9.1) and there are sufficient Flow Credits available, the bridge transmits the request on the destination (primary) interface. If the Delayed Request is a read, the bridge obtains the requested data and completion status. If the Delayed Request is a write, the bridge delivers the write data and obtains the completion status. The completion returned from the PCI Express interface is stored as

a Delayed Completion by the bridge. The bridge stores the Delayed Completion until the master repeats the initial request, with the exceptions noted in Section 2.5.1.2.2.

The bridge differentiates between transactions (by the same or different masters) by comparing the current transaction with information latched previously (for both Delayed Requests and Delayed Completions). When the Parity Error Response Enable bit of the Bridge Control register for the secondary interface is cleared, the bridge ignores the address and data parity latched previously when doing the comparison. The byte enables may optionally be ignored in the comparison if the master is reading from a prefetchable location, even though the master is required to repeat the transaction with the same byte enables. If the compare matches a Delayed Request (already enqueued), but the bridge is not ready to complete the request, the bridge does not enqueue the request again, but simply terminates the transaction with Retry. If the compare matches a Delayed Completion and the bridge is ready to complete the request, the bridge responds by signaling the status and provides the data if completing a read transaction.

The master must repeat the transaction exactly as the original request; otherwise, the bridge assumes it is a new transaction (since the bridge cannot distinguish masters). Two masters could request the exact same transaction and the bridge cannot and need not distinguish between them.

A bridge is permitted to enqueue one or more Delayed Requests at a time. If a bridge enqueues multiple Delayed Requests, the order in which it attempts them on the PCI Express interface is independent of the order in which they were originally attempted on the PCI interface. Furthermore, the order in which the transactions ultimately complete on the PCI interface is independent of the order in which they were attempted on the PCI interface and the order in which they completed on the PCI Express interface.

Refer to Section 2.9 for the deadlock avoidance requirements with respect to Delayed Transactions and memory write transactions.

2.5.1.2.1. Discarding a Delayed Request

The bridge is permitted to discard a Delayed Request from the time it is enqueued until it has been transmitted on the PCI Express interface since the requester is required to reissue the request until it completes and since no undesirable side effects will result if the transaction is later transmitted. Once a request is transmitted, the transaction becomes a Delayed Completion and cannot be discarded, except as noted in Section 2.5.1.2.2. The bridge is allowed to forward other PCI requests onto PCI Express prior to the completion of a previous PCI request, but must continue to enqueue data associated with previous requests (i.e., Delayed Completions) that have not yet completed on PCI.

2.5.1.2.2. Discarding a Delayed Completion

The bridge is allowed to discard Delayed Completions in only two cases. The first case is if the Delayed Completion is a read of a prefetchable region (or the command was Memory Read Line or Memory Read Multiple). The second case is for all Delayed Completions (read or write, prefetchable or not) if the master has not repeated the request before the Secondary Discard Timer interval is exceeded. When this occurs, the device is required to discard the Delayed Completion; otherwise, a deadlock may occur.

The Secondary Discard Timer for masters on the secondary interface is selectable to expire either within 2^{15} or 2^{10} conventional PCI/PCI-X common clock cycles depending on the state of the Secondary Discard Timer bit in the Bridge Control register. When the Discard Timer interval is exceeded on the secondary interface, the bridge must set the Discard Timer Status bit in the Bridge Control register. In addition, the bridge transmits an error message on the primary interface if both of the following requirements are met:

- The Discard Timer SERR# Enable bit is set in the Bridge Control register or, if Advanced Error Reporting is supported, the Delayed Transaction Discard Timer Expired Mask bit is clear in the Secondary Uncorrectable Error Mask register
- Either the SERR# Enable bit is set in the Command register or the PCI Express-specific error reporting enable bit is set in the Device Control register to enable transmission for the appropriate severity level (default severity is non-fatal).

Refer to Section 2.9 for the deadlock avoidance requirements with respect to Delayed Completions and memory write transactions.

2.5.1.3. Read Data Prefetching

The term "prefetch" refers to actions that the bridge takes when it reads data from the target in anticipation that the PCI master will consume it. Prefetching is a useful technique for hiding the latency of a burst read transaction, but its use is restricted. Memory that is prefetchable has the attribute that it returns the same data when read multiple times and does not alter any other device state when it is read. When prefetching, the bridge may read data that is not consumed by the master. The bridge is required to discard any prefetched read data not consumed when the master terminates the read transaction (see PCI Bridge 1.2, Chapter 5).

A PCI Express bridge is permitted to prefetch data from the primary PCI Express interface on behalf of a conventional PCI transaction that uses the Memory Read Line or Memory Read Multiple command. PCI Express Memory Read transactions initiated on behalf of a PCI transaction must not specify an address/data combination that would cause a memory space access to cross a naturally-aligned 4-KB address boundary or request a data quantity that exceeds the amount specified by the PCI Express Max_Read_Request Size parameter.

Masters attached to the secondary interface of the bridge are encouraged to use the Memory Read Line or Memory Read Multiple commands if they desire high performance (prefetchable) read transactions. If the system design guarantees that all transactions that originate on the secondary interface have a final destination at main memory, the bridge is permitted to perform prefetching on behalf of transactions on the secondary interface that use the PCI Memory Read command. However, if the bridge does so, it must support a device-specific bit (in Configuration Space as described in PCI Bridge 1.2) that allows this feature to be disabled (if Memory Read prefetching causes a problem). When prefetching memory read data, the bridge is permitted to set all bits in the PCI Express First and Last DWORD Byte Enable fields independent of the byte enables used by the originating bus master.

2.5.1.4. I/O and Configuration Requests

PCI Express bridges are permitted to forward I/O requests in the upstream direction. Since I/O requests are non-prefetchable, the bridge must forward I/O requests such that the byte enable information is preserved and such that no additional bytes are requested. This can be accomplished through appropriate use of the PCI Express First/Last DWORD Byte Enable fields and Length field.

Conventional PCI/PCI-X bridges only permit the upstream forwarding of Type 1 configuration transactions that are converted to Special Cycle transactions on the primary interface (see Chapter 3 of PCI Bridge 1.2). This case is not applicable to PCI Express bridges. Therefore, a PCI Express bridge is not permitted to claim configuration transactions on the secondary interface.

2.5.2. PCI-X Requester

Bridges are required to support the decomposition of PCI-X Memory Read Block transactions into two or more Memory Read Request TLPs on PCI Express. The decomposition of memory read transactions is required when a PCI-X memory read request crosses a 4-KB boundary or when the PCI Express Max_Read_Request_Size parameter is set to a value that is smaller than the byte length of the PCI-X memory read request.

Bridges are also required to support the fragmenting of PCI-X Memory Write and Memory Write Block transactions into two or more Memory Write Request TLPs on PCI Express. The fragmenting of write requests forwarded to PCI Express is analogous to the ability of PCI-X write transfers to disconnect on ADBs prior to exhaustion of the byte count. Write burst transactions must be broken up when the PCI Express Max_Payload_Size parameter is set to a value that is smaller than the byte length of the PCI-X memory write request or when a PCI-X request crosses a 4-KB address boundary.

2.5.2.1. Memory Read DWORD and Memory Read Block Requests

Bridges translate both PCI-X Memory Read DWORD and Memory Read Block transactions into PCI Express Memory Read Requests. In many cases, Memory Read Block requests may be forwarded to PCI Express without retaining any state related to the forwarded transaction. However, the bridge must take ownership of decomposed Memory Read Block requests and of all Memory Read DWORD transactions forwarded to PCI-X from PCI Express.

When a bridge takes ownership of a read request initiated on an interface operating in PCI-X mode, the bridge must create a Requester ID and Tag (see Section 2.5.1 for Tag assignment rules) for the PCI Express Transaction ID. The bridge uses the Bus Number for the PCI-X interface (the Secondary Bus Number register) and sets the Device Number and Function Number fields to zero. When the PCI Express completion returns to the bridge, the bridge forwards it to the PCI-X interface based on the Bus Number in the Requester ID field of the PCI Express completion header. The bridge must restore the original PCI-X Requester ID and Tag when forwarding the Split Completion.

A bridge uses the Completer ID from the PCI Express completion when generating the corresponding PCI-X Split Completion. Split Completions within a Sequence must be returned in address order.

2.5.2.2. Write Requests

As in conventional PCI, bridges operating with the secondary interface in a PCI-X mode are required to post memory write transactions that cross the bridge if space is available in the bridge. The conditions under which the bridge is permitted to terminate a memory write transaction with Retry are specified in Section 2.10. If Device ID Messages are supported, they are treated the same as memory write transactions, with respect to buffer management and termination (see Section 2.8.6 for more details).

When forwarding non-posted write transactions from PCI-X to PCI Express, bridges are required to forward the PCI Express transaction using the same Requester ID and Tag as used by the associated PCI-X transaction. When forwarding posted write transactions, bridges must use the Requester ID supplied by the PCI-X transaction but may assign any value to the PCI Express Tag field (since the field is undefined in PCI Express). It is recommended that bridges use the same value for the Tag field as originally supplied with the PCI-X transaction. PCI Express Tags for posted writes may be reused by the bridge as soon as the respective write is transmitted by the bridge. See Section 2.8.6 for the rules for forwarding Device ID Messages.

Bridges are permitted to combine memory writes (as defined in Chapter 3 of PCI 3.0) from the same PCI-X Sequence when forwarding to the PCI Express interface. Bridges are not permitted to combine memory writes across different PCI-X Sequences. Bridges may combine writes up to the full length of the PCI-X Sequence, until the PCI Express Max_Payload_Size limit is reached, or until a naturally-aligned 4-KB address boundary is reached, whichever comes first.

2.6. Split Completion Buffer Allocation

PCI Express bridges that support PCI-X modes contain two registers that limit the upstream and downstream forwarding of Split Requests (see PCI-X PT 2.0a for details on Split Requests). The Split Transaction Capacity register indicates the amount of buffer space the bridge has for storing Split Completions. If the bridge stores Split Completions for burst memory read requests in a separate area from other Split Completions, this register indicates the size (in units of ADQs) of the area for storing Split Completions for burst memory reads. If the bridge stores Split Completions for burst memory reads in the same area as some or all other Split Completions, this register indicates the size of this area in units of ADQs.

The Split Transaction Commitment Limit registers (see Sections 5.2.2.1.5 and 0 for detailed descriptions) indicate the cumulative Sequence size of Split Transactions that the bridge is permitted to have outstanding in the upstream and downstream directions at any given instant (in units of ADQs). If the bridge enqueues a request to be forwarded and the size of that request plus all those the bridge presently has outstanding in a particular direction (upstream or downstream) exceeds the contents of the Split Transaction Commitment register governing that direction, the bridge is not permitted to forward the request to the respective interface. Refer to the register descriptions in Chapter 5 for exceptions to this rule. After sufficient Split Completion transactions have been forwarded to their respective requesters such that the size of the enqueued request plus the total

outstanding commitment is less than the commitment limit, the bridge is permitted to forward the transaction.

If the bridge stores Split Completions for burst memory read requests in a separate area from other Split Completions, the Split Transaction Commitment Limit register applies only to burst memory reads. Such a bridge must never forward other Split Requests (e.g., I/O Read, I/O write, etc.) unless it has a place to store the corresponding Split Completions. If the bridge stores Split Completions for burst memory reads in the same area as some or all other Split Completions, this register applies to all Split Transactions stored with burst memory read transactions.

The upstream Split Transaction Commitment Limit register may be programmed to a value greater than the bridge's upstream Split Transaction Capacity register indicates. These Upstream Split Transaction register settings offer potential performance benefits for bridges that advertise finite FC credits (see Section 2.1.2) because the bridge is permitted to request data prior to completion buffer availability, effectively hiding some I/O latency. The Split Completion Overrun bit in the PCI-X Bridge Status register assists in the tuning of the Split Transaction Commitment Limit register setting.

In contrast, bridges that advertise infinite FC credits must not forward Split Transactions upstream if the Split Completions for these transactions would not fit in the bridge's buffer space at one time, even if the upstream Split Transaction Commitment Limit register is programmed to a value greater than the upstream Split Transaction Capacity register indicates (refer to the Implementation Note below). As a result, these bridges will never set the Split Completion Overrun bit in the PCI-X Bridge Status register.

See Chapter 8 and Appendix D of PCI-X PT 2.0a for additional details.

IMPLEMENTATION NOTE

Avoiding Deadlock if the Secondary Interface is in PCI-X Mode

PCI Express Base 1.0a ordering rules do not require that posted writes be permitted to pass completions (read or write), introducing the possibility of a deadlock case if downstream backpressure occurs on the completions. If the secondary interface is in conventional PCI mode, the bridge is required to provide sufficient buffer space for holding all completion data associated with outstanding requests forwarded to PCI Express and will thereby prevent deadlock from arising. In PCI-X mode, application bridges may create problematic backpressure. If one or more PCI-X application bridges are present downstream of the PCI Express bridge, the bridge must have its upstream Split Transaction Commitment Limit set no larger than the upstream Split Transaction Capacity. If there is a PCI-X bridge on the path between the PCI Express bridge and the application bridge, that bridge may be programmed as such in lieu of the PCI Express bridge.

2.7. Transaction Support

The following sections summarize the bridge support defined by this specification for each PCI Express and conventional PCI/PCI-X transaction in one of the following four operating cases:

- The bridge initiating the transaction as a *transmitter* on the *primary* interface
- The bridge responding to the transaction as a *receiver* on the *primary* interface
- The bridge initiating the transaction as a *master* on the *secondary* interface
- The bridge responding to the transaction as a *target* on the *secondary* interface

The symbols used in the table entries are defined as follows:

- NA – Not allowed. This specification does not allow any use of the transaction. As a Receiver of a command that is Not Allowed, the bridge must respond with Unsupported Request on PCI Express and Master Abort on PCI, unless otherwise noted in this specification.
- REQ – Required. This specification requires support of the transaction by a bridge for the operating case.
- OPT – Optional. This specification defines optional support of the transaction by bridges for the operating case. An implementation may optionally support the transaction for the operating case but must do so as specified by this document. If support for a transaction is not provided, the bridge must respond to transactions that target it as described in NA (not allowed) above.

2.7.1. PCI Express Transaction Support

Table 2-4 lists the support defined for transactions transmitted by and received by the bridge on the PCI Express primary interface.

Table 2-4: Transaction Support Summary for the PCI Express Interface

TLP Type	Transaction	Primary Interface	
		Transmitter	Receiver
MRd	Memory Read Request	REQ	REQ
MRdLk	Memory Read Request – Locked	NA	OPT[6]
MWr	Memory Write Request	REQ	REQ
IORd	I/O Read Request	REQ	OPT
IOWr	I/O Write Request	REQ	OPT
CfgRd0	Configuration Read Type 0	NA	REQ
CfgWr0	Configuration Write Type 0	NA	REQ
CfgRd1	Configuration Read Type 1	NA	REQ
CfgWr1	Configuration Write Type 1	NA	REQ

[6] See Section 2.13 for specific requirements to support exclusive accesses in system architectures that require them.

TLP Type	Transaction	Primary Interface	
		Transmitter	Receiver
Msg	Message Request	REQ	REQ
MsgD	Message Request with Data Payload	NA	REQ
MsgD (Vendor-Defined)	Vendor-Defined Message Request with Data Payload	OPT[7]	OPT[7], [8]
Cpl	Completion without Data	REQ	REQ
CplD	Completion with Data	REQ	REQ
CplLk	Completion without Data for MRR – Locked	OPT	NA
CplDLk	Completion with Data for MRR – Locked	OPT	NA

2.7.2. Conventional PCI/PCI-X Transaction Support

Table 2-5 lists the support defined for transactions transmitted by and received by the bridge on the secondary interface.

Table 2-5: Transaction Support Summary for the Secondary Interface

Cmd	PCI/PCI-X Transaction	Conventional PCI Secondary Interface		PCI-X Secondary Interface	
		Master	Target	Master	Target
0000b	Interrupt Acknowledge	NA	NA	NA	NA
0001b	Special Cycle	REQ[9]	NA	REQ[9]	NA
0010b	I/O Read	OPT	REQ	OPT	REQ
0011b	I/O Write	OPT	REQ	OPT	REQ
0100b	Rsvd	NA	NA	NA	NA
0101b	Rsvd/Device ID Message	NA	NA	OPT[10]	OPT[10]
0110b	Memory Read/Memory Read DWORD	REQ	REQ	REQ	REQ
0111b	Memory Write	REQ	REQ	REQ	REQ

[7] If message translation is supported, it must be supported in both the upstream and downstream directions.

[8] See Chapter 2 of PCI Express Base 1.0a for the requirements for handling Vendor-Defined Messages that are not supported (or forwarded) by the bridge.

[9] Only initiated by the bridge when forwarding a Type 1 Configuration write transaction that specifies conversion to a Special Cycle transaction on the secondary interface.

[10] Support is optional for PCI-X Mode 1 and PCI-X Mode 2. If message translation is supported, it must be supported in both the upstream and downstream directions. See Section 2.8.6 for the requirements for handling Device ID Messages that are not forwarded by the bridge.

Cmd	PCI/PCI-X Transaction	Conventional PCI Secondary Interface		PCI-X Secondary Interface	
		Master	Target	Master	Target
1000b	Rsvd/Alias to Memory Read Block	NA	NA	NA	REQ
1001b	Rsvd/Alias to Memory Write Block	NA	NA	NA	REQ
1010b	Configuration Read	REQ	NA	REQ	NA
1011b	Configuration Write	REQ	NA	REQ	NA
1100b	Memory Read Multiple/Split Completion	OPT	REQ	REQ	REQ
1101b	Dual Address Cycle	OPT	OPT	REQ	REQ
1110b	Memory Read Line/Memory Read Block	OPT	REQ	REQ	REQ
1111b	Memory Write and Invalidate/Memory Write Block	OPT	REQ	REQ	REQ

2.8. Message Transactions

PCI Express defines a set of messages that are used as a method for in-band communication of events (e.g., interrupts) generally replacing the need for sideband signals. PCI Express Vendor-Defined Messages are available for general purpose messaging and other vendor specific functions. PCI Express bridge support requirements for message transactions are described in the sections below.

PCI Express messages are routed either explicitly or implicitly depending on specific bit field encodings in the message request header. An explicitly routed message is routed based either on a specific address or on an ID field contained within the message header while the destination of an implicitly routed message is inferred from the message Type field. See PCI Express Base 1.0a for details.

The Device ID Messages described in PCI-X PT 2.0a provide similar functionality to PCI Express Vendor-Defined Messages. PCI Express bridges optionally support translation and forwarding of Device ID Messages to/from PCI Express Vendor-Defined Messages as described in Section 2.8.6.

2.8.1. INTx Interrupt Signaling

INTx Interrupt Signaling messages are used for in-band communication of the state of the PCI line based interrupts INTA#, INTB#, INTC#, and INTD# for devices downstream of the bridge. Support for these messages is required. See Section 8.2 for details.

2.8.2. Power Management Messages

Power Management Messages are used to support Power Management Events signaled by sources integrated into the bridge and for devices downstream of the bridge. See Chapter 9 for details on bridge requirements for power management support.

2.8.3. Error Signaling Messages

Error Signaling Messages are transmitted by the bridge on its PCI Express primary interface to signal an error for a particular transaction, for the Link interface, for errors internal to the bridge, or for conventional PCI/PCI-X related errors detected on the secondary interface. The message types include ERR_COR, ERR_NONFATAL, and ERR_FATAL and the relevant mask bits are located in the PCI Express Capability Structure. References to error events and the rules associated with escalating them to these messages are made throughout Chapter 5 and Chapter 10.

2.8.4. Locked Transactions Support

The PCI Express Unlock Message is used to support Locked Transaction sequences in the downstream direction. See Section 2.13 of this specification for bridge support requirements and Chapter 6 of PCI Express Base 1.0a for details on Locked Transaction sequences.

2.8.5. Slot Power Limit Support

The Set_Slot_Power_Limit Message is transmitted to devices by the Root Complex or a Switch. Some bridges on add-in cards are required to comply with the Set Power Limit control mechanism as described in Chapter 6 of PCI Express Base 1.0a. Bridges that are targeted exclusively for integration on the system planar (e.g., the system board), as well as bridges that are targeted for integration on a card/module where power consumption of the entire card/module is below the lowest power limit specified for the card/module form factor (as defined in the corresponding electromechanical specification), are permitted to hardwire the value to 0b in the Slot Power Limit Scale and Slot Power Limit Value fields of the Device Capabilities register, and are not required to copy the Set Slot Power limit payload into that register. These bridges are, however, required to receive the Set_Slot_Power_Limit Message correctly but simply discard it.

2.8.6. Vendor-Defined and Device ID Messages

PCI Express Base 1.0a defines a set of Vendor-Defined Messages that may be used for vendor-specific purposes. Although PCI Express Base 1.0a defines the basic format as well as the required handling rules for these messages, the complete definition and use of these messages is left for vendor definition. All PCI Express switches are required to support forwarding of Vendor-Defined Messages.

PCI-X PT 2.0a includes similar functionality in the form of Device ID Messages. Again, only the basic format and handling rules are defined by PCI-X PT 2.0a with further details left for vendor definition. All PCI-X bridges that support PCI-X Mode 2 are required to support forwarding of

Device ID Messages. Forwarding support is optional in PCI-X bridges that only support PCI-X Mode 1.

Vendor-Defined and Device ID Messages may optionally be translated in format and forwarded across a PCI Express bridge in either direction when the secondary interface is operating in a PCI-X mode, according to the rules contained in the following sections. These rules include new header field definitions plus modifications to the existing baseline definitions contained in PCI Express Base 1.0a and PCI-X PT 2.0a in support of message interoperability

If message forwarding is supported, it is supported independent of whether the secondary interface is operating in PCI-X Mode 1 or PCI-X Mode 2. The Device ID Messaging Capable bit in the PCI-X Bridge Status register (Section 5.2.2.1.4) indicates whether the bridge supports the forwarding of messages. Message forwarding is not supported if the secondary interface is operating in conventional PCI mode. If message forwarding is not supported the bridge will:

- Respond to Vendor-Defined Messages received on PCI Express according to the Type-specific responses, as described in Chapter 2 of PCI Express Base 1.0a.
- Ignore Device ID Message transactions issued on the secondary interface (and allow them to end in Master-Abort).

A number of the fields in the Vendor-Defined and Device ID Message headers map directly between the two message formats or can be derived from essentially equivalent information. For other fields, equivalent information does not exist in the standard header fields but is provided by defining specific uses for bits within the respective vendor-specific fields as shown in Figure 2-1 (PCI Express for Vendor Definition field) and Figure 2-2 (PCI-X Class Specific field). Note that these additional definitions are only required for messages designed to permit forwarding across a bridge. Messages that are intended to stay entirely within their respective domains are not subject to these rules.

Figure 2-1 shows the format for a Vendor-Defined Message that is defined to permit forwarding across a bridge. Several new field definitions have been added to the message header by using locations available for vendor definition (bytes 12 to 15). The rest of the header remains as defined by PCI Express 1.0a.

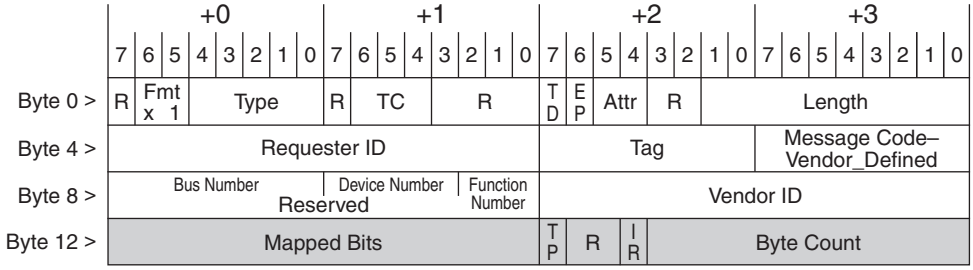

Figure 2-1: PCI Express Header Format for Vendor-Defined Message (Formatted to Permit Forwarding Across a Bridge)

The fields added to the Vendor-Defined Message header in Figure 2-1 are defined as follows:

- Mapped Bits[15:0] – byte 12 concatenated with byte 13. These bits are copied unmodified into the corresponding field of the target message format. The actual use of the Mapped Bits is left for vendor definition.

- TP (Translation Permitted) – bit 7 of byte 14. This bit must be set to 1b in Vendor-Defined Messages that support forwarding across a PCI Express bridge.
- IR (Initial Request) – bit 4 of byte 14. When set, indicates the first of a sequence of Vendor-Defined Messages making up a single logical message. Bridges translate these messages into a single Device ID Message Sequence on PCI-X.
- Byte Count[11:0] – bits 3:0 of byte 14 concatenated with byte 15. Indicates the total length in bytes of a single logical message. Bridges use this field to create the Byte Count field in the resulting Device ID Message.

In addition to the added fields, the Tag field of a Vendor-Defined Message that supports forwarding across a PCI Express bridge is defined and is required to be unique with respect to any outstanding non-posted transactions from the same requester.

Figure 2-2 shows the format for a Device ID Message that is defined to permit forwarding across a bridge. Several new field definitions have been added to the message header by using locations in the Class Specific field of the second address phase of the message. The rest of the header remains as defined by PCI-X PT 2.0a.

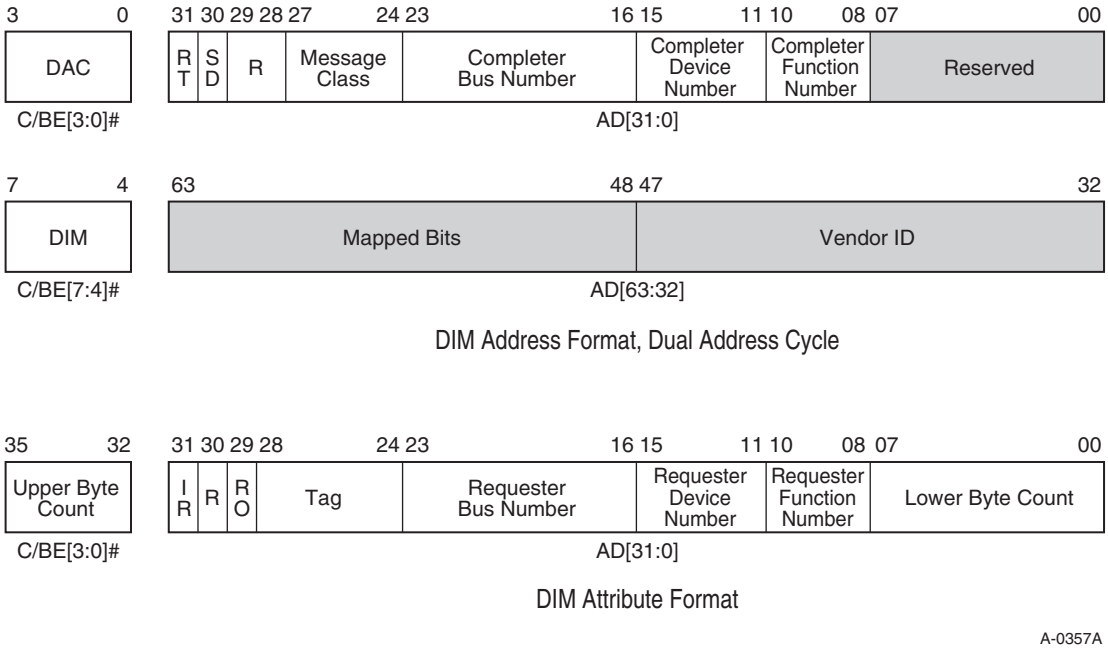

Figure 2-2: PCI-X Address and Attribute Phases for Device ID Message (Formatted to Permit Forwarding Across a Bridge)

The fields added to the Device ID Message header in Figure 2-2 are defined as follows:

- Mapped Bits[15:0] – bits 63:48 of the second address phase. These bits are copied unmodified into the corresponding field of the target message format. The actual use of the Mapped Bits is left for vendor definition.
- Vendor ID[15:0] – bits 47:32 of the second address phase. Contains the Vendor ID, as assigned by PCI-SIG, of the vendor defining the message.

- Reserved[7:0] – bits 7:0 of the first address phase. These bits are reserved for future use. They must be set to 0 by the message initiator and ignored by the receiver. PCI-X bridges forward these bits unmodified.

In addition to the added fields, Device ID Messages that support forwarding across a PCI Express bridge must set the Message Class field to 0001b.

IMPLEMENTATION NOTE

DIM Forwarding Between Secondary Interfaces in a Multi-Ported Bridge

Multi-ported PCI Express bridges that incorporate logical PCI-X bridge devices downstream of logical PCI Express bridges are required to meet the DIM forwarding requirements as stated in PCI-X PT 2.0a between logical bus segments connected by these logical bridges. Multi-ported bridges are not required to forward DIM transactions between secondary bus interfaces that are logically connected by a PCI Express primary bus in the logical bridge device hierarchy.

2.8.6.1. Forwarding from PCI Express to PCI-X

A bridge that receives a PCI Express Vendor-Defined Message destined for a device on the other side of the bridge translates the message into a PCI-X Device ID Message according to the rules provided in this section. Only Vendor-Defined Messages meeting the following requirements are permitted to be translated:

- Translation Permitted must be 1b.
- The Format/Type must be MsgD with "Routed by ID" routing. PCI-X Device ID Messages must carry a payload of at least one byte and the other PCI Express routing types are not applicable to the secondary interface of a PCI Express bridge.
- Tag[7:5] must be 000b.
- The Relaxed-Ordering Attribute must be 0b.

A message that does not meet these requirements must either be handled as an Unsupported Request or be silently discarded based on its Message Code – Type 0 and Type 1, respectively.

When a bridge creates a Device ID Message, it copies many of the fields from the Vendor-Defined Message and places them unmodified in the corresponding fields of the Device ID Message header. Other fields in the Device ID Message header are created from equivalent information in the Vendor-Defined Message. Note that a bridge must use the Dual Address Cycle format for a Device ID Message translated from a Vendor-Defined Message. Bridges are not permitted to modify the Requester ID and Tag of a forwarded message.

A bridge creates a Device ID Message header using the following rules:

- Route Type – Set based on the contents of the "r" sub-field of the Vendor-Defined Message Type field
  - Set to 0 (Explicit) for Routed by ID
- Silent Drop – Set based on the Vendor-Defined Message Message Code
  - Set to 0 for Message Code = Vendor_Defined Type 0
  - Set to 1 for Message Code = Vendor_Defined Type 1
- Message Class – Set to 0001b
- Completer Bus/Device/Function Number – Copied from the Vendor-Defined Message Bus/Device/Function Number fields
- Mapped bits – Copied from the Vendor-Defined Message Mapped Bits field
- Vendor ID – Copied from the Vendor-Defined Message Vendor ID field
- Initial Request – Copied from the Vendor-Defined Message Initial Request field
- Relaxed Ordering – Set to zero
- The Vendor-Defined No Snoop Attribute is not applicable to Device ID Messages (i.e., does not exist in the DIM header)
- Tag – Copied from the Vendor-Defined Message Tag field
- Requester Bus/Device/Function Number – Copied from the Vendor-Defined Message Requester ID field
- Upper/Lower Byte Count – Copied from the Vendor-Defined Message Byte Count field

The Initial Request and Byte Count fields are used along with the Requester ID (Bus, Device, and Function Numbers) and Tag to identify the transactions that make up a single logical message, or Sequence. All transactions making up a single Sequence must use the same Requester ID and Tag (Sequence ID). In addition, a requester must not reuse a Sequence ID until all transactions making up an earlier Sequence have been successfully transmitted on the interface.

The first transaction of a Sequence must set the Initial Request bit and the Byte Count field must reflect the total length of the Sequence. Subsequent transactions must clear the Initial Request bit and the Byte Count field must reflect the total remaining length of the Sequence. The final recipient of these message transactions may also use these fields to aid in the detection of any missing transactions.

IMPLEMENTATION NOTE

Creating Vendor-Defined Messages Longer than Max_Payload_Size

The Initial Request and Byte Count fields defined for Vendor-Defined Messages that support forwarding across a bridge are necessary to permit the forwarding of Device ID Messages that are longer than the setting of the PCI Express Max_Payload_Size parameter (PCI Express requesters must not initiate Vendor-Defined Messages with payloads larger than permitted by the setting of Max_Payload_Size). These fields provide a standardized means for Vendor-Defined Messages to carry a payload up to 4 KB in size regardless of the Max_Payload_Size setting. This methodology may also be used for Vendor-Defined Messages that are not forwarded across a PCI Express bridge.

The Vendor-Defined Message Length field is not used directly in creating a PCI-X Device ID Message but it can impact the size of any individual PCI-X Device ID Message transactions that make up a single Sequence. Bridges are permitted to combine transactions from the same Sequence into a single larger transaction but are not permitted to combine transactions from different Sequences.

The remaining fields in the Vendor-Defined Message header (e.g., Traffic Class) are not used in creating a PCI-X Device ID Message.

2.8.6.2. Forwarding from PCI-X to PCI Express

A bridge that receives a PCI-X Device ID Message destined for a device on the other side of the bridge translates the message into a PCI Express Vendor-Defined Message according to the rules provided in this section. Only messages using the Dual Address Cycle format with Message Class 1 are permitted to be translated. A message that does not meet these requirements must either be reported as an error or silently dropped as indicated by the setting of the Silent Drop bit in the message header.

When a bridge creates a Vendor-Defined Message, it copies many of the fields from the Device ID Message and places them unmodified in the corresponding fields of the Vendor-Defined message header. Other fields in the Vendor-Defined Message header are created from equivalent information in the Device ID Message. Note that a bridge must use the MsgD Format/Type for a Vendor-Defined Message translated from a Device ID Message since all Device ID Messages contain a payload. Bridges are not permitted to modify the Requester ID or Tag of a forwarded message.

A bridge creates a Vendor-Defined Message header using the following rules:

- "r" sub-field of the Type field – Set based on the Device ID Message Route Type bit
  - Set to 010b (Routed by ID) for Route Type = 0 (Explicit)
  - Set to 000b (Routed to Root Complex) for Route Type = 1 (Implicit)
- Traffic Class – Set to 000b
- No Snoop – Set to 0b

- Relaxed Ordering – Set to 0b
- Length – Set to the length of the length of the transaction in DWORDs
- Requester ID – Copied from the Device ID Message Requester Bus/Device/Function Number fields
- Tag – Copied from the Device ID Message Tag field
- Message Code – Set based on the Device ID Message Silent Drop bit
  - Set to Vendor_Defined Type 0 for SD = 0
  - Set to Vendor_Defined Type 1 for SD = 1
- Bus/Device/Function Number – Copied from the Device ID Message Completer Bus/Device/Function Number fields
- Vendor ID – Copied from the Device ID Message Vendor ID field
- Mapped bits – Copied from the Device ID Message Mapped Bits field
- Initial Request – Copied from the Device ID Message Initial Request bit
  - Note: Set Initial Request to 0b for any subsequent TLPs created for the same Device ID Message transaction due to Max_Payload_Size setting.
- Byte Count – Copied from the Device ID Message Upper/Lower Byte Count fields

The Device ID Message Initial Request and Byte Count fields are used along with the Requester Bus, Device, and Function Numbers (Requester ID) plus the Tag to identify the transactions that make up a single logical message (Sequence). All transactions making up a single Sequence use the same Requester ID and Tag (Sequence ID). A PCI-X requester must not reuse a Sequence ID until all transactions making up an earlier Sequence have completed.

PCI-X Device ID Messages support a maximum payload of 4 KB with no specified means to limit their length. PCI Express Vendor-Defined Messages, on the other hand, must not exceed the length specified in the Max_Payload_Size register. To comply with this requirement, bridges are required to fragment Device ID Message transactions larger than Max_Payload_Size into smaller Vendor-Defined Message transactions. In this case, fragments (except for the final fragment in the Sequence) must be a power of two in length and no smaller than 128 bytes. Bridges are also permitted to combine transactions from the same Sequence into a single larger transaction (up to the limit of Max_Payload_Size) but are not permitted to combine transactions from different sequences.

The first transaction of a Sequence must set the Initial Request bit and the Byte Count field must reflect the total length of the Sequence. Subsequent transactions must clear the Initial Request bit, and the Byte Count field must reflect the total remaining length of the Sequence. A recipient of these message transactions may also use these fields to aid in the detection of any missing transactions.

A bridge sets the Vendor-Defined Message Length field to indicate the actual length of a particular transaction measured in DWORDs. PCI-X permits messages to be of any length from 1 byte to 4 KB; therefore, a bridge may be required to add up to three pad bytes to the last transaction of a sequence. Any pad bytes must use a data value of 00h. Note that bridges do not modify the Byte Count field to account for any pad bytes since the pad bytes are not part of the actual message.

2.8.7. Hot Plug Signaling Messages

Hot Plug Signaling Messages are used by bridges that are implemented as the interface device on add-in cards and that support Removal Request functionality. The support requirements for bridges do not differ from the requirements placed on other PCI Express Endpoints/Legacy Endpoints as described in Chapters 2 and 6 of PCI Express Base 1.0a.

2.9. Transaction Ordering

2.9.1. Transaction Ordering Summary

Table 2-6 defines the ordering requirements for transactions that are forwarded through the bridge. The rules defined in this table apply uniformly to all types of transactions, including Memory, I/O, Configuration, and Messages.

For Table 2-6, the columns represent a first issued Transaction, and the rows represent a subsequently issued Transaction. The table entry indicates the ordering relationship between the two Transactions. The table entries are defined as follows:

- Yes–the second Transaction must be allowed to pass the first to avoid deadlock. (When blocking occurs, the second Transaction is required to pass the first Transaction. Fairness must be comprehended to prevent starvation.)
- Y/N–there are no requirements. The second Transaction may optionally pass the first Transaction or be blocked by it.
- No–the second Transaction must not be allowed to pass the first Transaction. This is required to support the Producer-Consumer strong ordering model.

Table 2-6: Ordering Rules Summary

Row Pass Column?		Posted Request	Non-Posted Request		Completion	
		Memory Write or Message Request (Col 2)	**Read Request** (Col 3)	**I/O or Configuration Write Request** (Col 4)	**Read Completion** (Col 5)	**I/O or Configuration Write Completion** (Col 6)
Posted Request	**Memory Write or Message Request** (Row A)	a) No b) Y/N	Yes	Yes	a) Y/N b) Yes	a) Y/N b) Yes
Non-Posted Request	**Read Request** (Row B)	No	Y/N	Y/N	Y/N	Y/N
	I/O or Configuration Write Request (Row C)	No	Y/N	Y/N	Y/N	Y/N
Completion	**Read Completion** (Row D)	a) No b)Y/N	Yes	Yes	a) Y/N b) No	Y/N
	I/O or Configuration Write Completion (Row E)	Y/N	Yes	Yes	Y/N	Y/N

Explanation of entries in Table 2-6:

A2 a — A Memory Write or Message Request with the Relaxed Ordering Attribute bit clear (0) must not pass any other Memory Write or Message Request.

A2 b — A Memory Write or Message Request with the Relaxed Ordering Attribute bit set (1) is permitted to pass any other Memory Write or Message Request.

A3, A4 — A Memory Write or Message Request must be allowed to pass Read Requests and I/O or Configuration Write Requests to avoid deadlocks.

A5, A6 a — Bridges may allow Memory Write and Message Requests to pass Completions or be blocked by Completions traveling in the upstream direction.

A5, A6 b — Bridges must allow Memory Write and Message Requests to pass Completions in the downstream direction to avoid deadlock.

B2, C2 — These Requests cannot pass a Memory Write or Message Request. This preserves strong write ordering required to support Producer/Consumer usage model.

B3, B4
C3, C4 — Read Requests and I/O or Configuration Write Requests are permitted to be blocked by or to pass other Read Requests and I/O or Configuration Write Requests.

B5, B6, C5,C6	These Requests are permitted to be blocked by or to pass Completions.
D2 a	If the Relaxed Ordering attribute bit is not set, a Read Completion cannot pass a previously enqueued Memory Write or Message Request.
D2 b	If the Relaxed Ordering attribute bit is set, a Read Completion is permitted to pass a previously enqueued Memory Write or Message Request.
D3, D4, E3, E4	Completions must be allowed to pass Read and I/O or Configuration Write Requests to avoid deadlocks.
D5 a	Read Completions associated with different Read Requests are allowed to be blocked by or to pass each other.
D5 b	Read Completions for one Request (will have the same Transaction ID) must return in address order.
D6	Read Completions are permitted to be blocked by or to pass I/O or Configuration Write Completions.
E2	I/O or Configuration Write Completions are permitted to be blocked by or to pass Memory Write and Message Requests. Such Transactions are actually moving in the opposite direction and, therefore, have no ordering relationship.
E5, E6	I/O or Configuration Write Completions are permitted to be blocked by or to pass Read Completions and other I/O or Configuration Write Completions.

2.9.2. Relaxed Ordering Attribute

Both PCI Express Base 1.0a and PCI-X PT 2.0a support the Relaxed Ordering attribute. A subset of PCI-X transactions support a Relaxed Ordering bit, which is conveyed in the Attribute phase (refer to PCI-X PT 2.0a for details). PCI Express transactions contain a Relaxed Ordering field in their Packet header's Transaction Descriptor field. PCI Express bridges, unlike PCI-X bridges, may optionally take advantage of the bit to relax transaction ordering requirements for writes as listed in Table 2-6. See Table 2-6 for Relaxed Ordering rules relevant to read completions. PCI Express bridges must forward this bit unmodified for all PCI-X and PCI Express transactions that contain the field, even if the bit is not used by the bridge.

Refer to PCI Express Base 1.0a and PCI-X PT 2.0a for more details on the Relaxed Ordering attribute.

2.9.3. No Snoop Attribute

Both PCI Express Base 1.0a and PCI-X PT 2.0a support a mechanism to indicate when a snoop is not required on a particular request, for hardware coherency management. PCI-X burst requests and Memory DWORD requests support the No Snoop bit, which is conveyed in the Attribute phase. PCI Express TLPs contain a No Snoop attribute field in their Transaction Descriptor field. This field is not applicable to configuration, I/O, or message transactions. PCI Express bridges do not use this bit but are required to pass it on unaltered.

This attribute does not alter transaction ordering. Refer to PCI Express Base 1.0a and PCI-X PT 2.0a for more details on this attribute.

2.10. Transaction Acceptance Rules

The following rules describe acceptance requirements placed on a PCI Express bridge when it responds as a target to transactions initiated on the secondary interface. These rules are required to avoid deadlock in a number of situations. The rules for transaction acceptance placed on a PCI Express bridge when it responds as a target to transactions initiated on the primary interface are not explicitly stated, but these rules are presented in terms of the Flow Credit advertisement and buffer allocation rules listed in Sections 2.1, 2.2, and 2.6, as well as the transaction ordering rules listed in Section 2.9.

Bridge acceptance requirements as a target of a transaction initiated on a secondary interface operating in conventional PCI mode are as follows:

- A bridge is permitted to terminate with retry or disconnect a memory write (that crosses the bridge) only if the bridge's locations for storing such transactions are full of previously posted memory write transactions moving in the same direction.
- A bridge is required to accept memory write transactions regardless of how many Delayed Transactions the bridge has enqueued.
- A bridge that has accepted a Delayed Transaction from the secondary interface is permitted to terminate with Retry a non-posted transaction on that interface until the current Delayed Transaction is complete. Bridges are permitted to execute a limited number of Delayed Transactions at a time.
- A bridge must never make the acceptance (posting) of a memory write transaction contingent on the prior completion of a non-locked transaction that the bridge initiates on the secondary interface. Bridges are allowed to refuse to accept a memory write for temporary conditions which are guaranteed to be resolved with time. A bridge can make the acceptance of a memory write transaction as a target contingent on the prior completion of locked transaction as a master only if the bridge has already established a locked operation with its intended target on the secondary interface.

Additional requirements for a secondary interface operating in a PCI-X mode (see Chapter 8 of PCI-X PT 2.0a for additional details) are as follows:

- A bridge is permitted to terminate with Retry or Disconnect at Next ADB a memory write transaction (that crosses the bridge) only if the bridge's locations for storing such transactions are full of previously posted memory write transactions moving in the same direction.
- A bridge is required to accept memory write transactions regardless of how many previous Split Transactions the bridge has enqueued.
- A bridge that has accepted a Split Transaction from the PCI-X interface (i.e., issued a Split Response on that interface) is permitted to terminate with Retry a non-posted transaction on that interface until the current Split Transaction is complete (i.e., the bridge sent all Split Completion data for the Sequence or a Split Completion Message to the requester). Bridges are permitted to execute a limited number of Split Transactions at a time.

- A bridge must never make the acceptance (posting) of a memory write transaction contingent on the prior completion of a non-locked transaction that the bridge initiates on the secondary interface. Bridges are allowed to refuse to accept a memory write for temporary conditions which are guaranteed to be resolved with time. A bridge can make the acceptance of a memory write transaction as a target contingent on the prior completion of locked transaction as a master only if the bridge has already established a locked operation with its intended target on the secondary interface.
- A bridge is permitted to terminate a Split Completion transaction with Retry or Disconnect at Next ADB when its buffers are full for one of the following reasons:
  - The value of the Split Transaction Commitment Limit field is larger than the value of the Split Transaction Capacity field in the downstream Split Transaction Control register, allowing the bridge to forward more Split Requests than it has room for Split Completions.
  - A corrupted Split Completion (i.e., a Split Completion whose size or address did not match its Split Request, or a corrupt Requester Bus Number field in the Split Completion address caused it to cross the wrong bridge) crossed the bridge in the upstream direction some time since the last assertion of reset on the primary interface.
- A bridge that supports optional Device ID Messages applies the rules for memory writes to these transactions.

2.11. Arbitration and Latency Requirements

PCI 3.0 and PCI-X PT 2.0a list requirements for conventional PCI/PCI-X interface arbitration that are relevant to a PCI Express bridge's secondary interface. The secondary interface arbitration requirements placed on PCI Express bridges are the same as the requirements placed on PCI bridges and PCI-X bridges. Additionally, PCI 3.0 places several specific latency requirements on bridges and other devices when they are the master or target of a transaction. Like the arbitration requirements, the latency requirements placed on PCI Express bridges are the same as their PCI and PCI-X bridge counterparts. Refer to PCI 3.0 and PCI-X PT 2.0a for details on these requirements.

2.12. Zero Length Memory Reads

PCI Express Base 1.0a states that a Memory Read Request of 1 DW with the 1st Byte Enable[3:0] field equal to 0h, or "zero length read," may be used by devices as a type of "flush" Request. For a Requester, the "flush" semantic allows a device to ensure that previously issued Posted Writes have been completed at their PCI Express destination. The Completion for a zero length read must specify a Length of 1 DW, and include a data payload of 1 DW. The contents of the data payload are unspecified and may be any value.

PCI Express bridges are required to forward zero length Memory Read Request TLPs received from PCI Express to a secondary interface operating in a PCI-X mode as a Memory Read DWORD transaction with no byte enables set when the request accesses non-prefetchable memory. If the secondary interface is operating in conventional PCI mode and the request accesses non-prefetchable memory, a bridge must use a Memory Read transaction with no byte enables set.

If a bridge is not assuming that all upstream accesses are to prefetchable memory and a Memory Read transaction with no byte enables set is received from conventional PCI, a bridge must forward the transaction to PCI Express as a zero length read. Additionally, bridges are required to forward Memory Read DWORD transactions with no byte enables set received from PCI-X to PCI Express as a zero length read.

2.13. Exclusive Access

Locked transaction sequences are generated by the Host CPU(s) as one or more reads followed by a number of writes to the same location(s). Bridges must not forward or initiate locks in the upstream direction towards the PCI Express interface. Support of lock propagation in the downstream direction is optional for bridges, although some system architectures may require the support to prevent deadlock scenarios. Bridges that do not support the propagation of locks to the secondary interface must respond to Memory Read Request–Locked TLPs (i.e., a TLP with Type of MRdLk) with a Completion for Locked Memory Read without Data (i.e., a TLP with Type of CplLk). The CplLk TLP must include a Completion Status of Unsupported Request.

The initiation and propagation of a locked transaction sequence through PCI Express is performed as follows:

- A locked transaction sequence begins with a MRdLk TLP. Within the PCI Express fabric, all traffic for TC0 and VC0 is blocked (by the upstream device) from using the path between the Root Complex and the locked bridge.
- Any successive reads for the locked transaction also use MRdLk TLPs.
- Completions for successful MRdLk TLPs use the CplDLk TLP Type with Completion Status of Successful Completion and completions for unsuccessful MRdLk TLPs use the CplLk TLP Type with Completion Status of Unsupported Request.
- When the CplDLk TLP for the first MRdLk TLP is returned, the bridge must not accept new requests from the secondary interface. Additionally, the bridge must not transmit requests on PCI Express that map to the default Virtual Channel (VC0) but it must continue to return completions associated with requests received from PCI Express. Also, the bridge must continue to accept completions received on either the primary or secondary interface, subject to the acceptance rules listed in Section 2.10.
- All writes for the locked sequence use MWr TLPs.
- The bridge must continue to block requests as described above until the bridge is unlocked from the primary interface by the reception of the PCI Express Unlock Message. This message is used to indicate the end of a locked sequence. If the bridge is not locked or does not support lock, the Unlock Message must be ignored and discarded.
- Once this message is received, the bridge must unlock itself and is immediately permitted to resume the acceptance of requests and deassert LOCK# on the secondary interface (and is encouraged to do so as rapidly as possible).

Refer to Chapter 6 of PCI Express Base 1.0a, Chapter 8 of PCI-X PT 2.0a, and Appendix F of PCI 3.0 for more details on exclusive accesses.

3

3. Address Spaces

PCI Express bridges support all of the address spaces defined for devices in PCI 3.0 and for bridges in PCI Bridge 1.2. The address space support requirements and options for a PCI Express bridge with a secondary interface operating in conventional PCI mode are as follows:

- Required PCI-compatible configuration address range
- Required PCI Express enhanced configuration address decoding
- Required memory-mapped I/O address range (32-bit window)
- Optional prefetchable memory address range (32- or 64-bit window)
- Optional I/O address range
- Optional VGA address decoding – Legacy support described in Section 11.1

PCI Express bridges with a secondary interface operating in one of the PCI-X modes have the additional requirement of supporting a 64-bit prefetchable memory address range via implementation of the Prefetchable Memory Base and Limit registers and the Prefetchable Base and Limit Upper 32 Bits registers. Additionally, bridges operating with a secondary interface in PCI-X Mode 2 are required to support downstream 12-bit configuration address space (see PCI-X PT 2.0a for details).

The memory and I/O address ranges are defined using a set of base and limit registers in the bridge's configuration header. The base and limit address registers define the address ranges in which a bridge forwards transactions from its primary interface to its secondary interface. These registers are effectively inversely decoded to determine the address ranges on the secondary interface of a bridge in which transactions will be forwarded upstream (from the secondary to primary interface). A bridge does not perform any address translation (a flat addressing model is used).

The following sections describe each of these address ranges in more detail. See Chapter 4 for a description of configuration space.

3.1. Memory Space

Memory transactions are forwarded across the bridge when their address falls within a window defined by one or both of the Memory Base and Memory Limit registers and the Prefetchable Memory Base and Prefetchable Memory Limit registers as described in the following sections. The bridge must treat memory transactions received on PCI Express that do not fall within the described forwarding ranges or the ranges defined by the Base Address registers as Unsupported Requests.

3.1.1. Memory-Mapped I/O Space

The memory-mapped I/O range is intended to be used to map memory address ranges of devices that are not prefetchable (i.e., have side effects on reads or non-memory-like behavior).

The Memory Base and Memory Limit registers are required for a bridge. They are used by the bridge to determine whether to forward PCI Express and conventional PCI/PCI-X memory read and memory write transactions across the bridge. The memory-mapped I/O address range defined by these registers is always aligned to a 1-MB boundary and has a size granularity of 1 MB. If the Memory Base is set to a value higher than the Memory Limit, the address range is disabled (see Section 5.1.2.8). The following register bits also affect the response by the bridge to memory transactions:

- Memory Enable bit in the Command register
- Bus Master Enable bit in the Command register
- VGA Enable bit in the Bridge Control register

The Memory Enable bit in the bridge's Command register must be set to enable the memory address range. The VGA Enable Bit is discussed in Section 5.1.2.13. More details can be found for all bits in the descriptions of the appropriate registers. The prefetchable memory address range (see Section 3.1.2) defined by the prefetchable memory address registers also affects response to memory transactions.

A bridge forwards memory transactions from its primary interface to its secondary interface (downstream) if a memory address is in the range defined by the Memory Base and Memory Limit registers (when the base is less than or equal to the limit) as illustrated in Figure 3-1. Conversely, a memory transaction on the secondary interface that is within this address range will not be forwarded upstream to the primary interface. Any memory transactions on the secondary interface that are outside this address range will be forwarded upstream to the primary interface (provided they are not in the address range defined by the set of prefetchable memory address registers).

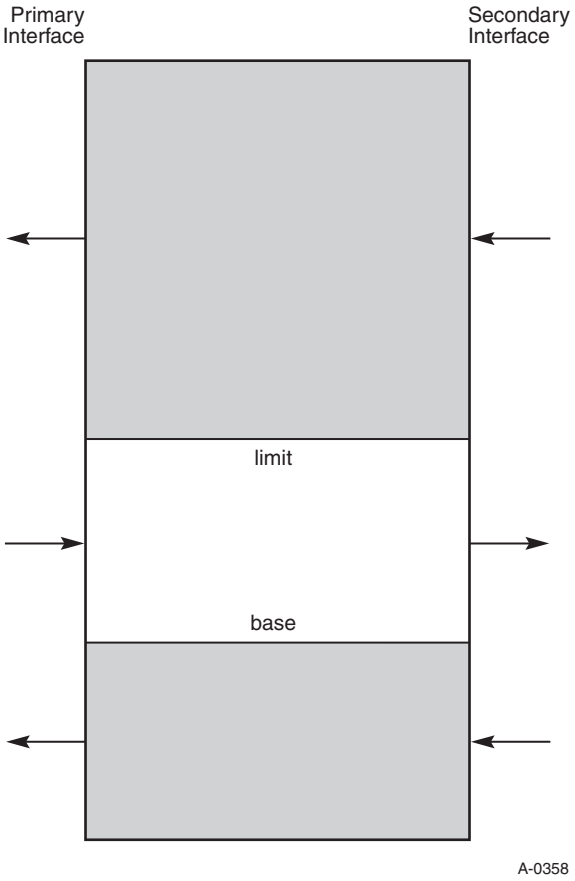

Figure 3-1: Memory Address Range Example

3.1.2. Prefetchable Space

The prefetchable memory range is intended to be used to map memory address ranges of devices that are prefetchable (i.e., have memory-like behavior and do not have side effects on reads). Support of this range is optional when the secondary interface does not support PCI-X modes. Bridges that support PCI-X modes are required to support a 64-bit prefetchable memory address range. PCI Express and PCI-X transactions use this window but, since these transactions supply a length or byte count (PCI-X DWORD transactions imply a byte count), no "prefetching" of data occurs.

The Prefetchable Memory Base, Prefetchable Memory Limit, Prefetchable Base Upper 32 Bits, and Prefetchable Limit Upper 32 Bits registers in the bridge configuration header specify an address range that is used by the bridge to determine whether to forward PCI Express and conventional PCI/PCI-X memory read and memory write transactions across the bridge. The prefetchable memory address range defined by these registers is always aligned to a 1 MB boundary and has a size granularity of 1 MB. If the address specified by the Prefetchable Memory Base and Prefetchable Base Upper 32 Bits registers is set to a value higher than the address specified by the Prefetchable Memory Limit and Prefetchable Limit Upper 32 Bits registers, the address range is disabled (see Section 5.1.2.9). The following register bits also affect the response by the bridge to memory transactions:

- Memory Enable bit in the Command register
- Bus Master Enable bit in the Command register
- VGA Enable bit in the Bridge Control register

To use the prefetchable range, a target device downstream of the bridge explicitly indicates to configuration software that a memory address range is prefetchable by setting the Prefetchable bit in the corresponding Base Address register (BAR) within the device's Configuration Space header. Configuration software uses this information to program the prefetchable memory address range in the bridge and to map the prefetchable address ranges of secondary interface targets into that range.

The Memory Enable bit in the bridge's Command register must be set to enable the prefetchable memory address range. The VGA Enable bit is discussed in Section 5.1.2.13. More details can be found for all bits in the descriptions of the appropriate registers. The memory mapped I/O address range (see Section 3.1.1) defined by the Memory Base and Memory Limit registers also affects the response to memory transactions. A bridge forwards memory transactions downstream from its primary interface to its secondary interface if a memory address is in the range defined by the Prefetchable Memory Base and Prefetchable Memory Limit registers. Conversely, a memory transaction on the secondary interface that is within this address range will not be forwarded upstream to the primary interface. Any memory transactions on the secondary interface that are outside this address range will be forwarded upstream to the primary interface (provided they are not in the address range defined by the memory mapped I/O address range registers).

Unlike non-prefetchable memory-mapped I/O memory, prefetchable memory may be located below, above, or span across the first 4-GB address boundary. Figure 3-2 illustrates a prefetchable memory window that spans across the 4-GB address boundary. Memory locations above 4 GB must be accessed using 64-bit addressing. If a bridge implements 64-bit addressing for prefetchable memory on the secondary interface of the bridge, the Prefetchable Base Upper 32 Bits and Prefetchable Limit Upper 32 Bits registers must be implemented. PCI Express memory transactions that use the Short Address (32-bit) format may target a non-prefetchable memory window or the portion of a prefetchable memory window that is below the first 4-GB address boundary. Memory transactions that use the Long Address (64-bit) format may target the portion of a prefetchable memory window that is at or above the first 4-GB address boundary.

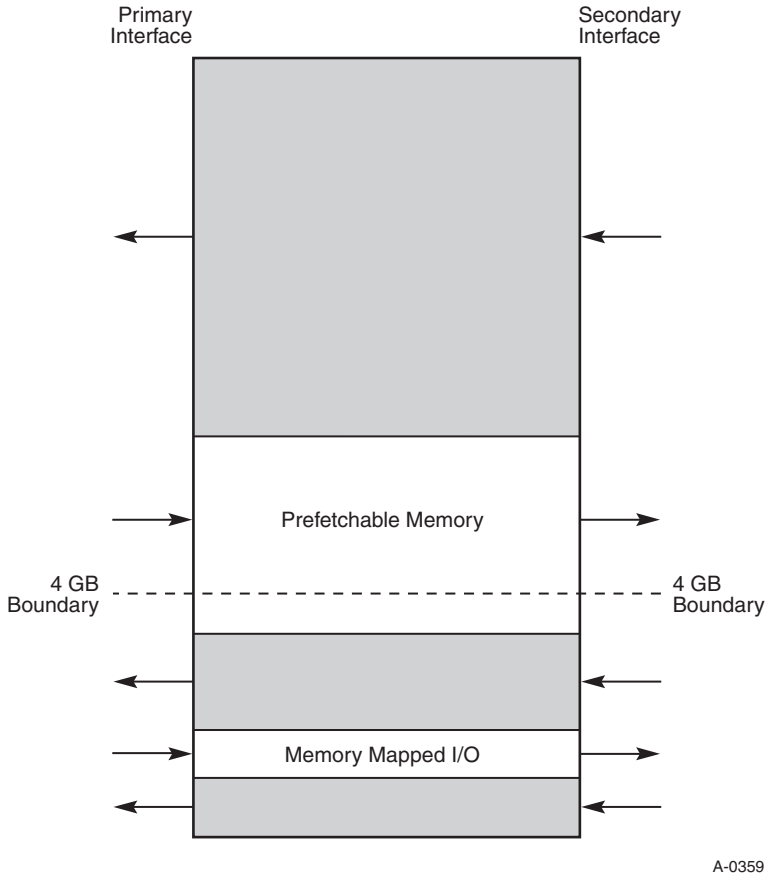

Figure 3-2: 64-bit Prefetchable Memory Address Range Example

3.2. I/O Space

The optional I/O Base, I/O Limit, I/O Base Upper 16 Bits, and I/O Limit Upper 16 Bits registers in the bridge configuration header specify an address range that is used by the bridge to determine whether to forward I/O read and I/O write transactions across the bridge. Note that if the address specified by the I/O Base and I/O Base Upper 16 Bits registers is set to a value greater than the address specified by the I/O Limit and I/O Limit Upper 16 Bits registers, the address range is disabled (see Section 5.1.2.6). The response by the bridge to I/O transactions is also affected by the register bits listed below:

- I/O Enable bit in the Command register
- Bus Master Enable bit in the Command register
- VGA Enable bit in the Bridge Control register
- ISA Enable bit in the Bridge Control register

The I/O Enable bit of the Command register must be set to enable the bridge's I/O address range (as required by PCI 3.0). The VGA Enable bit and the ISA Enable bit are discussed in later sections. More details can be found for all bits in the descriptions of the appropriate registers.

The I/O Base and I/O Limit registers are used by a bridge to determine whether to forward I/O transactions across the bridge as illustrated in Figure 3-3. The I/O address range defined by these registers is always aligned to a 4-KB boundary and has a size granularity of 4 KB. A bridge forwards I/O read and I/O write transactions from its primary interface to its secondary interface (downstream) if the address is in the range defined by the I/O base and I/O limit registers (when the base is less than or equal to the limit). Conversely, I/O transactions on the secondary bus in the address range defined by these registers are not forwarded upstream by the bridge. I/O transactions on the secondary bus that are outside the defined address range are forwarded upstream (from the secondary interface to the primary interface). If a bridge does not implement an I/O address range, the bridge must forward all I/O transactions on its secondary interface upstream to its primary interface.

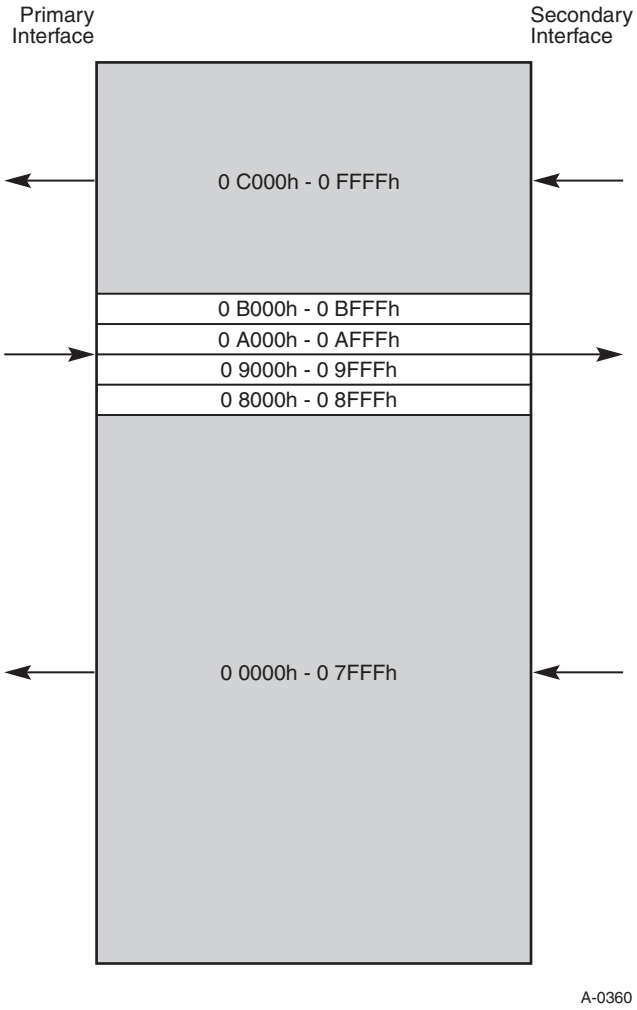

Figure 3-3: I/O Address Range Example

4

4. Configuration Mechanisms

This chapter describes the bridge's role in the forwarding and translating of configuration transactions in support of hierarchical configuration. Sections 4.1.1 and 4.1.2 describe the support provided by bridges for Type 0 and Type 1 configuration transactions. Sections 4.1.3, 4.1.4, and 4.1.5 describe in detail how PCI Express Type 1 Configuration Requests are translated into Type 1, Type 0, and Special Cycle configuration transactions. Additionally, two configuration (register addressing) mechanisms and support for the configuration retry mechanism are described.

4.1. Configuration Transactions

PCI Express Base 1.0a defines Transaction Layer Packet Types for:

- Configuration Read Type 0
- Configuration Write Type 0
- Configuration Read Type 1
- Configuration Write Type 1

PCI Express Configuration Writes and Reads possess a global format for a 3 DW header with data and without data, respectively. The Request Header format used by all PCI Express configuration transactions is shown in Figure 4-1. The FMT[1:0] and Type[4:0] fields determine which of the four types a particular configuration transaction is. A PCI Express Type 0 configuration transaction is used to access a bridge's configuration space and PCI Express Type 1 configuration transaction is used to access a device that resides downstream of a bridge.

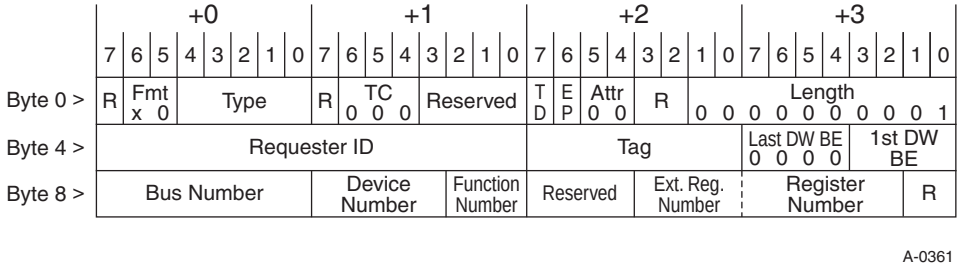

Figure 4-1: Request Header Format for Configuration Transactions

The address formats for conventional PCI and PCI-X configuration transactions are shown in Figure 4-2. A conventional PCI or PCI-X Type 0 configuration transaction is issued on the conventional PCI or PCI-X bus by the bridge to access a device on the local segment, while a Type 1 configuration transaction is issued to be forwarded downstream by a bridge residing on the local

segment. Configuration Requests are only initiated by the Root Complex in PCI Express-based systems.

Configuration space addressing fields not mentioned herein are decoded according to the respective interface specification.

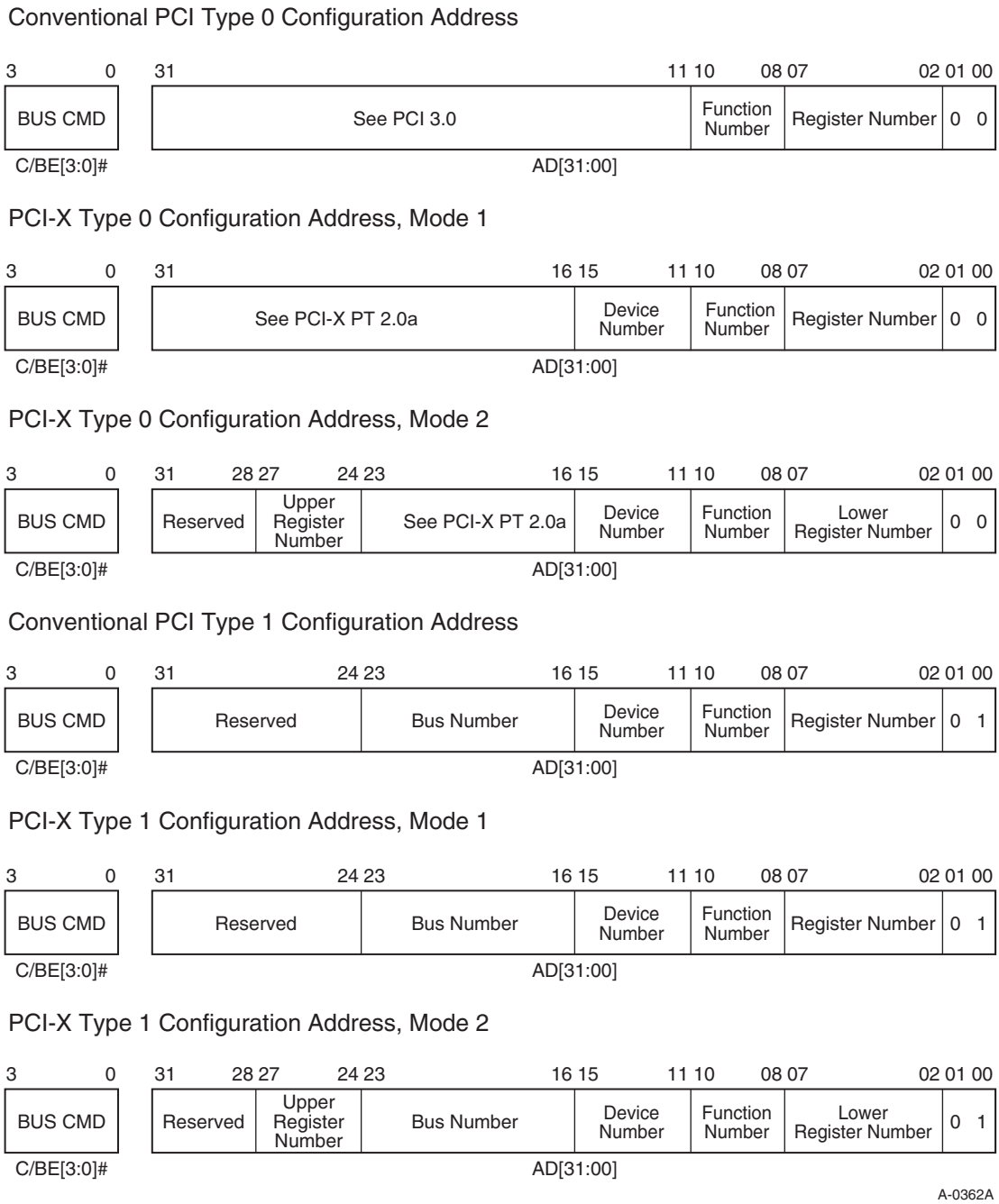

Figure 4-2: Conventional PCI and PCI-X Configuration Address Formats (From PCI-X PT 2.0a)

4.1.1. Type 0 Configuration Transactions

A PCI Express bridge only responds to Type 0 Configuration Requests on its primary PCI Express interface that address the bridge's configuration space. A Type 0 Configuration Request is used to configure the bridge and is not forwarded downstream by the bridge (from its primary to secondary interface). A bridge ignores Type 0 configuration transactions that originate on the secondary interface of the bridge.

PCI Express bridges, like PCI Express Endpoint devices, process a PCI Express Type 0 Configuration Request when the following conditions are true:

- The header Type field indicates a Configuration Read Type 0 or Configuration Write Type 0.
- The Request addresses the configuration space of any implemented function in the device.

If the Type 0 Request cannot be processed, the device must follow the rules for handling Unsupported Requests as described in PCI Express Base 1.0a.

4.1.2. Type 1 Configuration Transactions

The Bus Number field contained in the header of a PCI Express Type 1 configuration transaction request header specifies a unique PCI bus in the PCI hierarchy on which the target of the transaction resides. The bridge compares the specified Bus Number with two configuration registers that are programmed by system software and/or firmware to determine whether to forward a Type 1 configuration transaction across the bridge. The two configuration registers are listed below:

- Secondary Bus Number (configuration register offset 19h)
- Subordinate Bus Number (configuration register offset 1Ah)

If a Type 1 configuration transaction is received on the PCI Express primary interface of the bridge, the following tests are applied, in sequence, to the Bus Number field to determine how the transaction must be handled:

- If the Bus Number field is equal to the Secondary Bus Number register value and the conditions for converting the transaction into a Special Cycle transaction are met (as described in Section 4.1.5), the bridge must forward the Configuration Request to secondary interface as a Special Cycle transaction. If the conditions are not met, the bridge must forward the Configuration Request to the secondary interface as a Type 0 configuration transaction (see Section 4.1.3).
- If the Bus Number field is not equal to the Secondary Bus Number register value but is in the range of the Secondary Bus Number and Subordinate Bus Number (inclusive) registers, the Type 1 Configuration Request is specifying a Bus Number that is located behind the bridge. In this case, the bridge must forward the Configuration Request to the secondary interface as a Type 1 configuration transaction (see Section 4.1.4).
- If the Bus Number field value does not satisfy the above criteria, the Type 1 Configuration Request is specifying a Bus Number that is not located behind the bridge. In this case, the Configuration Request is invalid and the bridge should follow the rules for handling Unsupported Requests as described in PCI Express Base 1.0a.

PCI Express bridges are not permitted to claim Type 1 configuration transactions on the secondary interface.[11]

4.1.3. Type 1 to Type 0 Conversion

If a PCI Express Type 1 Configuration Request's Bus Number field is equal to the Secondary Bus Number register value and the conditions for conversion to a Special Cycle transaction are not met, the bridge must forward the Configuration Request to the secondary bus as a Type 0 configuration transaction. In this case, a device connected to the secondary interface of the bridge is the target of the resulting Type 0 configuration transaction.

To translate and convert a PCI Express Type 1 Configuration Request to a conventional PCI, PCI-X Mode 1, or PCI-X Mode 2 Type 0 configuration transaction, the bridge must do the following:

- Generate address bits AD[1:0] as 00b.
- Generate address bits AD[7:2] directly (i.e., unmodified) from the Configuration Request's Register Address field.
- Generate address AD[10:8] directly (i.e., unmodified) from the Configuration Request's Function Number field.
- For a secondary bus operating in conventional PCI, the value driven on address AD[15:11] is not specified.
- For a secondary bus operating in PCI-X Mode 1 or Mode 2, generate address AD[15:11] from the Configuration Request's Device Number field.
- For a secondary bus operating in conventional PCI or PCI-X Mode 1, the bridge must check that the received Extended Register Address field is zero. If the Extended Register Address field is non-zero, the bridge is not permitted to forward the transaction and must treat it as an Unsupported Request on PCI Express and a received Master-Abort on the destination bus. If the field is zero, the bridge must decode the PCI Express Device Number field and assert a single address bit in the range AD[31:16] during the address phase (for device numbers in the range 0 0000b-0 1111b). Refer to PCI 3.0 for more details on IDSEL decoding.
- For a secondary bus operating in PCI-X Mode 2, generate address AD[27:24] directly (i.e., unmodified) from the Configuration Request's Extended Register Address field. A single address bit in the range AD[23:16] is asserted (for device numbers in the range 0 0000b-0 0111b). Refer to PCI-X PT 2.0a for more details on IDSEL decoding. Address bits AD[31:28] are reserved and must be generated as 0000b.

When doing a Type 1 to Type 0 conversion, a PCI Express bridge is permitted to provide additional methods of IDSEL generation but must still generate AD[31:16] or AD[23:16] in accordance with the guidelines set forth in PCI 3.0 and PCI-X PT 2.0a for the respective bus modes. Board designers may use alternate methods of IDSEL generation that are independent from those

[11] PCI Express Base 1.0a prohibits the propagation of Configuration Requests from Downstream to Upstream as well as peer-to-peer.

provided by the bridge. However, the board designer must follow the interrupt routing requirements specified in Section 8.1.

4.1.4. Type 1 to Type 1 Forwarding

If a PCI Express Type 1 Configuration Request is received on the primary interface of the bridge and the value specified by the Bus Number field is within the range of Bus Numbers between the Secondary Bus Number (exclusive) and the Subordinate Bus Number (inclusive), the bridge must forward the Configuration Request to its secondary interface as a Type 1 configuration transaction. In this case, the target of the Configuration Request does not reside on the secondary interface of the bridge but is located on a bus segment further downstream. The Type 1 configuration transaction generated on the secondary bus is potentially addressing a device that resides behind other bridges that may be attached to the PCI Express bridge's secondary interface.

To translate the forwarded transaction from a PCI Express Type 1 Configuration Request to a conventional PCI, PCI-X Mode 1, or PCI-X Mode 2 Type 1 configuration transaction, the bridge must do the following:

- Generate address bits AD[1:0] as 01b.
- Generate the conventional PCI/PCI-X Register Number, Function Number, Device Number, and Bus Number (address bits AD[23:2]) directly (i.e., unmodified) from the PCI Express Configuration Request's Register Address, Function Number, Device Number, and Bus Number fields, respectively.
- For a secondary bus operating in conventional PCI or PCI-X Mode 1, the bridge must check that the received Extended Register Address field is zero. If the Extended Register Address field is non-zero, the bridge is not permitted to forward the transaction and must treat it as an Unsupported Request on PCI Express and a received Master-Abort on the destination bus. If the field is zero, the bridge must generate AD[27:24] as 0000b.
- For a secondary bus operating in PCI-X Mode 2, generate address AD[27:24] directly (i.e., unmodified) from the Configuration Request's Extended Register Address field. Address bits AD[31:28] are reserved and must be generated as 0000b.

4.1.5. Type 1 to Special Cycle Forwarding

A PCI Express bridge that receives a Type 1 Configuration Write Request transaction on its primary interface will convert it to a Special Cycle on its secondary interface when all of the following conditions are met by the Configuration Request:

- The Bus Number field matches the Secondary Bus Number register value.
- The Device Number field is all ones (equals 1 1111b).
- The Function Number field is all ones (equals 111b).
- The Register Address and Extended Register Address are both all zeros (equal 00 0000b and 0000b, respectively).

The address during a Special Cycle is ignored by conventional PCI and PCI-X devices, and the bridge is allowed to drive any value on AD[31:0] during the address phase. The data for the Special Cycle on the secondary interface is the write data received from the Type 1 Configuration Write Request on the primary interface. Note that Type 1 configuration transactions that specify conversion by a bridge to a Special Cycle are restricted to a data burst length of 1 (see PCI 3.0 for more information).

If the conditions for conversion of the Configuration Write Request to a Special Cycle are not met, the bridge follows the rules for translating and converting the Request into a Type 0 configuration transaction as specified in Section 4.1.3.

4.2. PCI 3.0-Compatible and PCI Express Enhanced Configuration Mechanisms

The bridge architecture supports two configuration models:

- PCI-compatible configuration mechanism
- PCI Express enhanced configuration mechanism

The PCI 3.0-compatible PCI Express configuration mechanism supports the PCI configuration space programming model defined in PCI 3.0. By adhering to this model, systems incorporating PCI Express interfaces remain compliant with conventional PCI bus enumeration and configuration software.

The enhanced mechanism is provided to increase the size of available configuration space and to optimize access mechanisms.

The PCI Express Enhanced Configuration Mechanism is discussed in detail for Root Complexes, Switches, and Endpoints in Chapter 7 of PCI Express Base 1.0a. The following sections provide an overview and describe the interoperability with conventional PCI and PCI-X configuration mechanisms.

4.2.1. Overview

PCI-X Mode 1 uses the same 256-byte Configuration Space as conventional PCI. PCI-X Mode 2 adds four additional bits to the Configuration Space address to expand the space to 4096 bytes and can, therefore, take advantage of the PCI Express Enhanced Configuration Mechanism. Because of the capability differences and the risk of address aliasing due to programming errors, the PCI Express bridge must enforce conditions on the forwarding of PCI Express Configuration Requests that possess a non-zero Extended Register Address field.

The rules governing the handling of PCI Express Configuration Requests with non-zero Extended Register Address fields are described in the following sections.

4.2.2. Interoperability with Secondary Bus Modes

A PCI Express bridge is permitted to forward a Type 1 Configuration Request with non-zero Extended Register Address bits received from the primary interface as a Type 0 or Type 1 configuration if the secondary interface is operating in PCI-X Mode 2. If the bridge's secondary interface is operating in conventional PCI or PCI-X Mode 1, the bridge may only forward the transaction if the Extended Register Address bits are all zero.

If a configuration transaction targets a secondary bus operating in conventional PCI or PCI-X Mode 1 and the Configuration Request's Extended Register Address field is non-zero, the bridge must treat the transaction as if it received a Master-Abort on the destination bus. That is, the bridge must do the following:

- Set the appropriate status bits for the destination bus as if the transaction had actually executed and received a Master-Abort.
- Generate a PCI Express Unsupported Request Completion.

4.3. Configuration Retry Mechanism

Bridges are required to support the configuration retry mechanism described in Chapters 2 and 6 of PCI Express Base 1.0a. The mechanism for handling PCI Express configuration retry for configuration requests that traverse the bridge from PCI Express to conventional PCI/PCI-X is described below.

Bridges are required to return a completion for all configuration requests that traverse the bridge from PCI Express to conventional PCI/PCI-X prior to expiration of the Completion Timeout timer in the Root Complex. The means by which the bridge comprehends the timer value for the Completion Timeout mechanism is beyond the scope of this specification (see Chapter 2 of PCI Express Base 1.0a for details on the Completion Timeout mechanism). This requires that bridges take ownership of all configuration requests forwarded across the bridge. If the configuration request to conventional PCI/PCI-X completes successfully prior to the bridge's timer expiration, the bridge returns a completion with Normal Status on PCI Express for that request. If the configuration request to conventional PCI/PCI-X encounters an error condition (including errors reported via PCI-X Split Completion error messages) prior to the bridge's timer expiration, the bridge returns an appropriate error completion on PCI Express. Refer to Chapter 10 for the bridge error response details. If the configuration request to conventional PCI/PCI-X does not complete either successfully or with an error, prior to timer expiration, the bridge is required to return a completion with Configuration Retry Status (CRS) on PCI Express for that request.

Even after the bridge has returned a completion with CRS on PCI Express, the bridge continues to keep the configuration transaction alive on conventional PCI/PCI-X. This requires the bridge to keep retrying the transaction until it completes on conventional PCI. For PCI-X, if the configuration request received a Split Response for the configuration transaction prior to the bridge returning a completion with CRS on PCI Express, the bridge maintains the transaction information in its queues until the Split Completion is returned. If the configuration request did not receive a Split Response for the configuration transaction prior to the bridge returning a completion with CRS, the bridge may discard the transaction and remove it from its queues or continue to retry the transaction on PCI-X until it completes successfully or with an error.

When the configuration transaction completes on conventional PCI/PCI-X after the return of a completion with CRS on PCI Express, the bridge must discard the completion information. Since discarding a completion could have negative consequences if a location was read that has read side-effects, bridges are also required to implement the Bridge Configuration Retry Enable bit (bit 15) of the Device Control register (of the PCI Express Capability List item). If this bit is cleared, the bridge will not return a completion with CRS on behalf of configuration requests forwarded across the bridge. The lack of a completion should result in eventual Completion Timeout at the Root Complex.

Bridges, by default, do not return CRS for Configuration Requests forwarded to a conventional PCI/PCI-X device behind the bridge. In the default case, the potential for variable (and potentially lengthy) completion delays on conventional PCI must be comprehended when selecting the Completion Timeout value in the Root Complex. See PCI 3.0 (Chapter 3) and PCI-X PT 2.0a for details on PCI latency requirements.

IMPLEMENTATION NOTE

Selecting a Timeout Limit for Returning Configuration Retry Completions

PCI Express Base 1.0a provides for a range of completion timeout values for requestors of non-posted transactions, from 50 µs to 50 ms. Bridges must return a completion with CRS on PCI Express for a configuration transaction forwarded to conventional PCI/PCI-X if the transaction does not complete on conventional PCI/PCI-X within 50 µs. Since the retry completion takes a certain amount of time (depending on the depth of the PCI Express hierarchy) to traverse the PCI Express hierarchy from the bridge to the Root Complex, bridges should provide a guard band for this timeout value. A typical implementation would set the default value for the timeout to be around 25 µs and provide a method to tune this timeout value within a range of 50 µs to 50 ms via software. The software tuning method is implementation specific.

5

5. Configuration Registers

Configuration register fields are assigned one of the attributes described in Table 5-1. All PCI Express bridges initialize register fields to specified default values.

Table 5-1: Register (and Register Bit-Field) Types

Register Attribute	Description
HwInit	Hardware Initialized register: Register bits are initialized by firmware or hardware mechanisms such as pin strapping or serial EEPROM. (System firmware hardware initialization is only allowed for system integrated devices.) Bits are read-only after initialization and can only be reset (for write-once by firmware) with Fundamental Reset (see PCI Express Base 1.0a, Chapter 6).
RO	Read-Only register: Register bits are read-only and cannot be altered by software. Register bits may be initialized by hardware mechanisms such as pin strapping or serial EEPROM.
RW	Read-Write register: Register bits are read-write and may be either set or cleared by software to the desired state.
RW1C	Read-Only Status, Write-1-to-Clear Status register: Register bits indicate status when read, a set bit indicating a status event may be cleared by writing a 1. Writing a 0 to RW1C bits has no effect.
ROS	Sticky - Read-only register: Registers are read-only and cannot be altered by software. Registers are not initialized or modified by hot reset. Where noted, devices that consume Aux power must preserve sticky register values when Aux power consumption (either via Aux Power PM Enable or PME Enable) is enabled. In these cases, registers are not initialized or modified by hot, warm or cold reset (see Chapter 6 of PCI Express Base 1.0a).
RWS	Sticky - Read-Write register: Registers are read-write and may be either set or cleared by software to the desired state. Bits are not initialized or modified by reset. Where noted, devices that consume Aux power must preserve sticky register values when Aux power consumption (either via Aux power or PME Enable) is enabled. In these cases, registers are not initialized or modified by hot, warm or cold reset (see Chapter 6 of PCI Express Base 1.0a).
RW1CS	Sticky - Read-only status, Write-1-to-clear status register: Registers indicate status when read, a set bit indicating a status event may be cleared by writing a 1. Writing a 0 to RW1CS bits has no effect. Bits are not initialized or modified by hot reset. Where noted, devices that consume Aux power must preserve sticky register values when Aux power consumption (either via Aux power or PME Enable) is enabled. In these cases, registers are not initialized or modified by hot, warm, or cold reset (see Chapter 6 of PCI Express Base 1.0a).

Register Attribute	Description
WT	Write-Transient register: Register bits always read 0, independent of the value written. The writing of a 1 is an event that is used to qualify the writing of other bits in the register. The writing of a 0 has no effect.
RsvdP	Reserved and Preserved: Reserved for future RW implementations. Registers are read-only and must return 0 when read. Software must preserve the value read for writes to bits.
RsvdZ	Reserved and Zero: Reserved for future RW1C implementations. Registers are read-only and must return 0 when read. Software must use 0 for writes to bits.

5.1. PCI-Compatible Configuration Registers

As in all PCI Express devices, the first 256 bytes of the configuration space in a PCI Express bridge form the PCI 3.0 compatibility region. This region may be accessed as defined in PCI 3.0 or via the enhanced PCI Express configuration access mechanism (see Chapter 7 of PCI Express Base 1.0a) without modifications to the device hardware or device driver software. This section describes the registers in the PCI 3.0 compatibility region for PCI Express bridges.

5.1.1. Common Configuration Registers

Figure 5-1 details allocation for the common register fields of PCI 3.0 Type 0 and Type 1 devices.

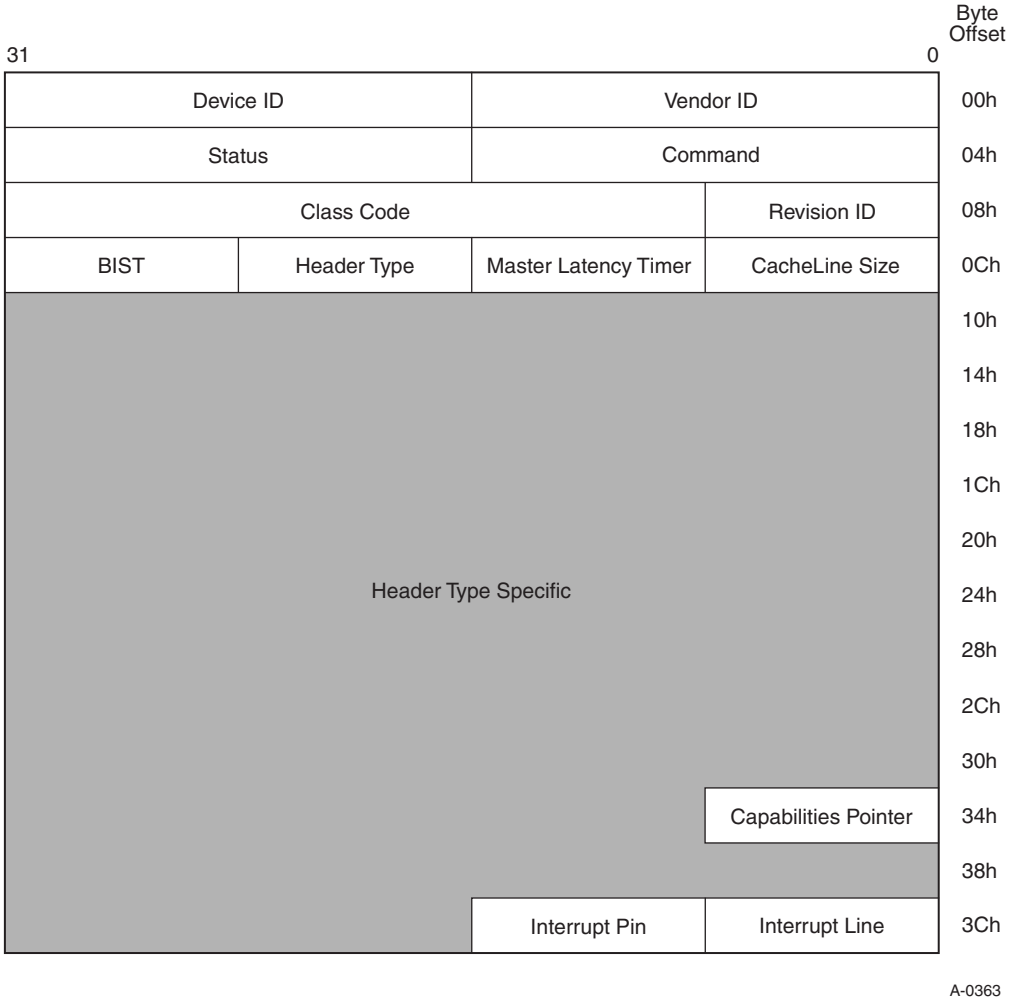

Figure 5-1: PCI 3.0 Type 0 and Type 1 Device Common Registers

These fields are defined for all PCI Express devices, including PCI Express bridges, in PCI Express Base 1.0a. The interpretation of these registers specific to PCI Express bridges is defined in this section. All registers and fields not described in this section have the same definition as in PCI Express Base 1.0a.

The PCI Express bridge-specific version of the Type 01h (bridge) configuration header format is presented in this section. Some register and bit definitions differ from those described in PCI 3.0, PCI-X PT 2.0a, and PCI Bridge 1.2, although the field names have been preserved for continuity. However, system software backward compatibility is preserved.

5.1.1.1. Command Register (Offset 04h)

Table 5-2 establishes the mapping between PCI 3.0 and this specification for the PCI 3.0 configuration space Command register.

Table 5-2: Command Register

Bit Location	Register Description	Attributes
0	**I/O Space Enable** – Controls the bridge's response as a target to I/O transactions on the primary interface that address a device that resides behind the bridge (see Section 3.2 for a description of the I/O Base and Limit registers). If the bridge does not support an I/O address range, this bit must be implemented as a read-only bit that returns 0 when read. If the bridge implements an I/O address range, this bit must be implemented as a read/write bit. The default state of this bit after reset must be 0. The state of internal transaction buffers is not specified when this bit is disabled after being enabled. The bridge can choose how it behaves when this condition occurs. Note: Software should ensure that outstanding transactions involving the bridge are completed prior to disabling this bit. Failing to quiesce transaction processing prior to disabling this bit may result in data loss and transaction processing errors. 0 - Respond to all I/O Requests on the primary interface with an Unsupported Request Completion. Forward all I/O transactions from the secondary interface to the primary interface (if supported). 1 - Enable forwarding of I/O Requests to the secondary interface.	RW

Bit Location	Register Description	Attributes
1	**Memory Space Enable** – Controls the bridge's response as a target to memory accesses on the primary interface that address a device that resides behind the bridge in both the non-prefetchable and prefetchable memory ranges (see Section 3.1) or targets a memory-mapped location within the bridge itself (see Section 5.1.2.1). A bridge must implement this bit as a read/write bit (support of a memory address range is required). The default state of this bit after reset must be 0. The state of internal transaction buffers is not specified when this bit is disabled after being enabled. The bridge can choose how it behaves when this condition occurs. Note: Software should ensure that outstanding transactions involving the bridge are completed prior to disabling this bit. Failing to quiesce transaction processing prior to disabling this bit may result in data loss and transaction processing errors. 0 - Respond to all Memory Requests on the primary interface as Unsupported Request Received. Forward all memory requests from the secondary interface to the primary interface. 1 - Enable forwarding of memory transactions to the secondary interface and any internal function.	RW
2	**Bus Master Enable** – Controls the ability of the bridge to issue memory and I/O read/write requests on the primary interface. Disabling this bit prevents the bridge from issuing any memory or I/O read/write requests on the primary interface. Note that because MSI/MSI-X transactions are in-band Memory Writes, disabling the Bus Master Enable bit disables MSI/MSI-X transactions as well. Further, note that when this bit is zero, the bridge must disable response as a target to all memory or I/O transactions on the secondary interface (they cannot be forwarded to the primary interface). This bit does not affect the issuing of completions on the primary interface or the forwarding of completions. The state of internal transaction buffers is not specified when this bit is disabled after being enabled. The bridge can choose how it behaves when this condition occurs. Note: Software should ensure that outstanding transactions involving the bridge are completed prior to disabling this bit. Failing to quiesce transaction processing prior to disabling this bit may result in data loss and transaction processing errors. 0 - Do not initiate memory or I/O transactions on the primary interface and disable response to memory and I/O transactions on secondary interface. 1 - Enable the bridge to operate as a master on the primary interface for memory and I/O transactions forwarded from the secondary interface. Default value of this field is 0.	RW

Bit Location	Register Description	Attributes
3	**Special Cycle Enable** – Does not apply to PCI Express bridges. Must be hardwired to 0.	RO
4	**Memory Write and Invalidate** – Controls the bridge's ability to translate PCI Express Memory Write Requests into PCI Memory Write and Invalidate transactions. A bridge that does not optionally promote Memory Write Requests to Memory Write and Invalidate transactions on PCI implements this bit as read-only with a value of 0. A bridge is permitted to convert a Memory Write Request into a Memory Write and Invalidate transaction when conditions for Memory Write and Invalidate command usage are met (see Section 2.4.1.1 for details). This bit is not used by the bridge when the secondary interface is in a PCI-X mode. 0 - Do not translate Memory Write Requests into PCI Memory Write and Invalidate transactions. 1 - May promote Memory Write Requests to PCI Memory Write and Invalidate transactions. If implemented, the default value of this field is 0.	RW
5	**VGA Palette Snoop** – Does not apply to PCI Express bridges. Must be hardwired to 0.	RO
6	**Parity Error Response** – Controls the bridge's setting of the Master Data Parity Error bit in the Status register in response to a received poisoned TLP from PCI Express. This bit does not control the forwarding of poisoned data from PCI Express to conventional PCI/PCI-X; i.e., a received poisoned TLP is forwarded with bad Parity/ECC to conventional PCI/PCI-X regardless of the setting of this bit. 0 - Disables the setting of the Master Data Parity Error bit. 1 - Enables the setting of the Master Data Parity Error bit. Default value of this field is 0.	RW
8	**SERR# Enable** – This bit enables reporting of non-fatal and fatal errors to the Root Complex. In addition, this bit enables transmission by the primary interface of ERR_NONFATAL and ERR_FATAL error messages on behalf of SERR# assertions detected on the secondary interface. Note that errors are reported if enabled either through this bit or through the PCI Express specific bits in the Device Control register. 0 - Disable the reporting of bridge non-fatal errors and fatal errors to the Root Complex. 1 - Enable the reporting of bridge non-fatal errors and fatal errors to the Root Complex. Default value of this field is 0.	RW
9	**Fast Back-to-Back Transactions Enable** – Does not apply to PCI Express bridges. Must be hardwired to 0.	RO

Bit Location	Register Description	Attributes
10	**Interrupt Disable** – Controls the ability of the PCI Express bridge to generate INTx interrupt messages. When set, the bridge is prevented from generating INTx interrupt messages on behalf of internal interrupt sources. This bit has no effect on INTx messages generated on behalf of INTx inputs associated with the secondary interface (if present). Bridges that do not generate INTx interrupt messages on behalf of internal sources may implement this bit as read-only with a value of 0. Any INTx interrupt messages already asserted on behalf of sources internal to the bridge must be deasserted when this bit is set. If implemented, the default value of this field is 0.	RW

5.1.1.2. Status Register (Offset 06h)

Table 5-3 establishes the mapping between PCI 3.0 and this specification for the PCI 3.0 configuration space Status register.

Table 5-3: Status Register

Bit Location	Register Description	Attributes
3	**Interrupt Status** – Indicates that an INTx interrupt message is pending on behalf of sources internal to the bridge. This bit does not reflect the status of INTx inputs associated with the secondary interface (if present). Default value of this field is 0.	RO
4	**Capabilities List** – Indicates the presence of a Capability List item. This bit indicates whether or not the bridge implements a Capabilities Pointer register pointing to a linked-list data structure of new capabilities. Since all PCI Express devices are required to implement the PCI Express capability structure, this bit must be set to 1.	RO
5	**66 MHz Capable** – Does not apply to PCI Express bridges. Must be hardwired to 0.	RO
7	**Fast Back-to-Back Transactions Capable** – Does not apply to PCI Express bridges. Must be hardwired to 0.	RO

Bit Location	Register Description	Attributes
8	**Master Data Parity Error** – This bit is used to report the detection of uncorrectable data errors by the bridge. This bit is set if the Parity Error Response bit in the Command register is set and either of the following two conditions occur: • The bridge receives a Completion with data marked poisoned on the primary interface. • The bridge poisons a write Request on the primary interface. 0 - No uncorrectable data error detected on the primary interface. 1 - Uncorrectable data error detected on the primary interface. Once set, this bit remains set until it is reset by writing a 1 to this bit location. If the Parity Error Response bit is set to zero, this bit will not be set when an error is detected. Default value of this field is 0.	RW1C
10:9	**DEVSEL# Timing** – Does not apply to PCI Express bridges. Must be hardwired to 00b.	RO
11	**Signaled Target-Abort** – This bit is set when the bridge generates a completion with Completer Abort Completion Status in response to a request received on its primary interface. 0 - Completer Abort Completion not transmitted on the primary interface. 1 - Completer Abort Completion transmitted on the primary interface. Default value of this field is 0.	RW1C
12	**Received Target-Abort** – This bit is set when the bridge receives a Completion with Completer Abort Completion Status on its primary interface. 0 - Completer Abort Completion Status not received on primary interface. 1 - Completer Abort Completion Status received on primary interface. Default value of this field is 0.	RW1C
13	**Received Master-Abort** – This bit is set when the bridge receives a Completion with Unsupported Request Completion Status on its primary interface. 0 - Unsupported Request Completion Status not received on primary interface. 1 - Unsupported Request Completion Status received on primary interface. Default value of this field is 0.	RW1C

Bit Location	Register Description	Attributes
14	**Signaled System Error** – This bit is set when the bridge sends an ERR_FATAL or ERR_NONFATAL message to the Root Complex and the SERR# Enable bit in the Command register is set. 0 - Neither ERR_FATAL nor ERR_NONFATAL transmitted on primary interface. 1 - ERR_FATAL or ERR_NONFATAL transmitted on primary interface. Default value of this field is 0.	RW1C
15	**Detected Parity Error** – This bit is set by the bridge whenever it receives a poisoned TLP or, if supported, a TLP with bad ECRC (Read Completion or Write Request) on the primary interface, regardless of the state the Parity Error Response bit in the Command register. 0 - Data poisoning and bad ECRC not detected by the bridge on its primary interface. 1 - Data poisoning or bad ECRC detected by the bridge on its primary interface. Default value of this field is 0.	RW1C

5.1.1.3. CacheLine Size Register (Offset 0Ch)

This read/write register specifies the system cacheline size in units of DWORDs. This register must be implemented by PCI Express bridges that can generate the PCI Memory Write and Invalidate command (refer to PCI 3.0). The value in this register is also used by master devices to determine whether to use Read, Read Line, or Read Multiple commands for accessing memory (refer to PCI 3.0).

Bridges that allow memory bursting from the PCI interface using cacheline wrap addressing mode (refer to PCI 3.0) must implement this register to know when a burst sequence wraps to the beginning of the cache line.

This field must be initialized to 0 at reset.

A bridge may limit the number of cacheline sizes that it can support. For example, it may accept only powers of 2 less than 128. If an unsupported value is written to the CacheLine Size register, the bridge should behave as if a value of 0 was written (i.e., it should not use the cacheline based commands or permit cacheline wrap-addressed commands to burst).

A detailed description of the operation of the CacheLine Size register is provided in PCI 3.0.

5.1.1.4. Latency Timer Register (0Dh)

This register is also referred to as primary latency timer for Type 1 Configuration Space Header devices. The primary/master latency timer does not apply to PCI Express bridges. This register must be hardwired to 0.

5.1.1.5. Header Type Register (0Eh)

The Header Type register is a read-only register used to indicate the layout for bytes 10h through 3Fh of the device's configuration space. A PCI Express bridge returns a value of 01h to indicate that the header adheres to the Configuration Space layout defined by this specification, a layout that is compatible with PCI system software developed for Type 01h PCI and PCI-X bridges (an architecture further described in PCI Bridge 1.2 and PCI-X PT 2.0a). If a bridge is a multi-function device (i.e., it integrates other functions besides the bridge), a value of 81h is returned when the Header Type register is read.

5.1.1.6. BIST Register (0Fh)

The BIST register is an optional register used for control and status reporting of built-in self test capability. A bridge that does not support BIST must implement this register as a read-only register that returns 0 when read. Table 5-4 defines the bits in the BIST register when the bridge supports it. The built-in self test must not impact the Link state of the primary interface during its execution. The bridge may respond in an indeterminate fashion to transactions that it receives while the test is in progress, unless the transaction specifically targets the BIST register. The bridge is not required to forward any transactions to the secondary interface during BIST. The effect BIST has on the secondary interface is not specified. Software cannot rely on any behavior of secondary bus devices or bridge functions (other than configuration register access of the BIST register) while BIST is active. After BIST completes, the bridge and all downstream devices must be reinitialized by software.

Table 5-4: BIST Register

Bit Location	Register Description	Attributes
3:0	**BIST Result** – A value of 0 means the device has passed its test. Non-zero values mean the bridge device failed. Device-specific failure codes can be encoded in the non-zero value.	RO
6	**Start BIST** – Write a 1 to invoke BIST. Device resets the bit when BIST is complete. Software should fail the bridge device if BIST is not complete after 2 seconds.	RW
7	**BIST Capable** – Return 1 if device supports BIST. Return 0 if the device is not BIST capable.	RO

5.1.1.7. Capabilities Pointer (34h)

This register is used to point to a linked list of additional capabilities implemented by this bridge and is mandatory since the PCI Express capability structure is required of all PCI Express bridges. Chapter 6 and Appendix H of PCI 3.0 specify the linked list data structure and present a partial list of defined capabilities, respectively. Refer to PCI Express Base 1.0a and PCI-X PT 2.0a for details on the PCI Express and PCI-X capabilities.

This register must be implemented as a read-only register. The bottom two bits of the pointer are reserved for future use and must be set to 00b. However, software cannot depend on them being zero and must mask them off before using this register as a pointer to a configuration register which holds the first entry of a linked list of new capabilities.

5.1.1.8. Interrupt Line Register (3Ch)

As in PCI 3.0, the Interrupt Line register communicates interrupt line routing information. The register is read/write and must be implemented by any device (or device function) that uses an interrupt pin (see following description). Values in this register are programmed by system software and are system architecture specific. The device itself does not use this value; rather the value in this register is used by device drivers and operating systems.

5.1.1.9. Interrupt Pin Register (3Dh)

The Interrupt Pin register is a read-only register that adheres to the definition in PCI 3.0. The Interrupt Pin register is used to indicate which interrupt virtual wire, if any, the bridges use on behalf of internal sources. This register has no relationship to interrupt messages forwarded to PCI Express as a result of MSI/MSI-X transactions or INTx interrupts from PCI devices on the secondary interface. A value of 1 corresponds to INTA virtual wire. A value of 2 corresponds to INTB virtual wire. A value of 3 corresponds to INTC virtual wire. A value of 4 corresponds to INTD virtual wire. If a bridge is not a multi-function device, it may only use the INTA virtual wire to signal INTx interrupts (if implemented) on behalf of internal sources. Bridges that do not generate INTx interrupt messages on behalf of internal sources must implement this register as read-only with a value of 0.

5.1.2. Type 01h Configuration Registers

Figure 5-2 details allocation for register fields of PCI 3.0 Type 1 Configuration Space Header for PCI Express bridges. PCI Express bridges must use a Type 01h header.

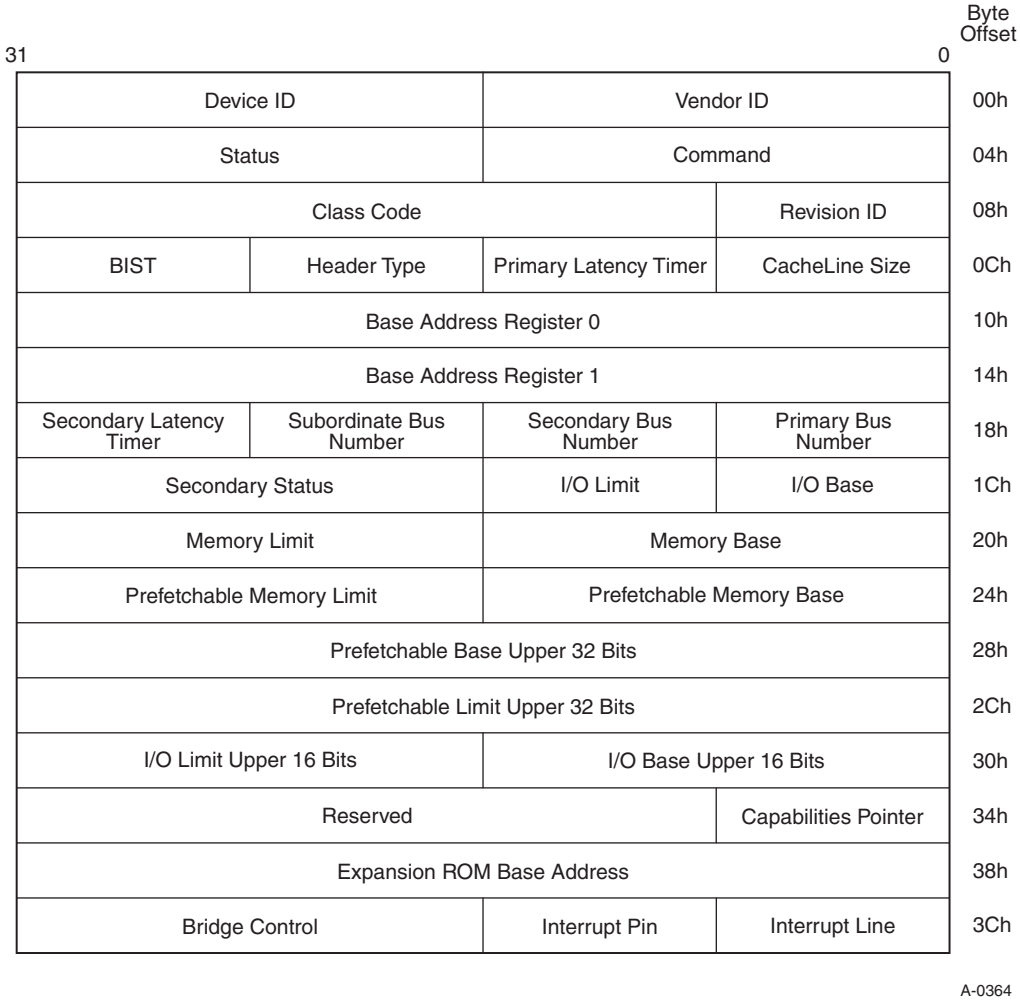

Figure 5-2: Type 1 Configuration Space Header

The PCI Express bridge-specific version of the Type 01h configuration header format is presented in this section. Some register and bit definitions differ from those described in PCI Bridge 1.2. However, system software backward compatibility is preserved.

5.1.2.1. Base Address Registers

The Base Address registers are optional registers used to map internal (device-specific) registers into Memory Space. I/O-mapped Base Address registers are prohibited in PCI Express bridges. These registers have no effect on the forwarding of transactions across a bridge as specified by the I/O, Memory, and Prefetchable Memory Base and Limit registers. Configuration software must map address ranges requested by the Base Address registers such that they are exclusive of the address ranges specified by the I/O, Memory, and Prefetchable Memory Base and Limit registers. Device-specific registers mapped by the Base Address registers must be accessible from both interfaces of the bridge.

Any implemented Base Address register is required to set the Prefetchable bit unless the range includes addresses with read side effects or addresses in which the device does not tolerate the byte merging of writes (as defined in PCI 3.0). If the Prefetchable bit is set in a Base Address register,

that Base Address register is required to support 64-bit addressing as described in PCI 3.0. Bridges that implement non-prefetchable BARs can optionally implement the 64-bit version.

Note that the bridge configuration header layout only provides two Base Address registers. Consequently, if a 64-bit memory mapping is needed, only one Base Address range can be supported (both Base Address registers will be consumed by the single 64-bit mapping).

If the bridge implements a single 32-bit Base Address register, the bridge is permitted to use either location. Base Address registers not used must be implemented as read-only registers that return 0 when read. Base Address registers must adhere to the Memory Base Address register requirements as described in PCI 3.0. Refer to PCI 3.0 for a detailed discussion of Base Address registers.

5.1.2.2. Primary Bus Number Register (18h)

The Primary Bus Number register is used to record the Bus Number of the logical PCI bus segment to which the primary interface of the bridge is connected. Configuration software is required to program the value into this register. This register is the only source that bridges use to determine the value of the Primary Bus Number. Bridges do not capture the Primary Bus Number from Type 0 Configuration Write requests. A bridge must implement this register as a read/write register and the default state after reset must be zero.[12]

5.1.2.3. Secondary Bus Number Register (19h)

The Secondary Bus Number register is used to record the Bus Number of the PCI bus segment to which the secondary interface of the bridge is connected. Configuration software programs the value in this register. The bridge uses this register to determine how to respond to a Type 1 Configuration Request on the primary interface and when to convert it to a Type 0 transaction on the secondary interface. A bridge must implement this register as a read/write register and the default state after reset must be zero.

5.1.2.4. Subordinate Bus Number Register (1Ah)

The Subordinate Bus Number register is used to record the Bus Number of the highest numbered PCI bus segment which is downstream of (or subordinate to) the bridge. Configuration software programs the value in this register. The bridge uses this register in conjunction with the Secondary Bus Number register to determine how to respond to a Type 1 Configuration Request on the primary interface and when to pass it to the secondary interface. A bridge must implement this register as a read/write register and the default state after reset must be zero.

[12] An exception is granted to a bridge that is always connected to bus segment 0 by design or implementation. For example, this occurs when a PCI Express to PCI/PCI-X bridge is integrated into a host bus bridge that generates bus segment 0. In this case, the bridge may implement this register as a read-only register that returns 0 when read.

5.1.2.5. Secondary Latency Timer Register (1Bh)

The Secondary Latency Timer register adheres to the definition of the Latency Timer in PCI 3.0 but applies only to the secondary interface of a bridge. A bridge that supports a burst transfer of more than two data phases on its secondary interface must implement the Secondary Latency Timer as a read/write register. The implementation is permitted to limit the granularity to eight PCI clocks by hardwiring the low three bits to 0. A bridge which does not support a burst transfer of more than two data phases on its secondary interface is permitted to hardwire the Secondary Latency Timer to a value of 16 or less. If implemented as a read/write register, the default value of the Secondary Latency Timer after reset must be 0.

The Latency Timer operation in PCI-X mode differs from its operation in conventional PCI mode in two areas: the default value in PCI-X mode and the restrictions on ending the current transaction. Refer to PCI-X PT 2.0a for further detail on its operation.

5.1.2.6. I/O Base Register and I/O Limit Register (1Ch)

PCI Express bridges support I/O Space for compatability with legacy devices that require its use. The I/O Base and I/O Limit registers are optional and define an address range that is used by the bridge to determine when to forward I/O transactions from one interface to the other. If a bridge does not implement an I/O address range, both the I/O Base and I/O Limit registers must be implemented as read-only registers that return zero when read. If a bridge supports an I/O address range, these registers must be initialized by configuration software so default states are not specified.

PCI Express uses the 32-bit Short Address Format (DWORD-aligned) for I/O transactions. If a bridge implements an I/O address range, the upper 4 bits of both the I/O Base and I/O Limit registers are writable and correspond to Address[15:12] of the PCI Express I/O Request. For the purpose of address decoding, the bridge assumes that the lower 12 address bits of the I/O base address (not implemented in the I/O Base register) are zero. Similarly, the bridge assumes that the lower 12 address bits of the I/O limit address (not implemented in the I/O Limit register) are FFFh. Thus, the bottom of the defined I/O address range will be aligned to a 4-KB boundary and the top of the defined I/O address range will be one less than a 4-KB boundary. The I/O Limit register can be programmed to a smaller value than the I/O Base register, if there are no I/O addresses on the secondary side of the bridge. In this case, the bridge will not forward any I/O transactions from the primary interface to the secondary and will forward all I/O transactions from the secondary bus to the primary interface.

The lower four bits of both the I/O Base and I/O Limit registers are read-only, contain the same value, and encode the I/O addressing capability of the bridge according to Table 5-5.

Table 5-5: I/O Addressing Capability

Register Bits [3:0]	I/O Addressing Capability
00h	16-bit I/O Addressing
01h	32-bit I/O Addressing
02h - 0Fh	reserved

If the lower four bits of the I/O Base and I/O Limit registers have the value 0h, the bridge supports only 16-bit I/O addressing (for ISA compatibility), and for the purpose of address decoding, the bridge assumes that the upper 16 address bits, PCI Express Address[31:16], of the I/O base and I/O limit address (not implemented in the I/O Base and I/O Limit registers) are zero. Note that the bridge must still perform a full 32-bit decode of the I/O address as required by PCI 3.0 (i.e., check that Address[31:16] is 0000h in the PCI Express header field). In this case, the I/O address range supported by the bridge will be restricted to the first 64 KB of I/O Space (0000 0000h to 0000 FFFFh).

If the low four bits of the I/O Base and I/O Limit registers are 01h, the bridge supports 32-bit I/O address decoding, and the I/O Base Upper 16 Bits and the I/O Limit Upper 16 Bits hold the upper 16 bits, corresponding to Address[31:16] of the PCI Express I/O Request, of the 32-bit I/O Base and I/O Limit addresses respectively. In this case, system configuration software is permitted to locate the I/O address range supported by the anywhere in the 4-GB I/O Space. Note that the 4-KB alignment and granularity restrictions still apply when the bridge supports 32-bit I/O addressing.

5.1.2.7. Secondary Status Register (1Eh)

Table 5-6 presents the register description for the Secondary Status register.

Table 5-6: Secondary Status Register

Bit Location	Register Description	Attributes
4:0	**Reserved**	RsvdZ
5	**66 MHz Capable** – This bit indicates whether or not the secondary interface of the bridge is capable of operating at 66 MHz in conventional PCI mode. Support of 66 MHz operation by a bridge is optional. 0 - The secondary interface is not capable of 66 MHz operation. 1 - The secondary interface is capable of 66 MHz operation.	RO
6	**Reserved**	RsvdZ

Bit Location	Register Description	Attributes
7	**Fast Back-to-Back Transactions Capable** – This bit indicates whether or not the secondary interface of the bridge is capable of decoding fast back-to-back transactions when the transactions are from the same master but to different targets. (A bridge is required to support fast back-to-back transactions from the same master.) 0 - The secondary interface is not capable of decoding fast back-to-back transactions to different targets. 1 - The secondary interface is capable of decoding fast back-to-back transaction to different targets.	RO
8	**Master Data Parity Error** – This bit is used to report the detection of an uncorrectable data error by the bridge. This bit is set if the bridge is the bus master of the transaction on the secondary interface, the Parity Error Response Enable bit in the Bridge Control register is set, and either of the following two conditions occur: • The bridge asserts PERR# on a read transaction. • The bridge detects PERR# asserted on a write transaction. In addition, when the secondary interface is in a PCI-X mode, the bridge will set this bit if either of the following occur: • The bridge detects an uncorrectable data error in a Split Completion or Split Completion Message. • The bridge receives a Split Completion Message for a non-posted write indicating an Uncorrectable (Split) Write Data Error. Once set, this bit remains set until it is reset by writing a 1 to this bit location. If the Parity Error Response Enable bit is set to zero, this bit will not be set when an error is detected. 0 - No uncorrectable data error detected on the secondary interface. 1 - Uncorrectable data error detected on the secondary interface. Default value of this field is 0.	RW1C
10:9	**DEVSEL Timing** – This bit field encodes the timing of the secondary interface DEVSEL# as listed below. The encoding must indicate the slowest response time that the bridge uses to assert DEVSEL# on its secondary interface when it is responding as a target in conventional PCI mode to any transaction except a Configuration Read or Configuration Write. 00 - Fast DEVSEL# decoding 01 - Medium DEVSEL# decoding 10 - Slow DEVSEL# decoding 11 - Reserved	RO

Bit Location	Register Description	Attributes
11	**Signaled Target-Abort** – This bit reports the signaling of a Target-Abort termination by the bridge when it responds as the target of a transaction on its secondary interface or when it signals a PCI-X Split Completion with Target-Abort. 0 - Target-Abort not signaled on secondary interface. 1 - Target-Abort signaled on secondary interface. Default value of this field is 0.	RW1C
12	**Received Target-Abort** – This bit reports the detection of a Target-Abort termination by the bridge when it is the master of a transaction on its secondary interface or when receiving a PCI-X Split Completion Message indicating Target-Abort. 0 - Target-Abort not detected on secondary interface. 1 - Target-Abort detected on secondary interface. Default value of this field is 0.	RW1C
13	**Received Master-Abort** – This bit reports the detection of a Master-Abort termination by the bridge when it is the master of a transaction on its secondary interface or when receiving a PCI-X Split Completion Message indicating Master-Abort. 0 - Master-Abort not detected on secondary interface. 1 - Master-Abort detected on secondary interface. Default value of this field is 0.	RW1C
14	**Received System Error** – This bit reports the detection of an SERR# assertion on the secondary interface of the bridge. 0 - SERR# assertion on the secondary interface has not been detected. 1 - SERR# assertion on the secondary interface has been detected. Default value of this field is 0.	RW1C

Bit Location	Register Description	Attributes
15	**Detected Parity Error** – This bit reports the detection of an uncorrectable address, attribute, or data error by the bridge on its secondary interface. This bit must be set when any of the following three conditions are true: • The bridge detects an uncorrectable address or attribute error as a potential target. • The bridge detects an uncorrectable data error when it is the target of a write transaction or PCI-X Split Completion. • The bridge detects an uncorrectable data error when it is the master of a read transaction (immediate read data or PCI-X split response). The bit is set irrespective of the state of the Parity Error Response Enable bit in the Bridge Control register. 0 - Uncorrectable address, attribute, or data error not detected on secondary interface 1 - Uncorrectable address, attribute, or data error detected on secondary interface Default value of this field is 0.	RW1C

5.1.2.8. Memory Base and Memory Limit Registers (20h and 22h)

The Memory Base and Memory Limit registers are required registers that define a (non-prefetchable) memory mapped I/O address range which is used by the bridge to determine when to forward memory transactions from one interface to the other (see Section 3.1 for additional details).

The upper 12 bits of both the Memory Base and Memory Limit registers are read/write and correspond to the upper 12 address bits (Address[31:20] on PCI Express and AD[31:20] on PCI) of 32-bit addresses. For the purpose of address decoding, the bridge assumes that the lower 20 address bits of the memory base address (not implemented in the Memory Base register) are zero. Similarly, the bridge assumes that the lower 20 address bits of the memory limit address (not implemented in the Memory Limit register) are F FFFFh. Thus, the bottom of the defined memory address range will be aligned to a 1-MB boundary and the top of the defined memory address range will be one less than a 1-MB boundary.

It is recommended that the Memory Limit register be programmed to a smaller value than the Memory Base register if memory-mapped I/O addresses will not be used on the secondary side of the bridge. (In some cases, software may desire to leave the memory I/O address range open even if the addresses are not being used.) As such, the bridge will not forward any memory transactions from the primary interface to the secondary interface and will forward all memory transactions from the secondary interface to the primary interface.

The bottom four bits of the Memory Base and Memory Limit registers are read-only and return zeros when read.

These registers must be initialized by configuration software so default states are not specified.

5.1.2.9. Prefetchable Memory Base and Prefetchable Memory Limit Registers (24h and 26h)

The Prefetchable Memory Base and Prefetchable Memory Limit registers are required for bridges that support PCI-X modes and are optional for bridges that only support conventional PCI mode. These registers define a prefetchable memory address range which is used by the bridge to determine when to forward memory transactions from one interface to the other (see Section 3.1 for additional details).

The upper 12 bits of the register are read/write and correspond to the upper 12 address bits, PCI Express Address[31:20] and AD[31:20] on PCI, of 32-bit addresses. For the purpose of address decoding, the bridge assumes that the lower 20 address bits of the prefetchable memory base address (not implemented in the Prefetchable Memory Base register) are zero. Similarly, the bridge assumes that the lower 20 address bits of the prefetchable memory limit address (not implemented in the Prefetchable Memory Limit register) are F FFFFh. Thus, the bottom of the defined prefetchable memory address range will be aligned to a 1-MB boundary and the top of the defined memory address range will be one less than a 1-MB boundary.

If the address specified by the Prefetchable Memory Base and Prefetchable Base Upper 32 Bits registers is set to a value higher than the address specified by the Prefetchable Memory Limit and Prefetchable Limit Upper 32 Bits registers, the address range is disabled. If the prefetchable memory and memory-mapped I/O ranges are disabled (see Section 5.1.2.8), the bridge will not forward any memory transactions from the primary interface to the secondary and will forward all memory transactions from the secondary interface to the primary interface.

The bottom 4 bits of both the Prefetchable Memory Base and Prefetchable Memory Limit registers are read-only, contain the same value, and encode whether or not the bridge supports 64-bit addresses. If these 4 bits have the value 0h, the bridge supports only 32-bit addresses. If these 4 bits have the value 01h, the bridge supports 64-bit addresses (as required for bridges that support PCI-X modes) and the Prefetchable Base Upper 32 Bits and Prefetchable Limit Upper 32 Bits registers hold the rest of the 64-bit prefetchable base and limit addresses respectively. All other encodings are reserved.

These registers must be initialized by configuration software so default states are not specified.

5.1.2.10. Prefetchable Base Upper 32 Bits and Prefetchable Limit Upper 32 Bits Registers (28h and 2Ch)

The Prefetchable Base Upper 32 Bits and Prefetchable Limit Upper 32 Bits registers are required extensions to the Prefetchable Memory Base and Prefetchable Memory Limit registers for bridges that support PCI-X modes and are optional for bridges that only support conventional PCI mode. The Prefetchable Base Upper 32 Bits and Prefetchable Limit Upper 32 Bits registers are read/write registers that specify the upper 32-bits, corresponding to PCI Express Address[63:32] and AD[63:32] on PCI, of the 64-bit base and limit addresses which specify the prefetchable memory address range (see Section 3.1.2 for additional details).

These registers must be initialized by configuration software so default states are not specified. If these registers are not implemented, they must be read-only and return zero when read.

5.1.2.11. I/O Base Upper 16 Bits and I/O Limit Upper 16 Bits Registers (30h and 32h)

The I/O Base Upper 16 Bits and I/O Limit Upper 16 Bits registers are optional extensions to the I/O Base and I/O Limit registers. If the I/O Base and I/O Limit registers indicate support for 16-bit I/O address decoding, the I/O Base Upper 16 Bits and I/O Limit Upper 16 Bits registers are implemented as read-only registers which return zero when read. If the I/O Base and I/O Limit registers indicate support for 32-bit I/O addressing, the I/O Base Upper 16 Bits and I/O Limit Upper 16 Bits registers must be initialized by configuration software so default states are not specified.

If 32-bit I/O address decoding is supported, the I/O Base Upper 16 Bits and the I/O Limit Upper 16 Bits register specify the upper 16 bits, corresponding to PCI Express Address[31:16] and AD[31:16] on PCI, of the 32-bit base and limit addresses respectively, that specify the I/O address range (see Chapter 3 for additional details).

5.1.2.12. Expansion ROM Base Address Register (38h)

The Expansion ROM Base Address register is an optional register that adheres to the definition contained in PCI 3.0. Note, however, that the offset of the register within the Type 1 configuration header for a bridge is different than that of the Type 0 configuration header specified in PCI 3.0.

5.1.2.13. Bridge Control Register (3Eh)

The Bridge Control register provides extensions to the Command register that are specific to a bridge. The Bridge Control register provides many of the same controls for the secondary interface that are provided by the Command register for the primary interface. There are some bits that affect the operation of both interfaces of the bridge. Definitions for each bit are specified in Table 5-7.

Table 5-7: Bridge Control Register

Bit Location	Register Description	Attributes
0	**Parity Error Response Enable** – Controls the bridge's response to uncorrectable address, attribute, and data errors on the secondary interface. If this bit is set, the bridge must take its normal action when an uncorrectable address, attribute, or data error is detected (refer to Section 10.2.1 for details on the actions that the bridge must take when this bit is set). If this bit is cleared, the bridge must ignore any uncorrectable address, attribute, and data errors that it detects and continue normal operation. A bridge must generate parity (or ECC, if applicable) even if parity error reporting is disabled. Also, a bridge must always forward data with poisoning, from conventional PCI/PCI-X to PCI Express on an uncorrectable conventional PCI/PCI-X data error, regardless of the setting of this bit. 0 - Ignore uncorrectable address, attribute, and data errors on the secondary interface. 1 - Enable uncorrectable address, attribute, and data error detection and reporting on the secondary interface. Default value of this field is 0.	RW
1	**SERR# Enable** – Controls the forwarding of secondary interface SERR# assertions to the primary interface. The bridge will transmit an ERR_FATAL or ERR_NONFATAL cycle (see Chapter 10 for the mapping of errors to severity level) on the primary interface when all of the following are true: • SERR# is asserted on the secondary interface. • This bit is set or Advanced Error Reporting is supported and the SERR# Assertion Detected Mask bit is clear in the Secondary Uncorrectable Error Mask register. • The SERR# Enable bit is set in the Command register or the PCI Express-specific bits are set (refer to Chapter 10 for details) in the Device Control register of the PCI Express Capability Structure. 0 - Disable the forwarding of SERR# from the secondary interface to ERR_FATAL and ERR_NONFATAL (SERR# might still be forwarded if the SERR Advanced Error mask bit is cleared). Refer to Section 5.2.3.2 for details. 1 - Enable the forwarding of secondary SERR# to ERR_FATAL or ERR_NONFATAL. Default value of this field is 0.	RW

Bit Location	Register Description	Attributes
2	**ISA Enable** – Modifies the response by the bridge to ISA I/O addresses. This applies only to I/O addresses that are enabled by the I/O Base and I/O Limit registers and are in the first 64 KB of PCI I/O address space (0000 0000h to 0000 FFFFh). If this bit is set, the bridge will block any forwarding from primary to secondary of I/O transactions addressing the last 768 bytes in each 1-KB block. In the opposite direction (secondary to primary), I/O transactions will be forwarded if they address the last 768 bytes in each 1-KB block. See Section 11.2 for further details. 0 - Forward downstream all I/O addresses in the address range defined by the I/O Base and I/O Limit registers. 1 - Forward upstream ISA I/O addresses in the address range defined by the I/O Base and I/O Limit registers that are in the first 64 KB of PCI I/O address space (top 768 bytes of each 1-KB block). Default value of this field is 0.	RW
3	**VGA Enable (Optional)** – Modifies the response of the bridge to VGA-compatible addresses. If this bit is set, the bridge will forward the following accesses on the primary interface to the secondary interface (and, conversely, block the forwarding of these addresses from the secondary to primary interface): • Memory accesses in the range 000A 0000h to 000B FFFFh • I/O addresses in the first 64 KB of the I/O address space (Address[31:16] for PCI Express are 0000h) and where Address[9:0] is in the range of 3B0h to 3BBh or 3C0h to 3DFh (inclusive of ISA address aliases - Address[15:10] may possess any value and is not used in the decoding) If the VGA Enable bit is set, forwarding of VGA addresses is independent of the value of the ISA Enable bit (located in the Bridge Control register), the I/O address range and memory address ranges defined by the I/O Base and Limit registers, the Memory Base and Limit registers, and the Prefetchable Memory Base and Limit registers of the bridge. The forwarding of VGA addresses is qualified by the I/O Enable and Memory Enable bits in the Command register. 0 - Do not forward VGA compatible memory and I/O addresses from the primary to the secondary interface (addresses defined above) unless they are enabled for forwarding by the defined I/O and memory address ranges. 1 - Forward VGA compatible memory and I/O addresses (addresses defined above) from the primary interface to the secondary interface (if the I/O Enable and Memory Enable bits are set) independent of the I/O and memory address ranges and independent of the ISA Enable bit. Default value of this field is 0. If this bit is not implemented, it must be read-only and return a value of 0 when read.	RW

Bit Location	Register Description	Attributes
4	**VGA 16-bit Decode (Optional)** – This bit enables the bridge to provide 16-bit decoding of VGA I/O address precluding the decoding of alias addresses every 1 KB. This bit only has meaning if the VGA Enable bit in this register is also set to 1, enabling VGA I/O decoding and forwarding by the bridge. This read/write bit enables system configuration software to select between 10- and 16-bit I/O address decoding for all VGA I/O register accesses that are forwarded from primary to secondary whenever the VGA Enable bit is set to 1. 0 - Execute 10-bit address decodes on VGA I/O accesses. 1 - Execute 16-bit address decodes on VGA I/O accesses. Default value of this field is 0. This bit must be implemented if the VGA Enable bit is implemented. If this bit is not implemented, it must be read-only and return a value of 0 when read.	RW
5	**Master-Abort Mode** – Controls the behavior of a bridge when it receives a Master-Abort termination (e.g., an Unsupported Request on PCI Express) on either interface. This bit does not affect the behavior of the bridge when forwarding a UR completion from PCI Express to the secondary interface if the secondary interface is operating in a PCI-X mode. 0 - Do not report Master-Aborts. When a UR response is received from PCI Express for non-posted transactions, and when the secondary side is operating in conventional PCI mode, return FFFF FFFFh on reads and complete I/O writes normally. When a Master-Abort is received on the secondary interface for posted transactions initiated from the primary interface, no action is taken (i.e., all data is discarded). 1 - Report UR Completions from P CI Express by signaling Target-Abort on the secondary interface when the secondary interface is operating in conventional PCI mode. For posted transactions initiated from the primary interface and Master-Aborted on the secondary interface, the bridge must return an ERR_NONFATAL (by default) or ERR_FATAL transaction (provided the SERR# Enable bit is set in the Command register). The severity is selectable only if Advanced Error Reporting is supported. Default value of this field is 0.	RW

Bit Location	Register Description	Attributes
6	**Secondary Bus Reset** – Forces the assertion of RST# on the secondary interface. The secondary RST# will be asserted by the bridge whenever this bit is set. The bridge's secondary bus interface and any buffers between the two interfaces (primary and secondary) must be initialized back to their default state whenever this bit is set. The primary bus interface and all configuration space registers must not be affected by the setting of this bit. Because RST# is asserted for as long as this bit is set, software must observe proper conventional PCI and PCI-X reset timing requirements. 0 - Do not force the assertion of the secondary interface RST#. 1 - Force the assertion of the secondary interface RST#. Default value of this field is 0.	RW
7	**Fast Back-to-Back Enable –** When in conventional PCI mode, controls ability of the bridge to generate fast back-to-back transactions to different devices on the secondary interface. A bridge that cannot generate fast back-to-back transactions must implement this bit as a read-only bit that returns 0 when read. During system initialization, configuration software will set this bit if all devices on the secondary interface are capable of fast back-to-back operation. 0 - Disable generation of fast back-to-back transactions on the secondary interface. 1 - Enable generation of fast back-to-back transactions on the secondary interface. Default value of this field is 0.	RW
8	**Primary Discard Timer** – Does not apply to PCI Express. Must be hardwired to 0.	RO
9	**Secondary Discard Timer** – When in conventional PCI mode, elects the number of PCI clocks that the bridge will wait for a master on the secondary interface to repeat a Delayed Transaction request (see PCI Bridge 1.2 for more details). The counter starts once the Completion (PCI Express Completion associated with the Delayed Transaction Request) has reached the head of the downstream queue of the bridge (i.e., all ordering requirements have been satisfied and the bridge is ready to complete the Delayed Transaction with the originating master on the secondary bus). If the originating master does not repeat the transaction before the counter expires, the bridge will delete the Delayed Transaction from its queue and set the Discard Timer Status bit. 0 - The Secondary Discard Timer counts 2^{15} PCI clock cycles. 1 - The Secondary Discard Timer counts 2^{10} PCI clock cycles. Default value of this field is 0.	RW

Bit Location	Register Description	Attributes
10	**Discard Timer Status** – Applies only in conventional PCI mode. This bit is set to a 1 when the Secondary Discard Timer expires and a Delayed Completion is discarded from a queue in the bridge. 0 - No discard timer error. 1 - Discard timer error.	RW1C
11	**Discard Timer SERR# Enable** – Applies only in conventional PCI mode. This bit enables the bridge to generate either an ERR_NONFATAL (by default) or ERR_FATAL transaction on the primary interface when the Secondary Discard Timer expires and a Delayed Transaction is discarded from a queue in the bridge. The severity is selectable only if Advanced Error Reporting is supported. 0 - Do not generate ERR_NONFATAL or ERR_FATAL on the primary interface as a result of the expiration of the Secondary Discard Timer. Note that an error message can still be sent if Advanced Error Reporting is supported and the Delayed Transaction Discard Timer Expired Mask bit is clear. 1 - Generate ERR_NONFATAL or ERR_FATAL on the primary interface if the Secondary Discard Timer expires and a Delayed Transaction is discarded from a queue in the bridge. Default value of this field is 0.	RW
15:12	**Reserved**	RsvdP

IMPLEMENTATION NOTE

Integrating Additional Functionality with a Bridge

In some applications, it is desirable to integrate other features with a bridge device (e.g. a DMA engine) that are beyond the scope of this specification. The additional features can be implemented as a separate device-function (i.e., a multi-function device with the bridge). This second function would use a Type 00h Configuration Space header and is subject to the requirements set for PCI Express devices in PCI Express Base 1.0a. In this implementation, the system is able to effectively manage the additional function and the transactions that it initiates.

5.1.3. PCI-X Effects on the Configuration Header

PCI Express bridges include the standard Type 01h Configuration Space header defined in Section 5.1.2. In conventional PCI mode, all of these registers function as specified in Section 5.1.2. If the secondary interface of the bridge is initialized to a PCI-X mode, the requirements for these registers change as follows:

- Secondary Bus Number register: System configuration software must not change the value in the Secondary Bus Number register while secondary devices have incomplete Split Transactions anywhere in the system. This is generally done by changing the Secondary Bus Number registers only when the system is being initialized (before device drivers load), or after all devices have been quiesced (for a hot-plug operation), or the secondary RST# signal from the bridge is asserted. After the Secondary Bus Number register is changed, system configuration software must execute at least one Configuration Write transaction to each device on the bridge's secondary bus. (This initializes the Bus Number registers in the secondary devices.)
- Secondary Latency Timer register: The default value of the appropriate Secondary Latency Timer register is 64 if the secondary interface is in PCI-X mode. (See PCI-X PT 2.0a for details.)
- CacheLine Size register: The contents of this register are ignored if the secondary interface is in PCI-X mode. If the secondary interface is in conventional PCI mode, the bridge continues to use this register as defined in Section 5.1.1.3.
- Bridge Control register:
  - Parity Error Response Enable bit: When the secondary interface is in parity mode, the function of this bit is as defined in Section 5.1.2.13. In ECC mode, this bit controls the device's response to uncorrectable ECC errors. If the bit is set, the device takes the action described in this specification for an uncorrectable ECC error. If the bit is cleared, the device records the error in the PCI-X Bridge ECC Control and Status, ECC First Address, ECC Second Address, and PCI-X Bridge ECC Attribute registers, but in all other respects, treats the transaction as if it had no error. Correctable ECC errors are corrected independent of the state of this bit.
  - Fast Back-to-Back Enable bit: Ignored by the bridge if the secondary interface is in PCI-X mode.
  - Secondary Discard Timer bit: Ignored by the bridge if the secondary interface is in PCI-X mode.
  - Discard Timer Status bit: This bit is never set if the secondary interface is in PCI-X mode.
  - Discard Timer SERR# Enable bit: Ignored by the bridge if the secondary interface is in PCI-X mode.
- Secondary Status register. If the secondary interface is in PCI-X mode, the Secondary Status register is restricted as described below:
  - Fast Back-to-Back Capable bit: This bit must be set to 0 in PCI-X mode.
  - Detected Parity Error bit and Master Data Parity Error bit: These bits are set as described in PCI-X PT 2.0a.
  - DEVSEL timing field: Indicates conventional DEVSEL# timing regardless of the operating mode.

5.2. Capabilities List Items and Extended Capabilities

Certain capabilities added to PCI after the publication of PCI 2.1 are supported by adding a set of registers to a linked list called the Capabilities List (see PCI 3.0). Some of these capabilities are required of all PCI Express and/or conventional PCI/PCI-X devices. Others are required only when specific functions, such as PCI-X, are supported by the bridge. Capabilities List items identified as optional are not required for compliance.

PCI Express Extended Capability registers are located in device configuration space at offset 256 or greater, as described in Chapter 7 of PCI Express Base 1.0a. These registers are accessible using only the PCI Express extended configuration space access mechanism. The Advanced Error Reporting capability described in Section 5.2.3 is an optional Extended Capability for bridges.

For information on optional capabilities that apply to both bridges and Endpoints (simple devices), refer to PCI Express Base 1.0a, PCI 3.0, and PCI-X PT 2.0a.

5.2.1. PCI Express Capability Structure

The PCI Express Capability Structure is a Capabilities List item in PCI 3.0-compatible configuration space. PCI Express bridges must implement the PCI Express Capability Structure as defined in Chapter 7 of PCI Express Base 1.0a. There are very few differences between the interpretation of the PCI Express Capability Structure registers and fields for Endpoint devices and PCI Express bridges. The bridge-specific requirements are as follows:

- Implement registers common to all PCI Express devices (byte offset 0h to 13h).
- The Device/Port Type field of the PCI Express Capabilities register must be set to 0111b (PCI Express to PCI/PCI-X Bridge).
- The Enable Relaxed Ordering and the Enable No Snoop fields of the Device Control register must be hardwired to 0b if the bridge does not implement any internal sources that utilize the Relaxed Ordering and No Snoop functionality. These bits do not affect bridge requirements to forward the Relaxed Ordering and No Snoop header fields in both directions unmodified.
- The Correctable Error Detected, Non-Fatal Error Detected, and Fatal Error Detected Status bits in the Device Status register are set for the corresponding errors on both the PCI Express and conventional PCI/PCI-X interfaces.
- The Correctable Error Reporting Enable, Non-Fatal Error Reporting Enable, and Fatal Error Reporting Enable bits in the Device Control register control error message generation on behalf of errors on both the PCI Express and conventional PCI/PCI-X interfaces.
- The Unsupported Request Reporting Enable bit in the Device Control register only controls error reporting on behalf of Unsupported Request errors detected on the PCI Express interface.
- PCI Express bridges must honor the Transaction Pending bit in the Device Status register. PCI Express bridges must set this bit whenever the bridge is waiting for a completion from PCI Express on behalf of a non-posted transaction originated by an internal source. This requires

that bridges keep track of all outstanding non-posted transactions initiated on PCI Express on behalf of internal sources. This bit is not affected by transactions forwarded through the bridge.

- Bridges are required to implement the Bridge Configuration Retry Enable bit (bit 15) in the Device Control register (see Section 4.3 for details). When this read-write bit is set to 1b, bridges return a completion with Completion Retry Status on PCI Express if a configuration transaction forwarded to the secondary interface did not complete within the implementation-specific timeout period. When this bit is set to 0b, bridges do not generate completions with Completion Retry Status on behalf of configuration transactions. The default value of this bit is 0b.

See Section 2.2 for bridge-specific recommendations on the implementation and use of the Link Control register's RCB field.

All registers and fields not described in this section are defined as described in Chapter 7 of PCI Express Base 1.0a.

5.2.2. PCI-X Capability

PCI-X Mode 1 and Mode 2 support, as defined in PCI-X PT 2.0a, are optional for PCI Express bridges. The Capability associated with these modes is described below as it applies to these bridges.

5.2.2.1. PCI-X Bridge Capabilities List Item

PCI Express bridges that support PCI-X must include a PCI-X Capabilities List item as shown in Figure 5-3. Refer to PCI-X PT 2.0a for more information on PCI-X Capabilities List item requirements for non-bridge functions of multi-function devices.

Bit location 0 is the least significant bit in each of the registers.

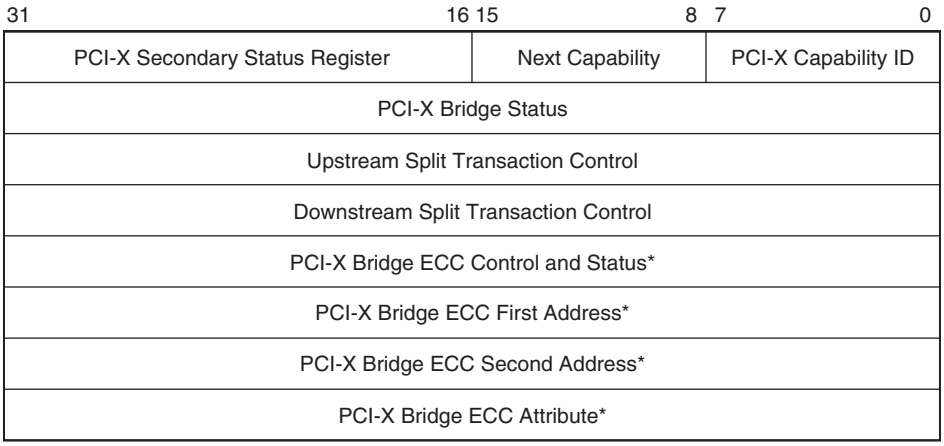

Figure 5-3: PCI-X Capabilities List Item for a Type 01h Configuration Header

The next few sections describe the PCI-X capability registers in detail. Refer to PCI-X PT 2.0a for all PCI-X bus protocol specifics associated with each of the individual register bits in the PCI-X capability registers.

5.2.2.1.1. PCI-X Capability ID

This register identifies this item in the Capabilities List as a PCI-X register set. It is read-only and returns 07h when read.

5.2.2.1.2. Next Capabilities Pointer

This register points to the next item in the Capabilities List as required by PCI 3.0.

5.2.2.1.3. PCI-X Secondary Status Register

The PCI-X Secondary Status register, shown in Table 5-8, reports status information about the secondary interface.

Table 5-8: PCI-X Secondary Status Register

Bit Location	Register Description	Attributes
0	**64-Bit Device** – This bit indicates the width of the bridge's secondary AD interface. 0 - The bus is 32 bits wide. 1 - The bus is 64 bits wide.	RO
1	**133 MHz Capable** – This bit indicates that the bridge's secondary interface is capable of 133 MHz operation in PCI-X mode. If the interface is capable of PCI-X 266 or PCI-X 533 operation, it is also capable of PCI-X 133 operation. See PCI-X PT 2.0a for the requirements. 0 - The maximum operating clock frequency is 66 MHz. 1 - The maximum operating clock frequency is 133 MHz.	RO
2	**Split Completion Discarded** – This bit is set if the bridge discards a Split Completion propagating downstream toward the secondary bus because the requester would not accept it. 0 - No Split Completion has been discarded. 1 - A Split Completion has been discarded. Default value of this field is 0.	RW1C
3	**Unexpected Split Completion** – This bit is set if an unexpected Split Completion is targeted at the bridge from PCI-X. 0 - No unexpected Split Completion has been received. 1 - An unexpected Split Completion has been received. Default value of this field is 0.	RW1C

Bit Location	Register Description	Attributes
4	**Split Completion Overrun** – This bit is set if the bridge terminates a Split Completion on the secondary bus with Retry or Disconnect at Next ADB because the bridge buffers are full. It is used by algorithms that optimize the setting of the downstream Split Transaction Commitment Limit register. See Appendix D in PCI-X PT 2.0a for more details. The bridge is also permitted to set this bit in other situations that indicate that the bridge commitment limit is too high. For example, if the bridge stores immediate completion data in the same buffer area as Split Completion data, the completer executes the transaction as an Immediate Transaction, and the bridge disconnects the transaction because the buffers became full. 0 - The bridge has accepted all Split Completions. 1 - The bridge has terminated a Split Completion with Retry or Disconnect at Next ADB because the bridge buffers were full. Default value of this field is 0.	RW1C
5	**Split Request Delayed** – This bit is set any time the bridge has a request to forward a transaction on the secondary bus but cannot because there is not enough room within the limit specified in the Split Transaction Commitment Limit field in the Downstream Split Transaction Control register. It is used by algorithms that optimize the setting of the downstream Split Transaction Commitment Limit register. See Appendix D of PCI-X PT 2.0a for more details. 0 - The bridge has not delayed a Split Request. 1 - The bridge has delayed a Split Request. Default value of this field is 0.	RW1C

<table>
<tr><th>Bit Location</th><th>Register Description</th><th>Attributes</th></tr>
<tr><td>9:6</td><td>Secondary Bus Mode and Frequency – This register enables configuration software to determine to what mode and (in PCI-X mode) what frequency the bridge set the secondary bus the last time secondary RST# was asserted. This is the same information the bridge used to create the PCI-X initialization pattern on the secondary bus the last time secondary RST# was asserted.

For PCI-X Mode 1, this register also indicates whether the secondary bus was initialized in parity mode or ECC mode the last time secondary RST# was asserted. If the secondary bus error protection method is subsequently changed by writing to the ECC Mode in the secondary PCI-X Bridge ECC Control and Status register, that change has no effect on this register.

<table>
<tr><th>Reg</th><th>Mode</th><th>Error Protection</th><th>Max. Clock Freq. (MHz) (ref)</th><th>Min. Clock Period (ns) (Note)</th></tr>
<tr><td>0h</td><td>conventional</td><td>parity</td><td>N/A</td><td>N/A</td></tr>
<tr><td>1h</td><td>PCI-X Mode 1</td><td>parity</td><td>66</td><td>15</td></tr>
<tr><td>2h</td><td>PCI-X Mode 1</td><td>parity</td><td>100</td><td>10</td></tr>
<tr><td>3h</td><td>PCI-X Mode 1</td><td>parity</td><td>133</td><td>7.5</td></tr>
<tr><td>4h</td><td>PCI-X Mode 1</td><td>ECC</td><td>reserved</td><td>reserved</td></tr>
<tr><td>5h</td><td>PCI-X Mode 1</td><td>ECC</td><td>66</td><td>15</td></tr>
<tr><td>6h</td><td>PCI-X Mode 1</td><td>ECC</td><td>100</td><td>10</td></tr>
<tr><td>7h</td><td>PCI-X Mode 1</td><td>ECC</td><td>133</td><td>7.5</td></tr>
<tr><td>8h</td><td>PCI-X 266 (Mode 2)</td><td>ECC</td><td>reserved</td><td>reserved</td></tr>
<tr><td>9h</td><td>PCI-X 266 (Mode 2)</td><td>ECC</td><td>66</td><td>15</td></tr>
<tr><td>Ah</td><td>PCI-X 266 (Mode 2)</td><td>ECC</td><td>100</td><td>10</td></tr>
<tr><td>Bh</td><td>PCI-X 266 (Mode 2)</td><td>ECC</td><td>133</td><td>7.5</td></tr>
<tr><td>Ch</td><td>PCI-X 533 (Mode 2)</td><td>ECC</td><td>reserved</td><td>reserved</td></tr>
<tr><td>Dh</td><td>PCI-X 533 (Mode 2)</td><td>ECC</td><td>66</td><td>15</td></tr>
<tr><td>Eh</td><td>PCI-X 533 (Mode 2)</td><td>ECC</td><td>100</td><td>10</td></tr>
<tr><td>Fh</td><td>PCI-X 533 (Mode 2)</td><td>ECC</td><td>133</td><td>7.5</td></tr>
</table>
Note: For bridges that support PCI-X Mode 1 using clock jitter class 2 and for bridges that support PCI-X Mode 2, this is minimum average clock period. See Chapter 2 of PCI-X EM 2.0a for further information.</td><td>RO</td></tr>
<tr><td>11:10</td><td>Reserved</td><td>RsvdZ</td></tr>
</table>

Bit Location	Register Description	Attributes
13:12	**PCI-X Capabilities List Item Version** – These bits indicate the format of the PCI-X Capabilities List item, and whether the bridge supports ECC in Mode 1. ECC control and status bits appear in versions 1 and 2. Bridges that do not support ECC use version 0. Bridges that support ECC in Mode 2 but not in Mode 1 use version 1. Bridges that support ECC in Mode 1 but do not support PCI-X Mode 2 and bridges that support ECC both in Mode 2 and Mode 1 use version 2. **Register / Version / ECC Support / Capabilities List Item Size** 00b / 0 / none / 16 bytes 01b / 1 / Mode 2, not Mode 1 / 32 bytes 10b / 2 / Mode 1 or Modes 1 and 2 / 32 bytes 11b / reserved / reserved / reserved	RO
14	**PCI-X 266 Capable** – This bit indicates that the bridge's secondary interface is capable of PCI-X 266 operation. 0 - The secondary interface is not capable of PCI-X 266 operation. 1 - The secondary interface is capable of PCI-X 266 operation.	RO
15	**PCI-X 533 Capable** – This bit indicates that the bridge's secondary interface is capable of PCI-X 533 operation. 0 - The secondary interface is not capable of PCI-X 533 operation. 1 - The secondary interface is capable of PCI-X 533 operation.	RO

5.2.2.1.4. PCI-X Bridge Status Register

The PCI-X Bridge Status register, shown in Table 5-9, supports PCI-X register fields that are specific to bridge architectures.

Table 5-9: PCI-X Bridge Status Register

Bit Location	Register Description	Attributes
2:0	**Function Number** – This register is read for diagnostic purposes only. It indicates the number of this function; i.e., the number in the Function Number field of the address of a Type 0 configuration transaction to which this bridge responds. The bridge uses the Bus Number, Device Number, and Function Number fields to create the Completer ID when responding with a Completion to a read/write of an internal bridge register. These fields are also used for forming the Requester ID when the bridge takes ownership of a transaction forwarded from PCI Express to PCI-X. See Chapter 2 for more details.	RO
7:3	**Device Number** – This register is read for diagnostic purposes only. It indicates the number of this device; i.e., the number in the Device Number field in the header of a PCI Express Type 0 Configuration Write Request that is assigned to this bridge by the upstream PCI Express device. The bridge uses this number as described for the Function Number field above. Each time the bridge is addressed by a PCI Express Type 0 Configuration Write Request, the bridge must update this register with the contents of the Device Number field of the Configuration Write Request, regardless of which register in the bridge is addressed by the transaction. Default value of this field is 0h.	RO
15:8	**Bus Number** – This register is read for diagnostic purposes only. It is an additional address from which the contents of the Primary Bus Number register in the Type 01h Configuration Space header is read. The bridge uses this number as described for the Function Number field above.	RO
16	**64-bit Device** – This bit does not apply to PCI Express and must be hardwired to 0.	RO
17	**133 MHz Capable** – This bit does not apply to PCI Express and must be hardwired to 0.	RO
18	**Split Completion Discarded** – This bit does not apply to PCI Express and must be hardwired to 0.	RO
19	**Unexpected Split Completion** – This bit is set if an unexpected Completion is targeted at the bridge from PCI Express. 0 - No unexpected Split Completion was received. 1 - An unexpected Split Completion was received. Default value of this bit is 0h.	RW1C

Bit Location	Register Description	Attributes
20	**Split Completion Overrun** – This bit is set by a bridge when it is unable to return Completion (header and/or data) Flow Control credits to the transmitter due to unavailability of buffer space for additional completions that are expected by the bridge. Bridges do not set this bit if no additional completions are expected (e.g., the bridge is operating with its upstream Split Transaction Commitment Limit register set equal to its upstream Split Transaction Capacity). This bit is used by algorithms that optimize the setting of the upstream Split Transaction Commitment Limit register. See Appendix D of PCI-X PT 2.0a for more details. The bridge is also permitted to set this bit in other situations that indicate that the bridge commitment limit is too high. The means by which it does this is outside the scope of this specification. 0 - The bridge has not encountered an overrun condition. 1 - The bridge has encountered an overrun condition. Default value of this bit is 0h.	RW1C
21	**Split Request Delayed** – This bit is set any time the bridge has a request to forward a transaction on the primary interface but cannot because there is not enough room within the limit specified in the Split Transaction Commitment Limit field in the Upstream Split Transaction Control register. It is used by algorithms that optimize the setting of the upstream Split Transaction Commitment Limit register. See Appendix D of PCI-X PT 2.0a for more details. 0 - The bridge has not delayed a Split Request. 1 - The bridge has delayed a Split Request. Default value of this bit is 0h.	RW1C
28:22	**Reserved**	RsvdZ
29	**Device ID Messaging Capable** – This bit indicates that the bridge is capable of translating between PCI-X Device ID Messages and PCI Express Vendor-Defined Messages when the secondary interface is operating in a PCI-X mode. Support of message translation is optional for both PCI-X Mode 1 and Mode 2. 0 - The bridge is not capable of translating DIMs. 1 - The bridge is capable of translating DIMs.	RO
30	**PCI-X 266 Capable** – Does not apply to PCI Express bridges. Must be hardwired to 0.	RO
31	**PCI-X 533 Capable** – Does not apply to PCI Express bridges. Must be hardwired to 0.	RO

IMPLEMENTATION NOTE

The Primary Bus Number and PCI-X Bus Number Registers

A PCI Express bridge's primary Bus Number is initialized in one location but can be read from two. The value is initialized in the Primary Bus Number in the standard Type 01h Configuration Space header and can be read both there and in the PCI-X Capabilities List item in the Bus Number register. The second "read-only" location is provided to keep the programming model of the PCI-X Bridge Status register consistent with the PCI-X Status register for other PCI-X devices.

5.2.2.1.5. Upstream Split Transaction Register

The Upstream Split Transaction register, shown in Table 5-10, controls behavior of the bridge buffers for forwarding Split Transactions from a secondary interface requester to a primary interface completer when the secondary interface is operating in PCI-X mode.

Table 5-10: Upstream Split Transaction Register

Bit Location	Register Description	Attributes
15:0	**Split Transaction Capacity** – Some bridges store Split Completions for memory reads in a separate buffer from Split Completions for I/O and configuration reads and writes. For such bridges, this register indicates the size of the buffer (in number of ADQs) for storing Split Completions for memory reads for requesters on the secondary interface addressing completers on the primary interface. If the bridge stores Split Read Completions in the same buffer as other Split Completions, this register indicates the size of this buffer in units of ADQs.	RO

Bit Location	Register Description	Attributes
31:16	**Split Transaction Commitment Limit** – Some bridges store Split Completions for memory reads in a separate buffer from Split Completions for I/O and configuration reads and writes. For such a bridge, this register indicates the cumulative Sequence size for all memory read transactions forwarded by the bridge from requesters on the secondary interface addressing completers on the primary interface. (See PCI-X PT 2.0a for a detailed discussion of Split Transaction commitment.) If the bridge stores Split Read Completions in the same buffer as other Split Completions, this register indicates the size of all upstream Split Transactions of these types that the bridge is permitted to commit to at one time. This register indicates the size of the commitment limit in units of ADQs. Software is permitted to program this register to any value greater than or equal to the contents of the Split Transaction Capacity register. A value less than the contents of the Split Transaction Capacity register causes unspecified results. If this register is set to FFFFh, the bridge is permitted to forward all Split Requests of any size regardless of the amount of buffer space available (an exception is described in Section 2.6). Software is permitted to change this register at any time. The most recent value of the register is used each time the bridge forwards a Split Transaction. If the register value is set to FFFFh, the bridge does not track the outstanding commitment. If the register is later set to something else, the bridge does not accurately track outstanding commitments until all outstanding commitments complete. Systems that require accurate limitation of Split Transactions must never set this register to FFFFh, or they must quiesce all devices that initiate traffic that crosses the bridge in this direction after the register setting is changed from FFFFh. An algorithm for setting this register is not specified. System software is permitted to use any method for selecting the value for this register. Individual devices and device drivers are not permitted to change the value of this register except under control of a system-level configuration routine. See PCI-X PT 2.0a for details and setting recommendations. Default value of this field equals the value stored in the Split Transaction Capacity register.	RW

5.2.2.1.6. Downstream Split Transaction Register

The Downstream Split Transaction register, shown in Table 5-11, controls behavior of the bridge buffers for forwarding Split Transactions from a primary interface requester to a secondary interface completer when the secondary interface is operating in PCI-X mode.

Table 5-11: Downstream Split Transaction Register

Bit Location	Register Description	Attributes
15:0	**Split Transaction Capacity** – Some bridges store Split Completions for memory reads in a separate buffer from Split Completions for I/O and configuration reads and writes. For such bridges, this register indicates the size of the buffer (in number of ADQs) for storing Split Completions for memory reads for requesters on the primary interface addressing completers on the secondary interface. If the bridge stores Split Read Completions in the same buffer as other Split Completions, this register indicates the size of this buffer in units of ADQs.	RO
31:16	**Split Transaction Commitment Limit** – Some bridges store Split Completions for memory reads in a separate buffer from Split Completions for I/O and configuration reads and writes. For such a bridge, this register indicates the cumulative Sequence size for all memory read transactions forwarded by the bridge from requesters on the primary interface addressing completers on the secondary interface. (See PCI-X PT 2.0a for a detailed discussion of Split Transaction commitment.) If the bridge stores Split Read Completions in the same buffer as other Split Completions, this register indicates the size of all downstream Split Transactions of these types that the bridge is permitted to commit to at one time. This register indicates the size of the commitment limit in units of ADQs. Software is permitted to program this register to any value greater than or equal to the contents of the Split Transaction Capacity register. A value less than the contents of the Split Transaction Capacity register causes unspecified results. If this register is set to FFFFh, the bridge is permitted to forward all Split Request of any size regardless of the amount of buffer space available. Software is permitted to change this register at any time. The most recent value of the register is used each time the bridge forwards a Split Transaction. If the register value is set to FFFFh, the bridge does not track the outstanding commitment. If the register is later set to something else, the bridge does not accurately track outstanding commitments until all outstanding commitments complete. Systems that require accurate limitation of Split Transactions must never set this register to FFFFh, or they must quiesce all devices that initiate traffic that crosses the bridge in this direction after the register setting is changed from FFFFh. An algorithm for setting this register is not specified. System software is permitted to use any method for selecting the value for this register. Individual devices and device drivers are not permitted to change the value of this register except under control of a system-level configuration routine. See PCI-X PT 2.0a for details and setting recommendations. Default value of this field equals the value stored in the Split Transaction Capacity register.	RW

5.2.2.1.7. PCI-X Bridge ECC Control and Status Register

The PCI-X Bridge ECC Control and Status register, shown in Table 5-12, provides information about ECC errors detected by the bridge when the secondary interface is in PCI-X mode and supports ECC. The register is relevant only to the secondary interface for PCI Express bridges. This register is defined only in versions 1 and 2 of the PCI-X Capabilities List item.

Registers that store information from the failing transaction always store information directly from the bus (uncorrected), even if correction of the error is possible.

Table 5-12: PCI-X Bridge ECC Control and Status Register

Bit Location	Register Description	Attributes
0	**Select Secondary ECC Registers** – This bit is read-only with a value of 1. This bit being hardwired to 1 indicates that the ECC error logging registers (PCI-X Bridge ECC Control and Status, PCI-X Bridge ECC First Address, PCI-X Bridge ECC Second Address, and PCI-X Bridge ECC Attribute registers) are always associated with the bridge's secondary interface. The value of this bit in the data pattern being written to the PCI-X Bridge ECC Control and Status register is ignored. Writes to the PCI-X Bridge ECC Control and Status register are always associated with the secondary interface.	RO
1	**Error Present in Other ECC Register Bank** – This bit does not apply to PCI Express bridges and must be hardwired to 0.	RO
2	**Additional Correctable ECC Error** – This bit is set if the bridge detects a correctable ECC error, as described in PCI-X PT 2.0a, Chapter 5, while the device is already indicating some other ECC error on the same interface (i.e., the ECC Error Phase register is non-zero). 0 - No additional correctable ECC error was detected. 1 - One or more additional correctable ECC errors were detected. Default value of this bit is 0h.	RW1C
3	**Additional Uncorrectable ECC Error** – This bit is set if the bridge detects an uncorrectable ECC error, as described in PCI-X PT 2.0a, Chapter 5, while the device is already indicating some other ECC error on the same interface (i.e., the ECC Error Phase register is non-zero). 0 - No additional uncorrectable ECC error was detected. 1 - One or more additional uncorrectable ECC errors were detected. Default value of this bit is 0h.	RW1C

Bit Location	Register Description	Attributes
6:4	**ECC Error Phase** – If the bridge detects either a correctable or uncorrectable ECC error, as described in PCI-X PT 2.0a, Chapter 5, this register indicates in which phase of the transaction the error occurred, and for data phase errors whether it was a 32-bit data error (7-bit ECC) or a 64-bit data error (8-bit ECC). If this register is set to 0, the bridge is enabled to latch information about an ECC error. If the device detects an error, it latches the phase of the error in this register and stores status information for the error in the ECC Status, ECC Address, and ECC Attribute registers. Writing a 1 to any of these bits clears this register and enables the device to capture the next error. **Register** – **ECC Error Phase** 0 – No error 1 – First 32 bits of address 2 – Second 32 bits of address 3 – Attribute phase 4 – 32- or 16-bit data phase 5 – 64-bit data phase 6 – reserved 7 – reserved Default value of this bit is 0h.	RW1C
7	**ECC Error Corrected** – If the ECC Error Phase register is non-zero, this bit indicates whether the error that was captured was corrected. Correctable ECC errors that occur on either primary or secondary interfaces while the Disable Single-Bit-Error Correction bit for that interface is 0 are the only errors that are corrected. If the ECC Error Phase register is zero, this bit is undefined. 0 - The error that was captured was not corrected. 1 - The error that was captured was corrected.	RO
15:8	**Syndrome** – The syndrome indicates information about the bit or bits that are in error, as described in PCI-X PT 2.0a, Chapter 5. **Bit** – **Syndrome** 8 – E0 9 – E1 10 – E2 11 – E3 12 – E4 13 – E5 14 – E6 15 – E7 for 64-bit data, 0b for 32-bit data, or E16/Chk for 16-bit data	RO

Bit Location	Register Description	Attributes
19:16	**Error First (or only) Command** – If the ECC Error Phase register is non-zero, this register indicates the contents of the C/BE[3::0]# bus for the first (or only) address phase of the transaction that included the error.	RO
23:20	**Error Second Command** – If the ECC Error Phase register is non-zero and the transaction that included the error used a dual address cycle, this register indicates the contents of the C/BE[3::0]# bus for the second address phase of the transaction that included the error.	RO
27:24	**Error Upper Attributes –** If the ECC Error Phase register is non-zero, this register indicates the contents of the C/BE[3::0]# bus for the attribute phase of the transaction that included the error.	RO
28	**ECC Control Update Enable** – This bit always reads as a 0. If this bit is 1 in the data pattern being written, the Disable Single-Bit-Error Correction and ECC Mode bits are also updated (written). If this bit is 0 in the data pattern being written, the Disable Single-Bit-Error Correction and ECC Mode bits are not updated.	WT
29	**Reserved**	RsvdP
30	**Disable Single-Bit-Error Correction** – If the conventional PCI/PCI-X interface of the bridge is in ECC mode and this bit is 0, correctable errors, as described in PCI-X PT 2.0a, Chapter 5, that are received on that interface are corrected. If the conventional PCI/PCI-X interface of the bridge is in ECC mode and this bit is 1, correctable errors that occur on that interface are not corrected and are treated as uncorrectable errors, including the setting of status bits and assertion of error indicator signals on the bus as described in PCI-X PT 2.0a, Chapter 5. Disabling single-bit error correction enhances the error detection capability of the ECC as described in PCI-X PT 2.0a, Chapter 5. If the conventional PCI/PCI-X interface of the bridge is in parity mode (ECC Mode bit is 0), this bit has no meaning and is ignored by the bridge. Writes to this register do not affect this bit unless the ECC Control Update Enable bit is a 1 in the data pattern being written. Default value of this bit is 0h.	RW

Bit Location	Register Description	Attributes
31	**ECC Mode** – This bit is read-only in conventional PCI mode, read/write in Mode 1 if ECC is supported in Mode 1, read-only in Mode 1 if ECC is not supported in Mode 1, and read-only in Mode 2. If this bit is 1, the secondary interface of the bridge is in ECC mode. If this bit is 0, the secondary interface of the bridge is in parity mode. Writes to this register do not affect this bit unless the ECC Control Update Enable bit is a 1 in the data pattern being written. In PCI-X Mode 1 for PCI Express bridges that support ECC in Mode 1, the state of this bit after the secondary interface is reset is determined by the mode of that interface and should be consistent with the error protection (ECC or parity as specified in PCI-X PT 2.0a, Chapter 6) conveyed when the bridge drives the PCI-X initialization pattern. If the secondary interface is in conventional PCI mode or in PCI-X Mode 1 for devices that do not support ECC in Mode 1, this bit is always a 0. If the secondary interface is in PCI-X Mode 2, this bit is always a 1.	RW/RO

5.2.2.1.8. PCI-X Bridge ECC Address Registers

There are two ECC address registers:

- ECC First 32 Bits of Address
- ECC Second 32 Bits of Address

For a PCI Express bridge, these registers always display secondary interface information (Select Secondary ECC Registers is hardwired to a 1). These registers are defined only in versions 1 and 2 of the PCI-X Capabilities List item.

If the ECC Error Phase register for the secondary interface is non-zero (indicating that an error has been captured), these registers indicate the contents of the AD[31:0] bus (for 64- and 32-bit buses) or the two phases of AD[31:16] bus (for 16-bit secondary buses) for the address phase or phases of the transaction that included the error. If the ECC Error Phase registers are zero, the contents of these registers are undefined.

The ECC First 32 Bits of Address register records the least significant 32 bits of the address, regardless of the type, length, or width of the transaction, or the phase in which the error occurred. If the transaction used a dual address cycle, the ECC Second 32 Bits of Address register records the most significant 32 bits of the address. If the transaction used a single address cycle, the contents of the ECC Second 32 Bits of Address register are 0. The bridge examines the transaction command in the first address phase, after correcting any correctable ECC errors, to determine whether the transaction used a dual or single address cycle. Registers that store information from the failing transaction always store information directly from the bus (uncorrected), even if correction of the error is possible.

These registers are read-only.

5.2.2.1.9. PCI-X Bridge ECC Attribute Register

For a PCI Express bridge, this register always displays secondary interface information (Select Secondary ECC Registers is hardwired to a 1). This register is defined only in versions 1 and 2 of the PCI-X Capabilities List item.

If the ECC Error Phase register for the secondary interface is non-zero (indicating that an error was captured), the ECC Attribute register indicates the contents of the AD[31:0] bus (for 64- and 32-bit buses) or the two phases of AD[31:16] bus (for 16-bit secondary buses) for the attribute phase of the transaction that included the error. If the ECC Error Phase register is zero, the contents of the ECC Attribute register are undefined.

This register records the contents of the bus during the attribute phase, regardless of the type or length of the transaction, or the phase in which the error occurred. Registers that store information from the failing transaction always store information directly from the bus (uncorrected), even if correction of the error is possible.

This register is read-only.

5.2.3. Advanced Error Reporting Capability

The PCI Express Advanced Error Reporting capability is an optional extended capability that may be implemented by PCI Express devices supporting advanced error control and reporting. This capability is defined for Root Complexes and other PCI Express devices in Chapter 7 of PCI Express Base 1.0a. The bridge-specific differences are presented below. Software must interpret the PCI Express Device/Port Type field (see Section 5.2.1 of this specification and Chapter 7 of PCI Express Base 1.0a) in the PCI Express Capability Structure to determine the version of the Advanced Error Reporting Capabilities List item.

PCI Express bridges that implement Advanced Error Reporting must use the version of the capabilities structure shown in Figure 5-4. Additional registers are defined that are used to control and report uncorrectable secondary bus errors. ECC-related functions and error reporting for the secondary bus are accessed through the PCI-X Mode 2 Capabilities List item's ECC registers as described in PCI-X PT 2.0a.

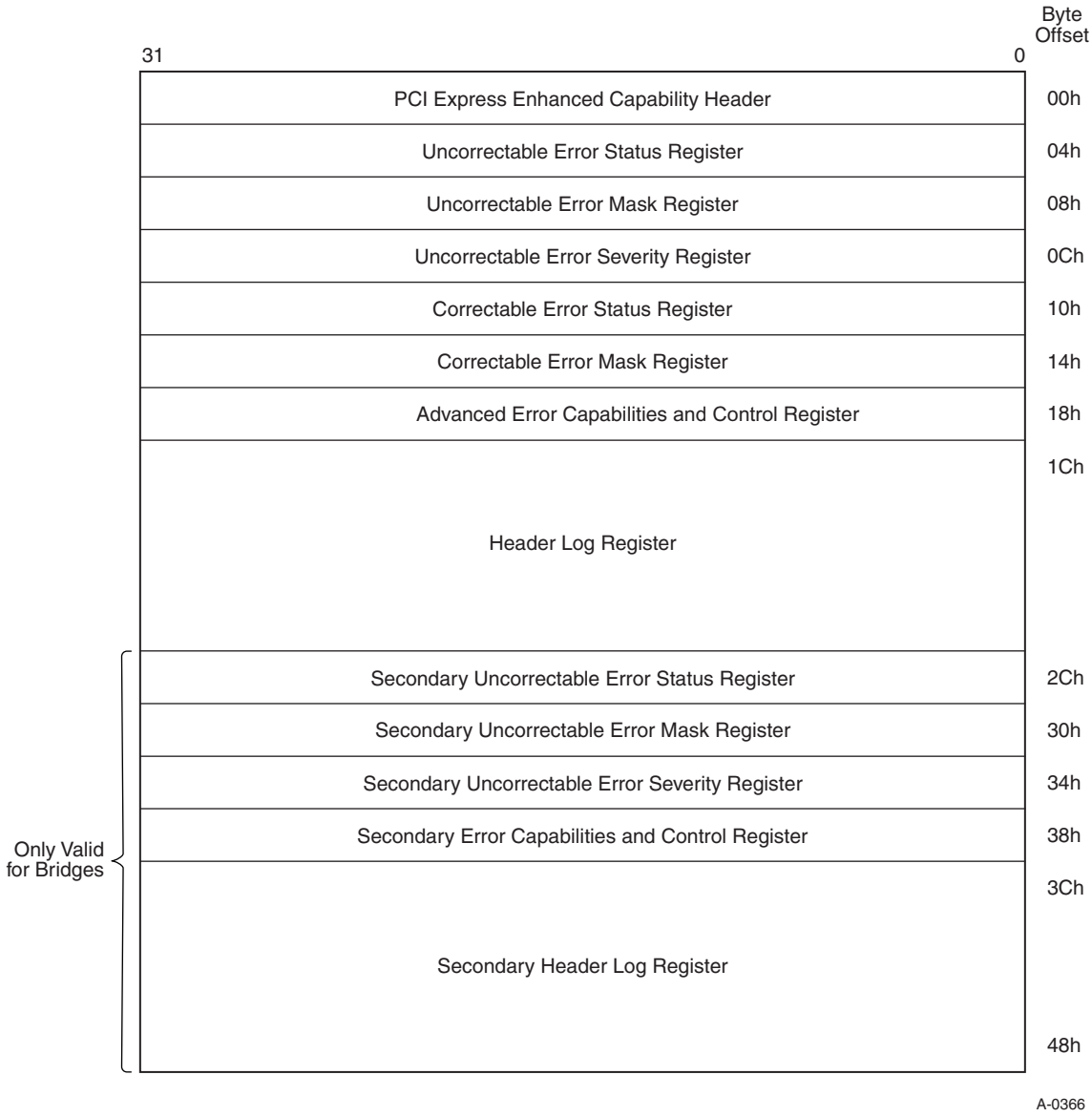

Figure 5-4: PCI Express Advanced Error Reporting Extended Capabilities List Item for Bridges

Note that if an error reporting bit field is marked as optional in the error registers, the bits must be implemented or not implemented as a group across the Status, Mask, and Severity registers. In other words, a device is required to implement the same error bit fields in corresponding Status, Mask, and Severity registers. Bits corresponding to bit fields that are not implemented must be hardwired to zero.

PCI Express Base 1.0a provides a detailed description of the Advanced Error Reporting Capabilities List item registers that are common across root devices, bridge devices, and simple Endpoint devices (i.e., registers with byte offsets 00h to 2Bh) and PCI Express bridges must follow the rules listed therein. Only the bridge-specific registers will be described in this document.

Chapter 10 details all the conventional PCI/PCI-X error conditions and the associated error logging and escalation to PCI Express, via the Advanced Error Reporting capability.

5.2.3.1. Secondary Uncorrectable Error Status Register (Offset 2Ch)

The Secondary Uncorrectable Error Status register, shown in Figure 5-5 and Table 5-13 reports error status of individual errors generated on the conventional PCI or PCI-X secondary bus interface. An individual error status bit that is set indicates that a particular error occurred; software may clear an error status by writing a 1 to the respective bit. See Chapter 10 for further information on these errors.

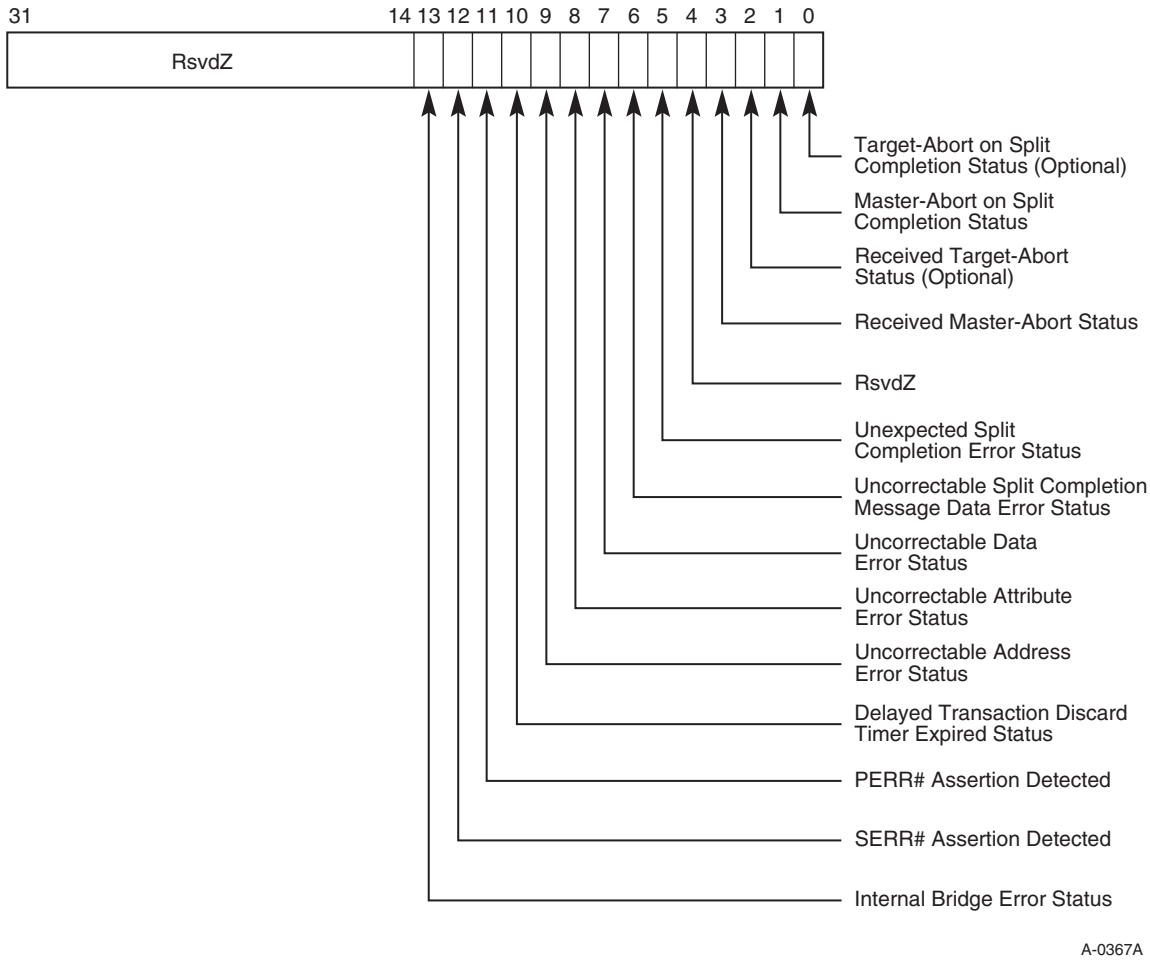

Figure 5-5: Secondary Uncorrectable Error Status Register

Table 5-13: Secondary Uncorrectable Error Status Register

Bit Location	Description	Register Attribute	Default Value
0	**Target-Abort on Split Completion Status** (Optional)	RW1CS	0
1	**Master-Abort on Split Completion Status**	RW1CS	0
2	**Received Target-Abort Status** (Optional)	RW1CS	0
3	**Received Master-Abort Status**	RW1CS	0
4	**Reserved**	RsvdZ	N/A
5	**Unexpected Split Completion Error Status**	RW1CS	0
6	**Uncorrectable Split Completion Message Data Error Status**	RW1CS	0
7	**Uncorrectable Data Error Status**	RW1CS	0
8	**Uncorrectable Attribute Error Status**	RW1CS	0
9	**Uncorrectable Address Error Status**	RW1CS	0
10	**Delayed Transaction Discard Timer Expired Status (No Header Log)**	RW1CS	0
11	**PERR# Assertion Detected**	RW1CS	0
12	**SERR# Assertion Detected (No Header Log)**	RW1CS	0
13	**Internal Bridge Error Status (No Header Log)**	RW1CS	0

5.2.3.2. Secondary Uncorrectable Error Mask Register (Offset 30h)

The Secondary Uncorrectable Error Mask register, shown in Figure 5-6 and Table 5-14, controls reporting of individual errors by the bridge to the PCI Express Root Complex via a PCI Express error message. A masked error (respective bit set in mask register) is not reported to the PCI Express Root Complex by the bridge. There is a mask bit per bit of the Secondary Uncorrectable Error Status register.

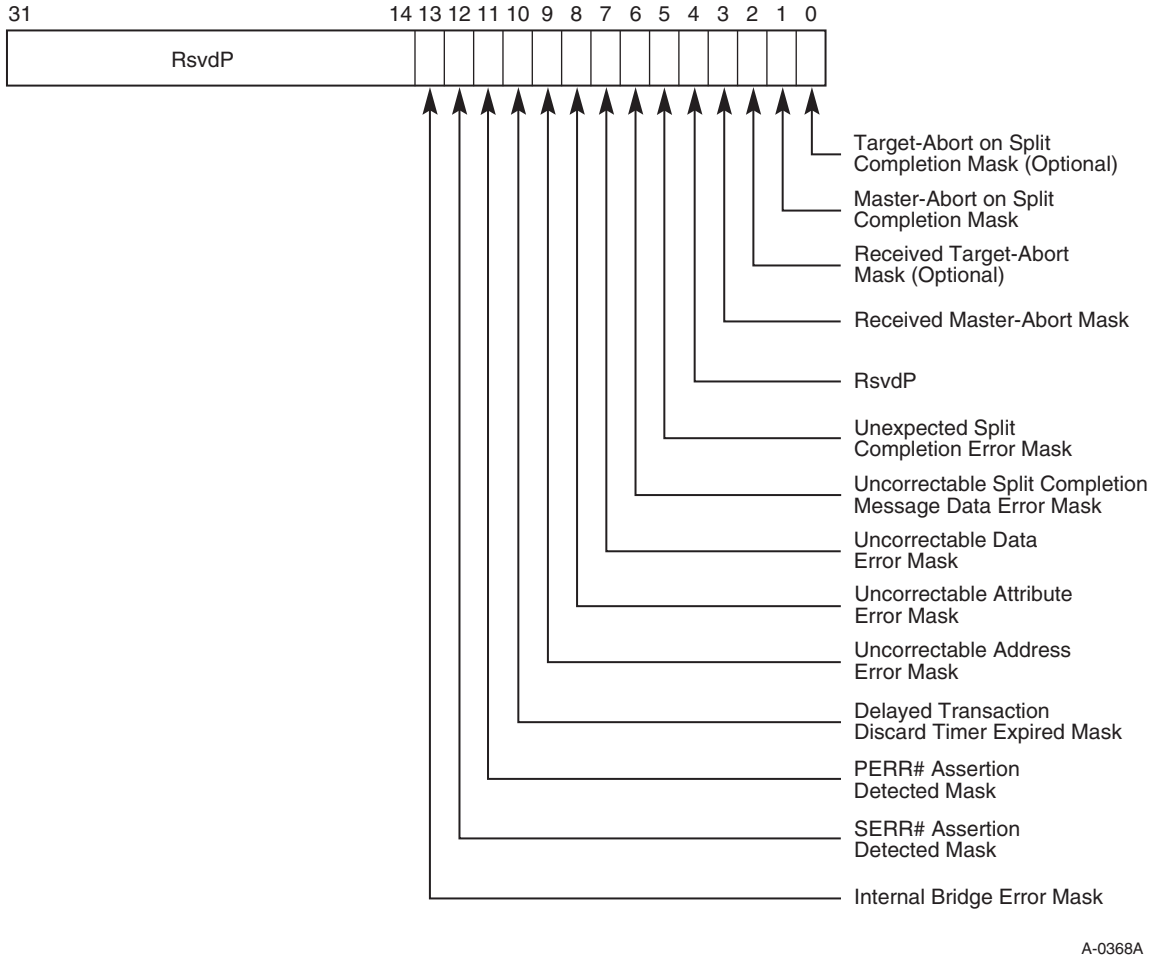

Figure 5-6: Secondary Uncorrectable Error Mask Register

Table 5-14: Secondary Uncorrectable Error Mask Register

Bit Location	Description	Register Attribute	Default Value
0	**Target-Abort on Split Completion Mask** (Optional)	RWS	0
1	**Master-Abort on Split Completion Mask**	RWS	0
2	**Received Target-Abort Mask** (Optional)	RWS	0
3	**Received Master-Abort Mask**	RWS	1
4	**Reserved**	RsvdP	N/A
5	**Unexpected Split Completion Error Mask**	RWS	1
6	**Uncorrectable Split Completion Message Data Error Mask**	RWS	0
7	**Uncorrectable Data Error Mask**	RWS	1
8	**Uncorrectable Attribute Error Mask**	RWS	1
9	**Uncorrectable Address Error Mask**	RWS	1
10	**Delayed Transaction Discard Timer Expired Mask**	RWS	1
11	**PERR# Assertion Detected Mask**	RWS	0
12	**SERR# Assertion Detected Mask**	RWS	1
13	**Internal Bridge Error Mask**	RWS	0

5.2.3.3. Secondary Uncorrectable Error Severity Register (Offset 34h)

The Secondary Uncorrectable Error Severity register, shown in Figure 5-7 and Table 5-15, controls whether an individual error is reported as a non-fatal or fatal error. An error is reported as fatal when the corresponding error bit in the severity register is set. If the bit is cleared, the corresponding error is considered non-fatal.

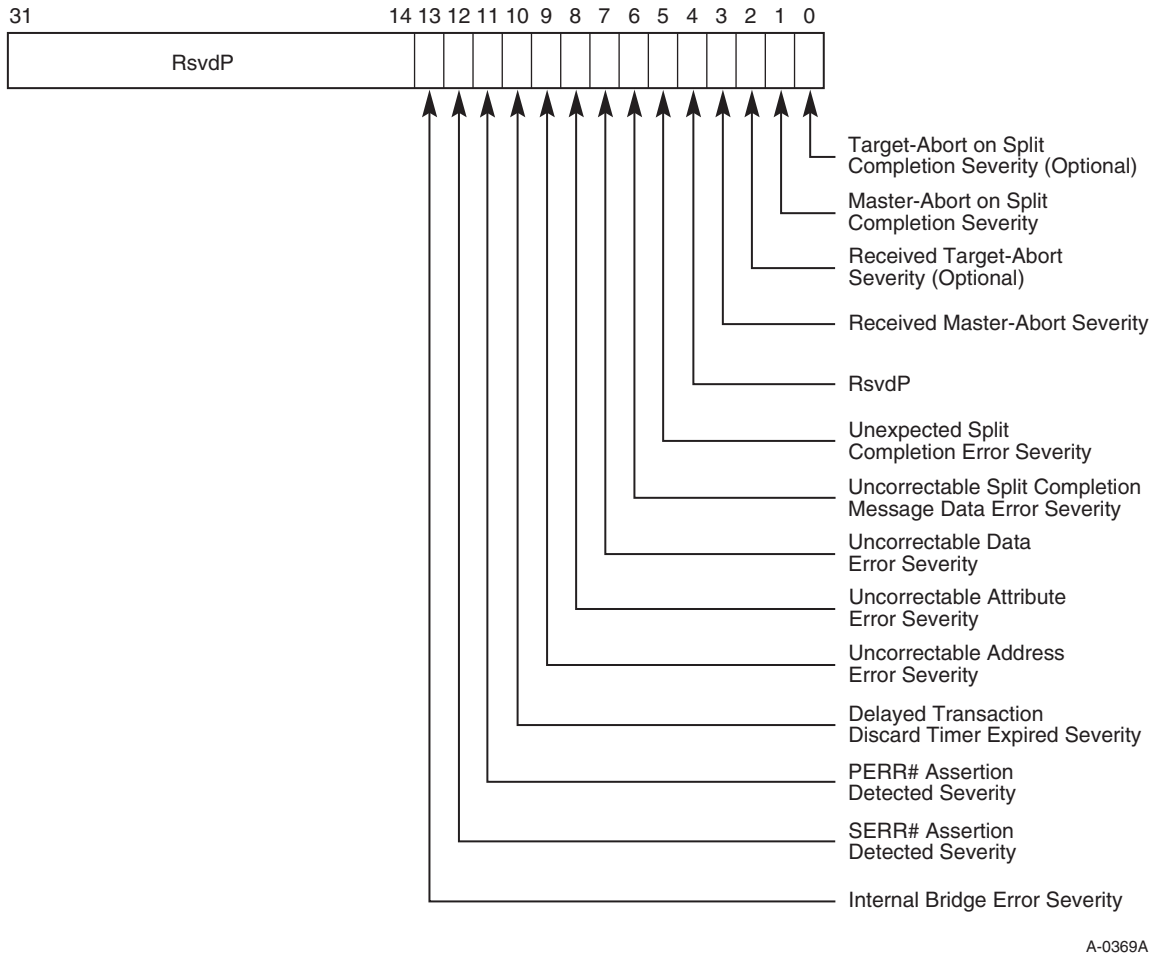

Figure 5-7: Secondary Uncorrectable Error Severity Register

Table 5-15: Secondary Uncorrectable Error Severity Register

Bit Location	Description	Register Attribute	Default Value
0	**Target-Abort on Split Completion Severity** (Optional)	RWS	0
1	**Master-Abort on Split Completion Severity**	RWS	0
2	**Received Target-Abort Severity** (Optional)	RWS	0
3	**Received Master-Abort Severity**	RWS	0
4	**Reserved**	RsvdP	N/A
5	**Unexpected Split Completion Error Severity**	RWS	0
6	**Uncorrectable Split Completion Message Data Error Severity**	RWS	1
7	**Uncorrectable Data Error Severity**	RWS	0
8	**Uncorrectable Attribute Error Severity**	RWS	1
9	**Uncorrectable Address Error Severity**	RWS	1
10	**Delayed Transaction Discard Timer Expired Severity**	RWS	0
11	**PERR# Assertion Detected Severity**	RWS	0
12	**SERR# Assertion Detected Severity**	RWS	1
13	**Internal Bridge Error Severity**	RWS	0

5.2.3.4. Secondary Error Capabilities and Control Register (Offset 38h)

The Secondary Error Capabilities and Control register, shown in Figure 5-8 and Table 5-16, contains the Secondary Uncorrectable First Error Pointer field. This field is a read-only register whose binary-encoded value indicates the bit position of the first error reported in the Secondary Uncorrectable Error Status register. The mechanism is rearmed when the bit position pointed to is cleared in the associated status register. The pointer value is not updated when the mechanism is rearmed.

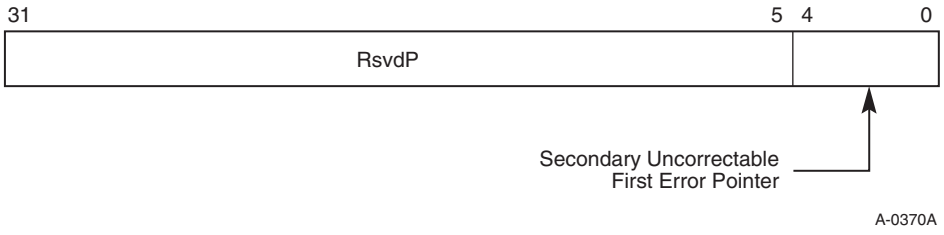

Figure 5-8: Secondary Error Capabilities and Control Register

Table 5-16: Secondary Error Capabilities and Control Register

Bit Location	Description	Register Attribute	Default Value
4:0	Secondary Uncorrectable First Error Pointer	ROS	0

5.2.3.5. Secondary Header Log Register (Offset 3Ch)

The Header Log register, shown in Figure 5-9 and Table 5-17, captures the header for the transaction on the secondary interface that generated an error. This register is locked from further header logs when the first unmasked uncorrectable error occurs and is rearmed for further header logs when the status register corresponding to the first uncorrectable error is cleared by software. The header log is undefined between logs. Refer to Chapter 10 for the details on the conditions under which the transaction header is logged.

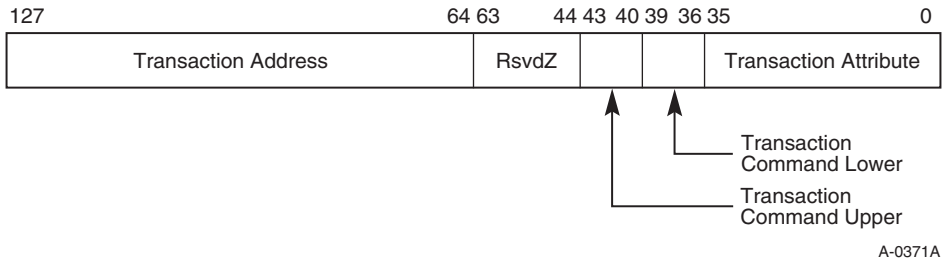

Figure 5-9: Secondary Header Log Register

Table 5-17: Secondary Header Log Register

Bit Location	Description	Register Attribute	Default Value
35:0	**Transaction Attribute** – The 36-bit value transferred on C/BE[3:0]# and AD[31:0] during the attribute phase.	ROS	0
39:36	**Transaction Command Lower** – The 4-bit value transferred on C/BE[3:0]# during the first address phase.	ROS	0
43:40	**Transaction Command Upper** – The 4-bit value transferred on C/BE[3:0]# during the second address phase of a DAC transaction.	ROS	0
127:64	**Transaction Address** – The 64-bit value transferred on AD[31:0] during the first and second address phases. The first address phase is logged to 95:64 and the second address phase is logged to 127:96. In the case of a 32-bit address, bits 127:96 will be set to zero.	ROS	0

5.2.4. PCI Power Management Capability

Refer to PCI Express Base 1.0a and PCI PM 1.1 for details of the PCI Power Management Capability. The requirements stated in PCI Express Base 1.0a for the PCI PM Capability, that are over and above what is required by PCI PM 1.1, apply to PCI Express bridges.

5.2.5. Slot Numbering Capability

The slot numbering registers are optional. They are required for bridges that connect to PCI expansion chassis. Refer to PCI Bridge 1.2 for additional details on slot numbering.

If the slot numbering registers are supported, the Slot Numbering Capabilities registers, shown in Figure 5-10, must appear in the Capabilities List. The value stored in the Capabilities Pointer registers (offset 34h) points to the Slot Numbering Capabilities registers, if they are the first item in the Capabilities List. If not, a subsequent list item points to the Slot Numbering Capabilities registers.

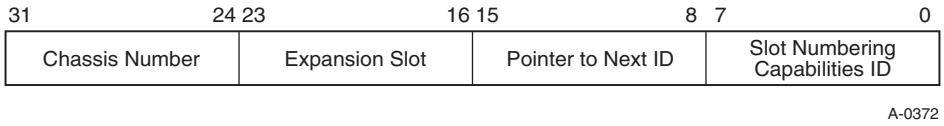

Figure 5-10: Slot Numbering Capabilities Registers

5.2.5.1. Slot Numbering Capabilities ID Register

This register identifies the Capabilities List item as a Slot Numbering registers item. It is read-only and returns the value of 04h when read.

5.2.5.2. Pointer to Next ID Register

This register contains the pointer to the next Capabilities List item, if supported. If there are no subsequent list items, this register will contain the value 0. This register is read-only.

5.2.5.3. Expansion Slot Register

The Expansion Slot register provides information used by system software in calculating the slot number of a device plugged into a PCI slot in an expansion chassis. Refer to PCI Bridge 1.2 for a complete discussion of the use of the Expansion Slot register.

The register is read-only and is initialized by hardware after any primary interface reset (i.e., PCI Express hot, warm, or cold reset). The method by which the system designer establishes the default value of this register is not controlled by this specification, but the content must be valid when the PCI system initialization software reads the register to determine how the system is configured. Any alternative that guarantees the contents will be valid before the PCI system initialization software executes is acceptable. For example, the bridge could initialize the Expansion Slot register based on the state of certain pins on the bridge at reset time. In this approach, the expansion chassis designer

pulls these pins up or down based on how the expansion chassis is wired. After reset, the pins assume their normal functions. In another example, the inputs to an external shift register could be hardwired with this information and the shift register read by the bridge at reset time. More elaborate schemes involving serial EEPROMs would be possible as well.

Table 5-18: Expansion Slot Register

Bit Location	Register Description	Attributes
4:0	**Expansion Slots Provided** – Contains the binary value of the number of PCI expansion slots located directly on the secondary interface of this bridge. Expansion slots located behind additional (subordinate) bridges on the secondary interface are not counted in this field.	HwInit
5	**First in Chassis** – If this bit is set, it indicates that this bridge is the first in an expansion chassis. A bridge with this bit set indicates the existence of an expansion chassis that requires a unique chassis number. 0 - This is not a parent bridge. 1 - This is a parent bridge.	HwInit

5.2.5.4. Chassis Number Register

The Chassis Number register contains the physical chassis number for the slots on the bridge's secondary interface. A different non-zero chassis number is assigned by system initialization software to each separate expansion unit that contains PCI expansion slots. Multiple bridges contained in the same chassis are assigned the same chassis number. Chassis number 0 is reserved for the chassis containing the CPU that initializes the Configuration Space for the system.

This register is read/write and may optionally be either non-volatile or initialized to zero by reset. If the system configuration software finds the number in this register is non-zero and does not conflict with another chassis number, the system configuration software will leave the register value unchanged. If the system configuration software finds this register is zero or that it conflicts with another chassis number, the configuration software will write a new chassis number into this register.

The method of storage and retrieval of the non-volatile chassis number is not controlled by this specification, so any technology that holds information across a power cycle will suffice. For example, the register can be a simple read/write register with a battery backup. Alternatively, the chassis number can be stored in a serial EEPROM and shifted in at power up.

Making the Chassis Number register non-volatile provides the most capability to the end user of an expansion chassis. In this case, even if bridges to expansion chassis are rearranged in the system, the chassis number remains unchanged unless a chassis is moved from one system to another and causes a duplication of chassis numbers. Therefore, this alternative is recommended for bridges that have non-volatile storage capabilities.

Refer to PCI Bridge 1.2 for a complete discussion of the use of the Chassis Number register.

6

6. Interface Signals

The following sections are intended to aid those interested in understanding the scope and nature of the signals relevant to the PCI Express bridge architecture. Signals listed as required are necessary for all compliant bridge implementations, while those listed as optional may or may not be incorporated into a particular bridge implementation.

A particular implementation may support signals associated with integrated features (e.g., a system management port) that are not covered in this chapter.

6.1. PCI Express Interface

The following signals are associated with the PCI Express interface of a bridge.

6.1.1. Required Signals

Table 6-1: Required Signals Associated with the PCI Express Interface

Signal Name	Type	Function/Notes
PETpx, PETnx	High-Speed LV Differential Output	Transmitter differential pair, where x is 1, ...N. Valid N are 1, 2, 4, 8, 12, 16, and 32. "p" and "n" denote positive and negative, respectively.
PERpx, PERnx	High-Speed LV Differential Input	Receiver differential pair, where x is 1, ...N. Valid N are 1, 2, 4, 8, 12, 16, and 32. "p" and "n" denote positive and negative, respectively.

6.1.2. Optional Signals

In addition to the signals that are required to transmit/receive data on the PCI Express interface, there are signals that may be necessary to implement the PCI Express interface in a system environment, or to provide certain desired functions (Table 6-2). These signals are referred to as auxiliary signals and are listed herein as optional bridge signals.

Table 6-2: Optional Signals Associated with the PCI Express Interface

Signal Name(s)	Type	Function/Notes
REFCLK+, REFCLK-	Low-Voltage Differential Input	100-MHz reference clock
PERST#	Input	Indicates when main power is within specified tolerance and stable
JTAG (TRST#, TCK, TDI, TDO, TMS)	Compliant with IEEE 1149.1	IEEE Test Access Port and Boundary Scan interface
SMBus (SMCLK, SMDAT)	Compliant with *System Management Bus, Version 2.0*	SMBus interface clock and signals
WAKE#	OD, uses +3.3V Aux	Signal for Link reactivation
+3.3V Aux	3.3V DC Power Input	Auxiliary power supply rail

6.2. Conventional PCI/PCI-X Interface

The signals described in this section are associated with the secondary interface of a PCI Express bridge. Descriptions are provided that are relevant to both conventional PCI and PCI-X.

6.2.1. Required Signals

The signals listed in Table 6-3 are the minimum requirement for a bridge that only implements a 32-bit conventional PCI interface.

Table 6-3: Minimum Signal Requirements for a 32-bit Conventional PCI Interface

Signal Name(s)	Type	Function/Notes
AD[31:0], PAR, C/BE[3:0]#, REQ# (Output)[13]	Tri-State	See PCI 3.0.
FRAME#, IRDY#, TRDY#, STOP#, DEVSEL#, PERR#	Sustained Tri-State (Conventional PCI Mode)	See PCI 3.0.
IDSEL, CLK (Input),[14] GNT# (Input)[15]	Input	See PCI 3.0.
RST# (Output), LOCK# (Output)	Output	See PCI 3.0.
SERR#	OD	See PCI 3.0.

[13] A REQ# output is not required if the bridge integrates the PCI bus arbitration logic.

[14] A CLK input is not required if the bridge integrates the PCI common-clock driver.

[15] A GNT# input is not required if the bridge integrates the PCI bus arbitration function.

6.2.2. Optional Signals

PCI 3.0 defines interface signals that are optional (e.g., used to support central resource functions) for PCI Express bridges. These signals are listed in Table 6-4 with their given PCI 3.0 signal names and bridges are permitted to implement them in addition to the signals that are required to support a 32-bit conventional PCI interface. Table 6-5, lists the additional signals (with their given PCI-X signal names) defined for the optional support of PCI-X Mode 1 and Mode 2. As stated in PCI-X PT 2.0a, 64-bit bus width extension support and ECC support are optional for PCI-X Mode 1. The signals necessary to add 64-bit support are the same for conventional PCI and PCI-X mode 1. PCI-X Mode 1 ECC support requires the multiplexing of ECC signals onto various other PCI-X signals in the same manner as performed in PCI-X Mode 2 (see PCI-X 2.0a for details).

Optional signals relevant to conventional PCI include central resource functions, such as PCI bus arbitration, PCI clock distribution, interrupt collection, as well as signals related to the 64-bit bus extension, 66-MHz support, capability identification, and test functions. These signals are presented in Table 6-4. Hot-plug controller signals are not included in the following descriptions due to the implementation-specific nature of the signals involved. The bridge may support other implementation-specific signals but the ones listed are expected to be the most prevalent options.

Table 6-4: Optional Signals Relevant to a Conventional PCI Interface (Includes Central Resource Functions) on a Bridge

Signal Name(s)	Type	Function/Notes
REQ64#, ACK64#	Category 1, 3.3V Sustained Tri-State	64-bit extension; see PCI 3.0 and PCI-X 2.0a.
AD[63:32], PAR64, C/BE[7:4]#	Category 1, 3.3V Tri-State	64-bit extension; see PCI 3.0 and PCI-X 2.0a.
TDI, TDO, TCK, TMS, TRST#	Category 4, Compliant with IEEE 1149.1	IEEE Test Access Port and Boundary Scan interface
M66EN	Category 4, Input	See PCI 3.0.
PRSNT[2:1]#	Category 4, Input	See PCI 3.0 and SHPC 1.0; one or more pairs are optional.
REQ[n]# (Input)	Category 2, Input	Request inputs for integrated PCI bus arbitration feature, where n signifies the number of request and grant signal pairs. See PCI 3.0 and PCI-X 2.0a.
GNT[n]# (Output)	Category 2, Output	Grant outputs for integrated PCI bus arbitration feature, where n signifies the number of request and grant signal pairs. See PCI 3.0 and PCI-X 2.0a.
PME# (Input)	Category 5, Input	See PCI PM 1.1.
INTA-D# (Input)	Category 3, Input	See PCI 3.0 and PCI-X 2.0a.
3.3V Aux	3.3 V DC Power Input	See PCI 3.0.

Signal Name(s)	Type	Function/Notes
CLKI	3.3 V, Input	PCI clock feedback for integrated clock driver
CLK0[n]	3.3V, Output	Integrated clock driver, where n signifies the number of clock outputs provided

The interface signals relevant to a PCI-X capable secondary interface on a bridge are shown in Table 6-5. The table lists the signals relevant to an interface that includes the optional 64-bit extension. The ECC signals (which are optional for Mode 1) are shown.

Table 6-5: Signals Relevant to a PCI-X Capable Interface (Optional 64-bit Extension Included) on a Bridge

Signal Name(s)	Type	Function/Notes
AD[63:0], C/BE[7:0]#, PAR64/ECC[7], REQ64#/ECC[6], ECC[5:2], ACK64#/ECC[1], PAR/ECC[0]	Category 1, 3.3 V or 1.5 V	See PCI-X 2.0a.
FRAME#, IRDY#, TRDY#, STOP#, DEVSEL#, IDSEL, PERR#, REQ# (Output)[16]	Category 2	See PCI-X 2.0a.
IDSEL, CLK (Input)[17], GNT (Input)[18]	Category 2	See PCI-X 2.0a.
RST# (Output), LOCK# (Output)	Category 2	See PCI-X 2.0a.
SERR#	Category 3, OD	See PCI-X 2.0a.
PCIXCAP, MODE2	Category 6, Input	See SHPC 1.0 and PCI-X 2.0a.

[16] A REQ# output is not required if the bridge integrates the PCI bus arbitration function.

[17] A CLK input is not required if the bridge integrates the PCI common-clock driver.

[18] A GNT# input is not required if the bridge integrates the PCI bus arbitration function.

7

7. Initialization Requirements

The actions that a bridge must take upon receipt of various reset events and interface initialization requirements are described in the following sections.

7.1. Reset Behavior

There are two types of reset that a PCI Express bridge will receive over the PCI Express primary interface:

- Physical layer resets that are platform specific and referred to as Fundamental Resets. These resets are categorized into either cold reset or warm reset depending on whether the bridge's main power is cycled with the reset or not, respectively.
- Protocol resets including PCI Express Hot Reset signaled through the reception of two consecutive TS1 ordered sets with the Hot Reset bit set (see Chapter 4 of PCI Express Base 1.0a for details), and an implicit primary reset triggered by the PCI Express primary interface transitioning to DL_Down status. These protocol resets are treated as hot resets by the bridge.

These primary interface reset sources are each described in sections that follow. All primary interface reset events initiate a Secondary Bus Reset which resets the secondary interface(s) of the bridge using the PCI bus reset mechanism (see PCI 3.0 for details).

In addition to primary interface reset sources, the bridge is also required to support specific mechanisms that permit a targeted reset of its secondary interface(s). These mechanisms are also discussed below and include Secondary Bus Reset via the Bridge Control register or a Standard Hot-Plug Controller (optional) reset action.

When attempting a Configuration access to devices on a conventional PCI or PCI-X segment downstream of a PCI Express bridge, the timing parameter T_{rhfa} must be respected after reset.

7.1.1. Fundamental Reset (Cold/Warm Reset)

A PCI Express bridge implementation must support a means of determining when the device is receiving main power that is both stable and within the specified tolerances. This Fundamental Reset state may be conveyed to the bridge device via the optional PCI Express PERST# signal or an implementation-specific signal. Generally, bridge logic (state machines and registers) will enter reset asynchronously when this signal is asserted. When Fundamental Reset follows the application of main power as previously described, it is referred to as a cold reset. The platform-specific power logic may also generate a Fundamental Reset without removing the bridge's main power; this is referred to as a warm reset. The bridge must treat cold and warm resets without distinction.

Upon a warm or cold reset, a bridge is required to return all internal state machines to their default states and initialize all register values to their defaults, including sticky register values (e.g., error status) that are supplied with main power but that retain their values through hot reset. When noted, sticky register values in bridges that have AUX power consumption enabled (via AUX Power PM Enable or PME Enable) are not affected by Fundamental Reset (see Chapter 6 of PCI Express Base 1.0a). To eliminate potential system device interoperability problems, bridges are also required to either tri-state their outputs or to drive them to safe levels during such a reset. Specific input/output requirements may also apply if bus parking is implemented and the bridge is the parking agent. See PCI Express Base 1.0a, PCI 3.0, and PCI-X 2.0a for additional details.

The bridge is required to propagate warm/cold reset from its primary interface to PCI reset on all secondary interfaces. Until a Fundamental Reset state is achieved, the bridge must keep all secondary interface reset signals asserted. With respect to conventional PCI and PCI-X secondary interfaces, additional minimum reset assertion time requirements apply. The conventional PCI and PCI-X minimum reset assertion time requirements differ among bus modes. Refer to PCI 3.0 and PCI-X 2.0a for specific requirements.

7.1.2. Primary Reset Due to Hot Reset

The reception on the primary interface of two consecutive TS1 ordered sets with the Hot Reset bit set must be interpreted by a bridge as hot reset. When a bridge receives a hot reset on its PCI Express primary interface, it must propagate that reset to all secondary interfaces using the PCI reset signal (see PCI 3.0 and PCI-X PT 2.0a for details). In addition, the bridge is required to discard all transactions being processed and return all registers, state machines, and externally observable state internal logic to the state specified default or initial conditions (except for registers defined as sticky).

Software is responsible for ensuring that hot reset assertion and deassertion are timed such that a bridge will adhere to proper reset assertion and deassertion durations on its secondary interface.

7.1.3. Primary Reset Due to DL_Down Link Status (Hot Reset)

When the PCI Express (primary) interface of a bridge is in normal operation and, for whatever reason, the Link goes down, the Transaction and Data Link Layers will enter the DL_Down status (see Chapter 2 of PCI Express Base 1.0a). The bridge is required to discard all transactions being processed and return all registers, state machines, and externally observable state internal logic to the state specified default or initial conditions (except for registers defined as sticky). In addition, the entry of the primary interface of the bridge into DL_Down status must initiate a reset of the secondary interface(s) of the bridge using the PCI bus reset signal (see PCI 3.0 for details). This reset may be asserted and deasserted asynchronously with respect to the PCI common clock.

7.1.4. Secondary Bus Reset via the Bridge Control Register

A reset of the secondary interface may be initiated by software through assertion of the Secondary Bus Reset bit in the Bridge Control register (Section 5.1.2.13). This targeted reset may be used for various reasons, including recovery from error conditions on the secondary bus, to initiate re-enumeration, to change the operating frequency of the bus (33/66/100/133 MHz), or to change the operating mode of the bus (conventional PCI or PCI-X). Changes in bus mode or frequency are not permitted unless the Secondary Bus Reset bit is asserted. A write to the Secondary Bus Reset bit will force the assertion of the secondary interface PCI reset (RST#) signal without affecting the primary interface or any configuration space registers. Additionally, the logic associated with the secondary interface is reinitialized and any transaction buffers associated with the secondary interface are cleared.

RST# is asserted as long as the Secondary Bus Reset bit is asserted so software must take care to observe proper conventional PCI and PCI-X reset timing requirements. A Secondary Bus Reset initiated via the Bridge Control register may be asserted and deasserted asynchronously with respect to the PCI common clock (refer to PCI 3.0 for details).

Software is responsible for ensuring that the bridge will not receive transactions that require forwarding to the secondary interface while Secondary Bus Reset is asserted.

7.1.5. Secondary Bus Reset via a Hot-Plug Controller

PCI Express bridges may support an optional Standard Hot-Plug Controller (SHPC) function that is associated with the secondary interface. The SHPC function may initiate a Secondary Bus Reset through a write to the hot-plug Command register with the Change Frequency command. A write to the hot-plug Command register may result in one of the following to occur on the secondary bus:

- Change of common-clock frequency (33/66/100/133 MHz).
- Change of mode (conventional PCI or PCI-X, ECC or parity).
- Change nothing. Present settings are rewritten to the register.

Any write to this register (resulting in one of the actions listed above) will lead to a reset of the associated secondary interfaces. This reset will cause the instructed bus frequency and/or mode changes to take effect. This reset may be asserted and deasserted asynchronously with respect to the PCI common clock (refer to PCI 3.0 for details).

7.1.6. Bus Parking During Reset

All bridges are required to drive the secondary bus AD[31:0], C/BE#[3:0], and PAR signals to a logic low level (zero) when the secondary interface's RST# is asserted. This requirement is independent of the location of the secondary bus arbiter (internal or external to the bridge). A small finite delay (a few clocks) is permitted from the assertion of the secondary RST# signal until the bridge drives the secondary signals to zero. This delay is intended to allow time for the internal logic that is responsible for driving AD[31:0], C/BE#[3:0], and PAR to zero to respond to the primary

interface reset or Secondary Bus reset event. During the time interval (if any) from the assertion of the secondary interface RST# signal and the parking of the AD[31:0], C/BE#[3:0], and PAR signals to zero, the bridge must tri-state these same signals.

See PCI-X PT 2.0a for bus parking requirements specific to the PCI-X modes.

7.2. Secondary Bus Mode and Frequency Initialization for PCI-X Modes

A bridge that supports PCI-X places its secondary bus in PCI-X mode based on the capabilities of the secondary bus and the devices connected there. Placing the secondary interface into a PCI-X mode may involve changes to signals that the bridge controls, such as the clock and V I/O. Refer to PCI-X 2.0a for specific requirements regarding these signals. Some of the requirements listed for PCI-X bridge in PCI-X 2.0a involve events triggered by primary reset. For a PCI Express bridge, these events will be triggered instead by the reception of a Fundamental Reset or hot reset (including the PCI Express Link transitioning to DL_Down status).

7.3. Initialization by System Software

When bridges are present in a system, system software will typically provide the following functions during the initialization process:

- Assign PCI Bus Numbers.
- Allocate address spaces (prefetchable memory, memory mapped I/O, and I/O).
- Program the IRQ or other system architecture-specific value into the Interrupt Line register (see Section 5.1.1.8).
- Initialize the PCI display subsystem.

Each function will be discussed in the following sections.

7.3.1. Assigned Bus Numbers

The system software must assign PCI Bus Numbers to each bridge in the system. The order of the assignments and when the assignments are made is not specified. All buses located behind a bridge must reside between the Secondary Bus Number and the Subordinate Bus Number (inclusive).

Prior to changing PCI Bus Numbers during system operation, software should ensure that outstanding transactions involving the bridge are completed. This mechanism is implementation specific (see PCI-X PT 2.0a and PCI Express Base 1.0a for details).

7.3.2. Allocating Address Spaces

When the system software enumerates a bridge, it must map all devices that reside below the bridge into one or more of the I/O, the memory mapped I/O, or the prefetchable memory address ranges supported by the bridge. The 16-bit I/O address range (see Section 3.2) has a granularity of 4 KB

(aligned) and a maximum size of 64 KB. This range is restricted to the first 64 KB of the PCI I/O address space (0000 0000h to 0000 FFFFh). If 32-bit I/O addressing is supported, the I/O address range is permitted to reside anywhere in the 4-GB PCI I/O address space (the 4-KB granularity still applies). The memory mapped I/O range (see Section 3.1) can reside anywhere in the lower 4 GB of address space, with a granularity of 1 MB (aligned). The prefetchable memory range (see Section 3.1.2) can reside in a 32- or 64-bit address space with a granularity 1 MB (aligned). Since bridges have only one range for each address space type (I/O, memory mapped I/O, or prefetchable memory), the system software must group all devices that use the same type of space into a single range. This implies that when a bridge has multiple bridges and/or devices behind it, the system software must be able to group them into a single range per address space type. For example, in Figure 7-1, device 1 on bus 2 and device 2 on bus 3 both require I/O address space (all other devices do not require I/O space). For the PCI Express bridge that resides between buses 1 and 2 to handle the decode of I/O space, the two requests for I/O space need to be mapped such that the bridge can decode all I/O transactions with a single range decode and forward them downstream. In this example, which is illustrated in Figure 7-2, device 1 on bus 2 would be assigned an I/O range (by programming its I/O BAR) and device 2 on bus 3 would be assigned a different I/O range (by programming its I/O BAR). To be decoded, these two ranges need to fall within the range defined by the bridge's I/O Base and Limit registers. The PCI Express bridge's I/O Base register is programmed with the value of device 1's I/O BAR register while its I/O Limit register is programmed with the value of device 2's I/O BAR. The PCI bridge is programmed such that its I/O Base and Limit registers match device 2's I/O address space requirements. The Root Complex would be programmed the same as the PCI Express bridge.

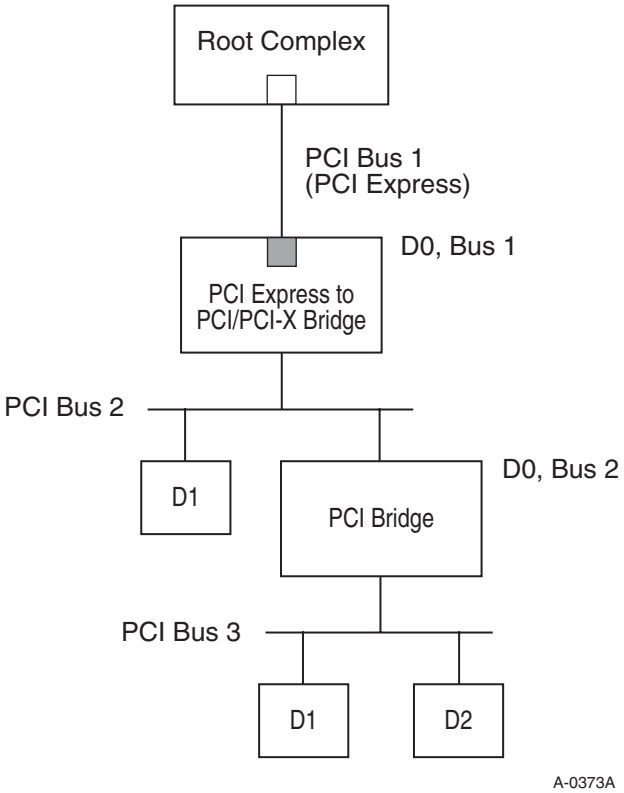

Figure 7-1: Example of Bus Hierarchy

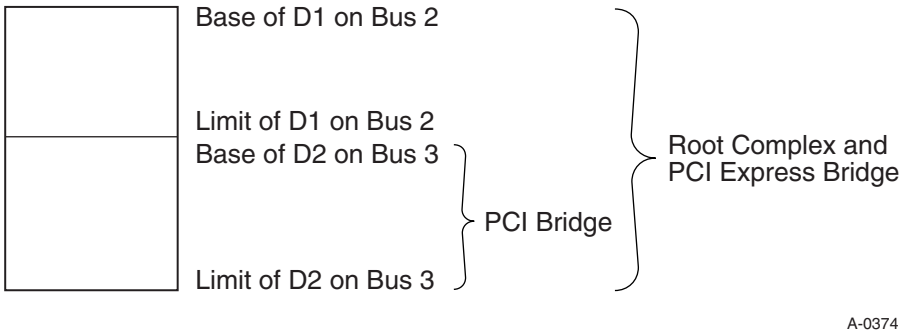

Figure 7-2: Example of Address Range Coalescing

The same methodology is required for the memory mapped I/O space and prefetchable memory space. The only difference is the number of devices that need to be assigned space and the size of the space for each device. To minimize the wasted space, the largest requests should be allocated first then filled in with smaller requests in the same contiguous range.

To summarize, all devices behind a given bridge must be allocated I/O, memory mapped I/O, or prefetchable memory space in such a way that the bridge can implement a single address range for each type.

7.3.3. PCI Display Subsystem Initialization

As specified in PCI 3.0 and in this document, the VGA Enable bit in the Bridge Control register, the bit which controls a bridge's response to VGA accesses, and the VGA 16-bit Decode bit (in the same register), the bit which determines if 10- or 16-bit VGA addressing decoding is used, are hardware initialized to 0h (not enabled) at power-up.

The software or firmware algorithm used to initialize the PCI subsystem must set the bridge's VGA Enable bit to 1h if the VGA device used for boot is discovered downstream of the bridge. Software/firmware responsible for VGA boot device discovery and for programming the bridge's Command register may optionally enable 16-bit VGA address decoding using the VGA 16-bit Decode bit.

VGA palette snooping is not applicable to PCI Express bridges and, therefore, the VGA Palette Snoop enable in the Command register is hardwired to 0h.

See Section 11.1 for additional information on VGA support.

8

8. Interrupt Support

The PCI Express bridge architecture provides for both interrupt routing from conventional PCI/PCI-X secondary interface(s) to the upstream PCI Express Link and for the in-band forwarding of interrupts generated by interrupt sources integrated into the bridge (e.g., SHPC). Interrupts may be conveyed using either PCI INTx interrupts or Message Signaled Interrupt transactions (MSI/MSI-X) as described below. MSI/MSI-X transactions are Memory Writes and, therefore, all bridges support their forwarding. The collection of PCI INTx interrupts from the secondary interface(s) is required of bridges except when the bridge is specifically designed for system board applications in which the interrupts are routed around the bridge.

The generation of MSI/MSI-X transactions by a bridge is optional, as described in Section 8.3. However, if a bridge integrates interrupt sources it is required to support generation of interrupts on behalf of these sources using the MSI mechanism, the MSI-X mechanism, or both. Additionally, bridges are required to support generation of virtual wire INTx interrupts on behalf of sources internal to the bridge, as described in Section 8.2.

8.1. Interrupt Routing

For system firmware and software to properly associate a device within or downstream of a bridge with its respective INTx interrupt, bridges must follow prescribed mapping rules when forwarding PCI INTx interrupts as PCI Express Assert_INTx and Deassert INTx messages. The mapping rules are described in Section 8.2. In the case of a bridge on a system board in which the interrupts are not routed through the bridge, the system firmware is required to report the mapping of INTx interrupts to system software.

8.2. PCI INTx Interrupts

The Message Signaled Interrupt (MSI/MSI-X) mechanism is the preferred interrupt signaling mechanism for the PCI Express interface. However, in some systems, there may be PCI devices subordinate to the bridge that do not support the MSI/MSI-X mechanism. The INTx virtual wire interrupt signaling mechanism, as described in PCI Express Base 1.0a, Chapter 2, is used by the PCI Express bridge in cases where the MSI/MSI-X mechanism cannot be used.

PCI INTx interrupts are "virtualized" in PCI Express using Assert_INTx and Deassert_INTx messages, where x is A, B, C, and D for the respective PCI INTx# interrupt signals defined in PCI 3.0. This message pairing provides a mechanism to preserve the level-sensitive semantics of the PCI interrupts. The Assert_INTx and Deassert_INTx messages transmitted on the PCI Express Link capture the asserting/deasserting edge of the respective PCI INTx# signal.

The Requester ID used in the PCI Express Assert_INTx and Deassert_INTx messages transmitted by the bridge (irrespective of whether the source is internal or external to the bridge) must equal the bridge's primary interface Bus Number and Device Number. The Function Number sub-field must be set to zero (refer to PCI Express Base 1.0a for more details).

Interrupt sources internal to the bridge are masked by the Interrupt Disable bit of the Command register (see Section 5.1.1.1).

Multi-ported PCI Express bridges must collapse the INTA#-INTD# pins from each of their downstream conventional PCI/PCI-X interfaces into four INTx "virtual wires" on their Upstream Port. For multi-ported bridge implementations that follow Option B illustrated in Figure 1-4, the mapping between the INTx# pin on a conventional PCI/PCI-X bus and the corresponding INTx messages on the PCI Express Link is based on the device number of the PCI/PCI-X bridge assigned to the port requesting the interrupt. The mapping is shown in Table 8-1. Interrupts for sources internal to the bridge are logically ORed with the interrupts from the secondary interface after the mapping in Table 8-1 is applied to the secondary interface signals. Bridge implementations that follow Option A in Figure 1-4 do not apply the mapping listed in Table 8-1 but, rather, simply collapse the secondary interrupts into their respective INTx "virtual wire" messages (e.g., all PCI INTA signals are logically ORed and forwarded as Assert_INTA and Deassert_INTA messages on PCI Express).

PCI Express Assert_INTx and Deassert_INTx messages are not masked by the Bus Master Enable bit located in the Command register (see Section 5.1.1.1).

Table 8-1: PCI INTx Mapping to INTx Virtual Wires for Option B, Figure 1-4

Device Number of Conventional PCI/PCI-X Bridge Supporting Secondary Interface (Interrupt Source)	INTx# Interrupt Line from Downstream Conventional PCI/PCI-X Interface	Mapping to INTx Virtual Wire on Primary Side of Bridge
0,4,8,12,16,20,24,28	INTA#	INTA
	INTB#	INTB
	INTC#	INTC
	INTD#	INTD
1,5,9,13,17,21,25,29	INTA#	INTB
	INTB#	INTC
	INTC#	INTD
	INTD#	INTA
2,6,10,14,18,22,26,30	INTA#	INTC
	INTB#	INTD
	INTC#	INTA
	INTD#	INTB
3,7,11,15,19,23,27,31	INTA#	INTD
	INTB#	INTA
	INTC#	INTB
	INTD#	INTC

IMPLEMENTATION NOTE

INTx# Interrupt Routing in Multi-Ported Bridges

Multi-ported PCI Express bridges implemented as illustrated in Option B of Figure 1-4 must collapse interrupts across multiple conventional PCI/PCI-X buses following the same rules as described for switches. As shown in Figure 8-1, a dual-headed PCI Express bridge would map (or "swizzle") the interrupts per Table 8-1 (in accordance with the respective logical PCI/PCI-X bridge's Device Number), collapse the INTA#-INTD# signals from its two logical PCI/PCI-X bridges, collapse the INTA#-INTD# signals from any internal sources, and convert the signals to in-band PCI Express messages.

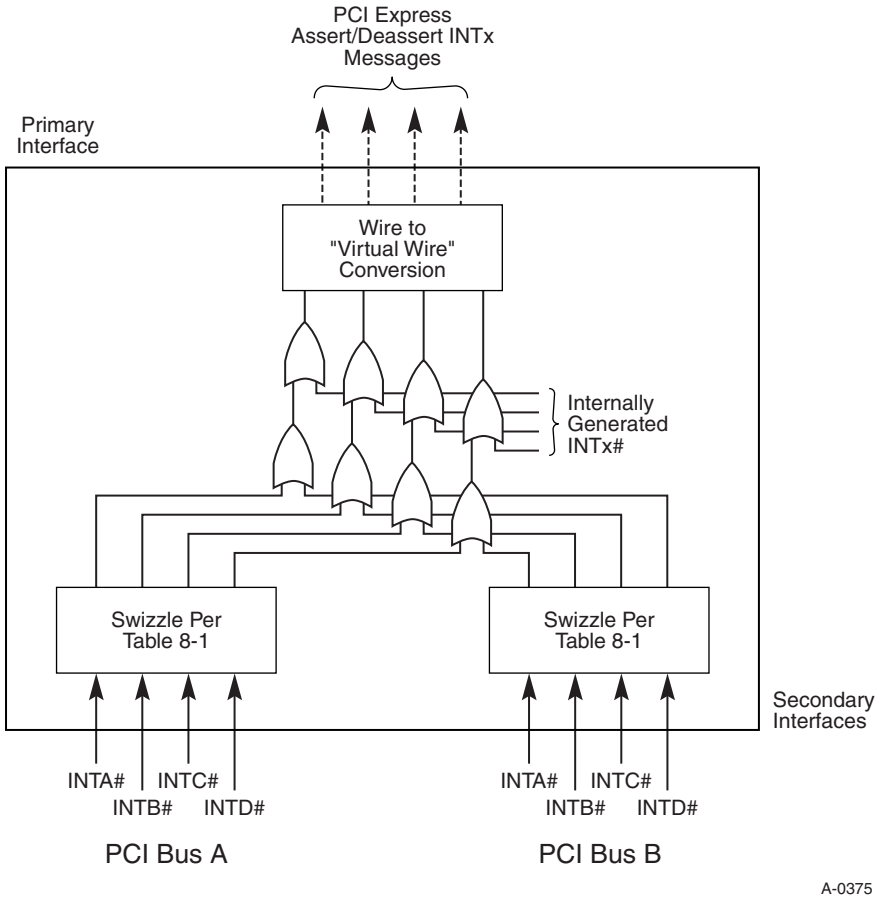

Figure 8-1: INTx# Collapsing and Conversion to INTx Messages in a Dual-Headed Bridge

8.3. MSI Interrupts

PCI Express bridges with internal sources that generate interrupts are required to support the MSI mechanism, the MSI-X mechanism, or both. If an internal source supports MSI, it must use the 64-bit Message Address version of the MSI capability structure. Implementation of these features is specified in PCI 3.0.

The requester of an MSI/MSI-X transaction must clear the No Snoop and Relaxed Ordering bits in the Requester Attributes.

8.4. Interrupt Synchronization

PCI 3.0 requires the interrupt handler (service routine), or the device which originates the INTx interrupt, to guarantee that all buffers are flushed between the device and the final destination. This can be accomplished by the interrupt service routine of the device driver by performing a read of the device after receiving an INTx interrupt or by the device itself performing a read of the location last written by the device prior to asserting an INTx# interrupt signal. In either case, the read will force buffers between the device and the final destination to be flushed.

No special buffer flushing requirements exist for devices that use MSI/MSI-X mechanism since MSI/MSI-X transactions, as posted transactions, push other posted transactions ahead of them.

9

9. Power Management

The bridge's role in power management event signaling, including its use of power management messages, is described in the following sections.

9.1. Basic Requirements

PME-capable PCI devices assert the PME# pin to signal a power management event. Bridges convert the PME# signal from the secondary interface to in-band PCI Express PME Messages. The collection and translation of physical PME# signals from the secondary interface of a bridge is required of bridges except when the bridge is specifically designed for system board applications in which the PME# signals are routed around the bridge and directly to the Root Complex (see Chapter 7 of PCI Express Base 1.0a for details). Methods of grouping and combining PME# signals from PCI devices on buses downstream of the bridge is implementation specific as it is in PCI-based systems today.

When converting from PCI level-triggered PME# signaling to edge-triggered PCI Express PME messages, care must be taken not to lose any PMEs from PCI devices. See the Implementation Note on page 147 for details.

9.2. Power Management Signaling

Power Management Messages are used to support Power Management Events (PME) signaled by sources integrated into the bridge and for devices downstream of the bridge. Note that the generation of Power Management Messages on the PCI Express interface must be implemented according to the mechanisms described in PCI Express Base 1.0a. This process consists of two steps: Link reactivation (see Chapter 5 of PCI Express Base 1.0a) and transmission of the PME Message.

To permit the system software to identify the device that signaled the PCI Power Management Event using a PM_PME message that is generated by the bridge, bridges must do the following:

- When the PME comes from a legacy agent on a PCI bus downstream, the PM_PME Message Requester ID reports the secondary interface Bus Number from which the PME was collected, and the Device Number and Function Number reported must both be zero.
- When the PCI Express bridge initiates a PME message on behalf of an internal source (e.g., on behalf of a SHPC), the Requester ID consists of the bridge's primary interface Bus Number, Device Number, and Function Number.

Figure 9-1 depicts an example of Power Management Event signaling involving a multi-ported PCI Express bridge with an integrated SHPC and PME# pin inputs on two secondary interfaces. In this

scenario, a PME generated by the SHPC associated with the bridge with Function Number 01h would result in the bridge generating a PM_PME message with a Requester ID comprised of a Bus Number field value of 06h, a Device Number field value of 0h, and Function Number field value of 01h. In a second example, the PME# input from PCI Bus 7 is asserted. This would result in the bridge generating a PM_PME message with a Requester ID comprised of a Bus Number field value of 07h, a Device Number field value of 0h, and Function Number field value of 0h.

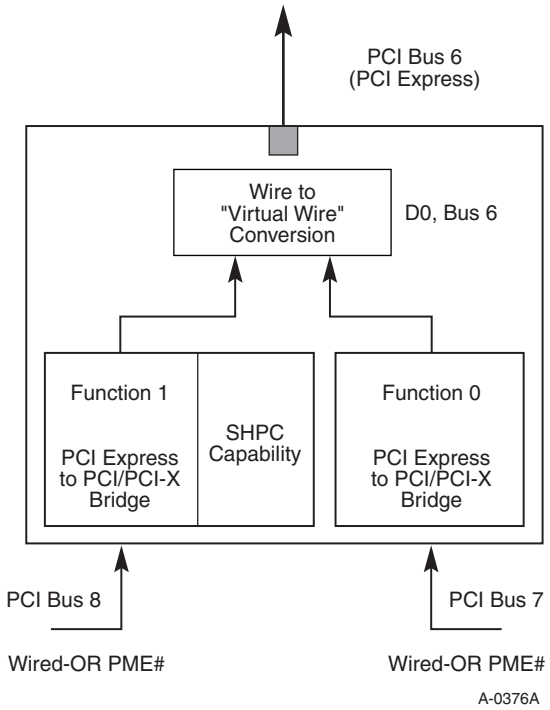

Figure 9-1: Example of PME Message Signaling in a PCI Express Bridge

It is the responsibility of the PCI Express bridge to engage in the PME fence protocol on behalf of its downstream PCI devices. The PCI Express PME_Turn_Off message will terminate at the PCI Express bridge and will not be communicated to the downstream PCI devices.

Note that a PCI Express bridge cannot issue a PM_PME message on behalf of a downstream PCI device while its upstream Link is in the L2 non-communicating state. In this case, the PCI Express interface logic of the bridge must first initiate a transition into the active Link state (L0) before the PM_PME message can be communicated. This is accomplished using the Wake mechanism as defined in PCI Express Base 1.0a. Note that depending on the intended application of the PCI Express bridge component (e.g., integration on the motherboard or on an add-in card), the supported Wake mechanism may be implemented using a sideband WAKE# signal or an in-band Beacon mechanism.

IMPLEMENTATION NOTE

PCI Express Bridge PME Considerations

PCI Express bridges must collect and translate power management events from the original PME# wired-OR signal connections to the PCI Express in-band PME messaging scheme. PCI Express bridges are required to identify all PME messages that they issue on behalf of downstream PCI devices as coming from the PCI bus segment where the PME# was collected.

A design consideration that must be comprehended is the potential for lost PME indications. This particular issue is unique to PCI Express bridges where the level-sensitive PME# signal is transformed into what is effectively an edge-triggered PME messaging scheme. This architecture can manifest a race condition. The corner case corresponds to the situation where one of the downstream PCI devices asserts a PME# signal that is input to the bridge. Following this assertion, the PCI Express bridge injects a PCI Express PME Message on behalf of the PCI device. If another device asserts PME# (on the same PME# input to the bridge) before the software servicing PM_PMEs clears the PME_Status bit of the first device that asserted PME#, given the wired-OR nature of PME# signaling, the PME# input at the bridge will remain asserted following the clearing of the first device's PME_Status bit.

The net result is that the first device signaling PME# assertion is serviced successfully, (single PM_PME message propagated upstream facilitated this), but the second device's PME# assertion is lost. In order to avoid loss of PME# assertions in the conversion of the level-sensitive PME# signal to the edge triggered PCI Express PM_PME message, the PCI PME# signal must be periodically polled and a PCI Express PM_PME message must be generated if PME# is sensed asserted. While this scheme introduces the possibility of spurious PM_PMEs, these are deemed benign and should not affect the performance of the system.

10

10. Error Handling

The actions that a PCI Express bridge must take in response to errors that arise during transaction processing are described in this chapter. The error handling descriptions are separated according to the interface on which the transaction originated that encountered the error. The originating side of the bridge is the bridge interface that responds as a target to the transaction. For Split Requests and memory writes, this is the side closest to the requester. For Split Completions, this is the side closest to the completer. The destination side is the side of the bridge that is opposite the originating side.

Some of the bridge error support requirements vary depending upon the mode (conventional PCI or PCI-X) of the bridge's secondary interface. Requirements for these two cases are presented together in each section below.

Bridges that are PCI-X Mode 1-capable optionally provide ECC protection for the PCI-X secondary interface. PCI-X Mode 2-capable bridges are required to provide ECC protection for the PCI-X secondary interface when that interface is operating in Mode 2. PCI-X Mode 2-capable bridges optionally provide ECC protection for the PCI-X secondary interface when it is operating in Mode 1.

PCI Express error concepts require appropriate mapping to the PCI error reporting structures and individual sections below cover the appropriate legacy error reporting requirements.

For all errors detected by the bridge while forwarding transactions from PCI Express to PCI-X and vice-versa, the bridge sets the appropriate error status bit (both legacy PCI error bit(s) and PCI Express error status bit(s)) and, when noted, generates an error message on PCI Express. Each detected error condition has a default error severity level (fatal or non-fatal) and, when noted, has a corresponding error message generated on PCI Express. The error severity level is programmable if the bridge supports the PCI Express Advanced Error Reporting capability. The Advanced Error Reporting capability also provides the capability to mask individual errors from generating an error message on PCI Express and also provides a log of the header associated with the transaction that generated an error.

Error message generation on PCI Express is controlled in all cases by three control bits (individual errors may have additional control bits):

- SERR# Enable bit in the primary Command register
- Fatal Error Reporting Enable bit in the PCI Express Device Control register
- Non-Fatal Error Reporting Enable bit in the PCI Express Device Control register

ERR_FATAL PCI Express messages are enabled for transmission if either the SERR# Enable bit or the Fatal Error Reporting Enable bit is set. ERR_NONFATAL messages are enabled if either the SERR# Enable bit is set or the Non-Fatal Error Reporting Enable bit is set.

The Fatal Error Detected, Non-Fatal Error Detected, and Correctable Error Detected status bits are set for the corresponding errors on both the PCI Express and PCI-X interfaces.

If not otherwise stated, the rules for setting the error status bits in the PCI Express Capability Structure and the rules for setting the legacy PCI error status bits pertaining to the primary interface, are as described in PCI Express Base 1.0a. The rules for Advanced Error Reporting (relating to error masks, error severity, header log, and error pointer) for both the PCI Express and conventional PCI/PCI-X interface related errors are the same as described for Advanced Error Reporting in PCI Express Base1.0a. Note that conventional PCI/PCI-X interface related errors are logged and reported through a separate set of AER registers from those used for error logging and reporting for PCI Express-related errors in the AER capability. For descriptions and address locations of these registers, refer to Section 5.2.3.

10.1. PCI Express Originating Interface

This section describes the bridge error support requirements for transactions that cross the bridge if the originating side of the bridge is a PCI Express interface and the destination interface is operating in any of the conventional PCI/PCI-X modes. Error reporting on transactions from PCI Express targeted at bridge internal functions follow the rules for a PCI Express Endpoint. The tables that follow summarize the error forwarding requirements (specific to transaction translation) described in the following sections. For the purpose of the error forwarding discussion, PCI Express completion error cases that are translated into Split Transactions (Split Completions or Split Completion Messages) on the conventional PCI/PCI-X interface are treated as originating from the PCI Express interface (see Figure 10-1) and are discussed in this section. PCI Express completion error cases that result in immediate completions on the conventional PCI/PCI-X interface are discussed in Section 10.2.

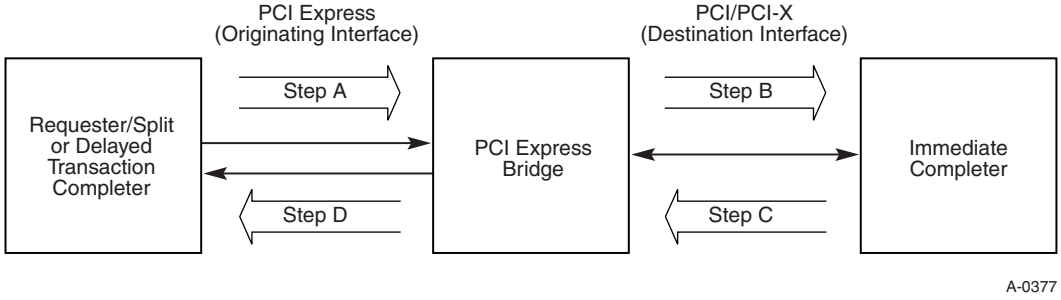

Figure 10-1: Transaction Error Forwarding Flow Diagram with PCI Express as the Originating Interface

Table 10-1 lists the requirements for forwarding uncorrectable data errors from PCI Express to conventional PCI/PCI-X. The first case is the reception of a Write Request or Read Completion with a Poisoned TLP (i.e., EP field has a value of 1b). In this case, the entire data payload of the PCI Express transaction must be considered as corrupt. If the secondary interface is operating in a mode that supports parity, the parity must be inverted (i.e., corrupted) on every data phase forwarded. If the secondary interface is operating in a mode that supports ECC, the ECC signature for every data phase must be modified ("poisoned") as follows:

- Generate the ECC signature(s) for the secondary interface using the data actually received (including the error).

- Include one of the ECC error signatures (see PCI-X PT 2.0a, Chapter 5) as follows:
  - For 32-bit data on the destination interface, use the PEL signature on all data phases.
  - For 64-bit data on the destination interface, use the PED signature on all data phases.
- Drive AD and C/BE# on the destination interface as in the case that no error exists.
- Drive the newly generated (poisoned) ECC signature(s) on ECC on the destination interface with each data subphase.

The resulting combination of AD, C/BE#, and ECC is still protected against single bit failures (if one occurs, the codeword appears to have a double bit error).

The inclusion (using XOR) of one of the PEL or PED signatures in the ECC signature forces even parity in the forwarded codeword. (An error indication codeword always has even parity.)

Table 10-1: Error Forwarding Requirements (Step A to Step B of Figure 10-1) for Received PCI Express Errors

Received PCI Express Error (Step A)	**Forwarded Conventional PCI/PCI-X Error (Step B)**	
	Conventional PCI/PCI-X Mode 1 (Parity)	**PCI-X Mode 1 (ECC) and Mode 2**
Write Request or Read Completion with Poisoned TLP	Poisoned Data Parity	Data Phase ECC with PEL or PED Signature
Request with ECRC (Optional Support) Error	Do Not Forward	

Table 10-2 provides the translation a bridge has to perform when it forwards a non-posted PCI Express request (read or write) to conventional PCI/PCI-X and the request is completed immediately on conventional PCI/PCI-X either normally or with an error condition. For details on these error cases as well as Split Response terminations with uncorrectable data errors, refer to Section 10.1.2.

Table 10-2: Error Forwarding Requirements (Step C to Step D of Figure 10-1) for Transactions Requiring a Completion (Immediate Response)

Immediate conventional PCI/PCI-X Termination (Step C)	**PCI Express Completion Status (Step D)**
Data Transfer with Uncorrectable Data Error (Reads)	Successful (Poisoned TLP)
Data Transfer with Uncorrectable Data Error (Non-Posted Writes)	Unsupported Request
Master-Abort	Unsupported Request
Target-Abort	Completer Abort

Table 10-3 provides the translation a bridge has to do when it forwards a PCI Express Completion with an Error encoding to a PCI-X Split Completion Error Message.

Table 10-3: Error Forwarding Requirements (Step A to Step B of Figure 10-1) for PCI-X Split Terminations Originating from the PCI Express Interface

PCI Express Completion Status (Step A)	PCI-X Split Completion Message (Completer Error Code) (Step B)
Unsupported Request	Master-Abort
Completer Abort	Target-Abort
Others	Master-Abort

10.1.1. Received PCI Express Uncorrectable Packet Errors

10.1.1.1. Received Poisoned TLP

A bridge must determine if a TLP is poisoned before forwarding the associated data to the conventional PCI/PCI-X interface. When forwarding a poisoned TLP, the bridge must:

- Set the Detected Parity Error bit in the Status register.
- If the poisoned TLP is a read completion and the Parity Error Response bit in the Command register is set, set the Master Data Parity Error bit in the Status register.
- Generate a ERR_NONFATAL/ERR_FATAL message on PCI Express, if enabled as described in Section 10. The default severity is ERR_NONFATAL and is only selectable if Advanced Error Reporting is supported.
- If Advanced Error Reporting is supported, follow the Advanced Error Reporting rules for setting the appropriate status bit, logging the header and updating the PCI Express First Error Pointer (see Chapter 6 of PCI Express Base 1.0a).
- Invert the parity bit associated with each DWORD of data that is transferred if the secondary interface is operating in a mode that supports parity.
- If the secondary interface is operating in a mode that supports ECC, generate the appropriate (poisoned) ECC signature for each DWORD or QWORD transferred if the secondary interface is operating in a mode that supports ECC.
  - For 32-bit data transfers, use the PEL ECC signature.
  - For 64-bit data transfers, use the PED ECC signature.
- Set the Master Data Parity Error bit in the Secondary Status register if the Parity Error Response Enable bit in the Bridge Control register is set, and the bridge sees the PERR# pin asserted when forwarding a write request transaction with bad parity or ECC to the conventional PCI/PCI-X bus. If Advanced Error Reporting is supported, set the PERR# Assertion Detected bit in the Secondary Uncorrectable Error Status register, and log the header and update the Secondary Uncorrectable Error Pointer if both are enabled. No error message is generated

when PERR# is seen asserted by the bridge when forwarding a Poisoned TLP transaction from PCI Express to PCI/PCI-X with bad parity or ECC.

10.1.1.2. Received ECRC Error (Optional Support)

Bridges optionally check for ECRC errors while receiving a TLP from PCI Express with ECRC protection. The ability of a bridge to detect ECRC errors is reported via the Advanced Error Reporting capability.

When a bridge checks and detects an ECRC error, it must:

- Drop the transaction (i.e., not forward it to the conventional PCI/PCI-X interface). Since no positive determination can be made of whether the transaction is a request (posted or non-posted) or completion, the bridge must discard the packet.
- Set the ECRC Error Status bit in the Uncorrectable Error Status register of the Advanced Error Reporting capability, log the header, update the First Error Pointer field in the Advanced Error Capabilities and Control register, and generate an error message on PCI Express per the Advanced Error Reporting rules (specified in Chapter 7 of PCI Express Base 1.0a). The default error severity for ECRC errors is ERR_NONFATAL. Follow the PCI Express Base 1.0a rules for setting the Non-Fatal Error Detected or Fatal Error Detected bit in the Device Status register.
- Set the Detected Parity Error bit in the bridge's Status register.

10.1.2. Secondary Interface Uncorrectable Data Errors

The following sections describe the bridge requirements for error handling when forwarding a non-poisoned PCI Express transaction to conventional PCI/PCI-X and the bridge detects an uncorrectable data error.

10.1.2.1. Immediate Reads and Split Responses

If the bridge forwards a read request (I/O, Memory, or Configuration) from the PCI Express primary interface and detects an uncorrectable data error on the destination interface while receiving an immediate response or Split Response (PCI-X modes only) from the completer, the bridge:

- Sets both the Master Data Parity Error bit (if the Parity Error Response Enable bit is set in the Bridge Control register) and the Detected Parity Error bit in the Secondary Status register.
- Asserts PERR# (if the Parity Error Response Enable bit is set in the Bridge Control register) on the secondary interface.
- If Advanced Error Reporting is supported, set the Uncorrectable Data Error Status bit and follow the rules as noted in Section 10 with respect to advanced error logging.
- If Advanced Error Reporting is supported, generate ERR_NONFATAL/ERR_FATAL message on PCI Express if enabled via the Uncorrectable Data Error Mask bit in the Secondary Uncorrectable Error Mask register. The default error severity is ERR_NONFATAL.

- Refer to PCI-X PT 2.0a for additional error logging when the secondary interface is in ECC mode.

After detecting an uncorrectable data error on the destination bus for an immediate read transaction, the bridge continues to fetch data until the byte count is satisfied, or the target on the destination bus ends the transaction or Sequence in some other way. When the bridge creates the PCI Express completion, it forwards it with Successful Completion Status and poisons the TLP (see PCI Express Base 1.0a for more details on data poisoning). The uncorrectable read data error does not affect the bridge's behavior in any other way.

For PCI-X, an uncorrectable data error on a Split Response does not affect the handling of subsequent split completions.

10.1.2.2. Non-Posted Writes

If the bridge observes PERR# asserted on the conventional PCI/PCI-X secondary interface while forwarding a non-poisoned non-posted write transaction from PCI Express, the bridge sets the appropriate error status bits as described in PCI Bridge 1.2 and PCI-X PT 2.0a for that interface. If the target completes the transaction immediately with a data transfer, the bridge generates a PCI Express completion with Unsupported Request status to report the error to the requester. In addition, the bridge generates an ERR_FATAL/ERR_NONFATAL message on PCI Express, if enabled. The severity is selectable only when Advanced Error Reporting is supported. The default severity is ERR_NONFATAL.

If Advanced Error Reporting is supported, the error message can be masked from being generated via the PERR# Assertion Detected Mask bit in the Secondary Uncorrectable Error Mask register. Also, the bridge sets the PERR# Assertion Detected bit in the Secondary Uncorrectable Error Status register and follows the rules as noted in Section 10 with respect to advanced error logging.

If the target signals Split Response, the bridge terminates the transaction as it would for a Split Request that did not have an error and takes no further action. (When the Split Completion returns, the bridge generates a PCI Express completion with a Completion Status in accordance with the requirements listed in Section 10.2.4.1.1 and Table 10-6.)

10.1.2.3. Posted Writes

When a bridge forwards a non-poisoned posted write transaction from PCI Express to conventional PCI/PCI-X, detects PERR# asserted by the conventional PCI/PCI-X target, and the Parity Error Response Enable bit in the Bridge Control register is set, the bridge sets the Master Data Parity Error bit in the Secondary Status register. The bridge continues to forward the remainder of the transaction and generates an ERR_FATAL or ERR_NONFATAL message on PCI Express, if enabled. The error severity is selectable only if Advanced Error Reporting is supported and the default severity is ERR_NONFATAL.

If Advanced Error Reporting is supported, the error message can be masked from being generated via the PERR# Assertion Detected Mask bit in the Secondary Uncorrectable Error Mask register. Also, the bridge sets the PERR# Assertion Detected bit in the Secondary Uncorrectable Error Status register and follows the rules as noted in Section 10 with respect to advanced error logging.

10.1.2.4. PCI-X Split Read Completions

When a bridge forwards a non-poisoned read completion from PCI Express to PCI-X and detects PERR# asserted by the PCI-X target, the bridge continues to forward the remainder of the completion.

If Advanced Error Reporting is supported, the bridge will also generate an ERR_FATAL or ERR_NONFATAL message and the severity for this message is selectable. The message can be masked from being generated via the PERR# Assertion Detected Mask bit in the Secondary Uncorrectable Error Mask register. Also, the bridge sets the PERR# Assertion Detected bit in the Secondary Uncorrectable Error Status register and follows the rules as noted in Section 10 with respect to advanced error logging.

10.1.3. Secondary Interface Uncorrectable Address/Attribute Errors

Address or attribute parity/ECC errors detected by conventional PCI/PCI-X devices, when a bridge forwards transactions from PCI Express to conventional PCI/PCI-X, are reported via the SERR# pin on conventional PCI/PCI-X. When a bridge detects the SERR# pin asserted, it sets the appropriate status bits as described in PCI Bridge 1.2 and PCI-X PT 2.0a. SERR# assertions on conventional PCI/PCI-X are escalated to either a PCI Express ERR_FATAL or ERR_NONFATAL message if the SERR# Enable bit is set in the Bridge Control register or the SERR# Assertion Detected Mask bit is clear in the Secondary Uncorrectable Error Mask register (when the Advanced Error Reporting capability is supported). Also, the error severity is selectable only when the Advanced Error Reporting capability is supported and the default severity is ERR_FATAL.

When the Advanced Error Reporting capability is supported, the bridge also sets the SERR# Assertion Detected bit in the Secondary Uncorrectable Error Status register and follow the rules as noted in Section 10 with respect to advanced error logging. Note that there is no header log associated with reporting SERR# pin assertions on conventional PCI/PCI-X.

10.1.4. Received Master-Abort on the Secondary Interface

The following sections describe the requirements for how a bridge must handle Master-Abort received on the secondary interface in response to posted transactions, non-posted transactions, Split Completions, and Device ID Messages. The response behavior of a bridge on a received Master-Abort is independent of whether the aborted transaction is an exclusive access or not.

10.1.4.1. Master-Abort on a Posted Transaction

When forwarding a posted memory write transaction to conventional PCI/PCI-X, a bridge must respond to a transaction that receives Master-Abort on conventional PCI/PCI-X by throwing away the entire transaction. Additionally, the bridge must set the Received Master-Abort bit in the Secondary Status register. The bridge is enabled to escalate Master-Aborts received on posted writes to an ERR_NONFATAL/ERR_FATAL message on PCI Express if either the Master-Abort Mode

bit in the Bridge Control register is set or the Received Master-Abort Mask bit in the Secondary Uncorrectable Error Mask register (when AER is supported) is clear. Error severity is selectable only if Advanced Error Reporting is supported and the default severity is ERR_NONFATAL.

If Advanced Error Reporting is supported, a bridge also sets the Received Master-Abort Status bit in the Secondary Uncorrectable Error Status register and follows the rules as noted in Section 10 with respect to advanced error logging.

10.1.4.2. Master-Abort on a Non-Posted Transaction

A bridge that forwards a non-posted PCI Express Request to a conventional PCI or PCI-X secondary interface must respond to a Master-Abort received on that interface by returning a completion with the status of Unsupported Request on the PCI Express primary interface. Additionally, the bridge must set the Received Master-Abort bit in the Secondary Status register.

If Advanced Error Reporting is supported, a bridge also generates an ERR_NONFATAL/ERR_FATAL message on PCI Express. The default error severity is ERR_NONFATAL. The error message can be masked via the Received Master-Abort Mask bit in Secondary Uncorrectable Error Mask register. Also, a bridge sets the Received Master-Abort Status bit in the Secondary Uncorrectable Error Status register and follows the rules as noted in Section 10 with respect to advanced error logging.

10.1.4.3. Master-Abort on a Split Completion

PCI Express completions that are forwarded on the secondary interface as PCI-X Split Completions and encounter a Master-Abort also result in the bridge setting the Received Master-Abort bit in the Secondary Status register. Additionally, the bridge must set the Split Completion Discarded bit in the PCI-X Secondary Status register, discard the entire transaction, and transmit an ERR_NONFATAL/ERR_FATAL message on the PCI Express primary interface, if enabled. This response behavior is independent of the state of the Master-Abort Mode bit. Error severity is selectable only if Advanced Error Reporting is supported and the default severity is ERR_NONFATAL.

If Advanced Error Reporting is supported, the error message can be masked from being generated via the Master-Abort on Split Completion Mask bit in the Secondary Uncorrectable Error Mask register. Also, a bridge sets the Master-Abort on Split Completion Status bit in the Secondary Uncorrectable Error Status register and follows the rules as noted in Section 10 with respect to advanced error logging.

10.1.4.4. Master-Abort on a Device ID Message

If a bridge initiates a transaction on a PCI-X secondary interface using the Device ID Message command with the Silent Drop bit set and no device responds (i.e., Master-Abort response), the bridge takes no error action and discards the transaction; that is, the bridge does not set the Received Master-Abort bit in the Status register. Otherwise, the bridge handles Master-Aborts on Device ID Messages as it would for posted memory writes.

10.1.5. Received Target-Abort on the Secondary Interface

The following sections describe the requirements for how a bridge must handle Target-Abort received on the secondary interface in response to posted transactions, non-posted transactions, Split Completions, and Device ID Messages. The behavior of a bridge in response to Target-Abort received on posted and non-posted transactions is independent of whether the transaction is part of an exclusive access or not.

10.1.5.1. Target-Abort on a Posted Transaction

For posted requests that receive a Target-Abort on the secondary interface, the bridge responds by dropping the entire transaction, setting the Received Target-Abort bit in the Secondary Status register, and generating an ERR_NONFATAL/ERR_FATAL message on PCI Express, if enabled. The error severity is selectable only when Advanced Error Reporting is supported and the default severity is ERR_NONFATAL.

If Advanced Error Reporting is supported, the error message can be masked from being generated via the Received Target-Abort Mask bit in the Secondary Uncorrectable Error Mask register. Also, the bridge sets the Received Target-Abort Status bit in the Secondary Uncorrectable Error Status register and follows the rules as noted in Section 10 with respect to advanced error logging.

10.1.5.2. Target-Abort on a Non-Posted Transaction

A bridge that forwards a PCI Express non-posted Request to the secondary interface must respond to a Target-Abort received on that interface by returning a completion with the status of Completer Abort on the PCI Express primary interface. The bridge also sets the Received Target-Abort status bit in the Secondary Status register when it receives a Target-Abort response on the secondary interface and generates an ERR_NONFATAL/ERR_FATAL message on PCI Express, if enabled. The default error severity is ERR_NONFATAL and the severity is only selectable if Advanced Error Reporting is supported.

If Advanced Error Reporting is supported, the error message can be masked from being generated via the Received Target-Abort Mask bit in the Secondary Uncorrectable Error Mask register. Also, a bridge sets the Received Target-Abort Status bit in the Secondary Uncorrectable Error Status register and follows the rules as noted in Section 10 with respect to advanced error logging.

10.1.5.3. Target-Abort on a Split Completion

PCI Express completions that are forwarded on the secondary interface as PCI-X Split Completions and encounter a Target-Abort result in the bridge setting the Received Target-Abort bit in the Secondary Status register. Additionally, the bridge must set the Split Completion Discarded bit in the PCI-X Secondary Status register, discard the entire transaction, and transmit an ERR_NONFATAL/ERR_FATAL message on the PCI Express primary interface, if enabled. Note that error message severity is selectable only if Advanced Error Reporting is supported and the default severity is ERR_NONFATAL.

If Advanced Error Reporting is supported, the error message can be masked from being generated via the Target-Abort on Split Completion Mask bit in the Secondary Uncorrectable Error Mask register. Also, the bridge sets the Target-Abort on Split Completion Status bit in the Secondary Uncorrectable Error Status register and follows the rules as noted in Section 10 with respect to advanced error logging.

10.1.5.4. Target-Abort on a Device ID Message

If a bridge initiates a transaction on a PCI-X secondary interface using the Device ID Message command and the target (completer or bridge) signals Target-Abort, the initiator's actions are the same as for memory write transactions, independent of the state of the Silent Drop bit.

10.1.6. PCI Express Unsupported Request (UR) Completion Status

If the bridge receives a completion with Unsupported Request status on the PCI Express primary interface in response to any forwarded non-posted PCI-X transaction, the bridge sets the Received Master-Abort bit in the bridge's Status register and creates a Split Completion Message. The Split Completion Message is created as described in PCI-X PT 2.0a with the PCI-X Bridge Error class code and Master-Abort error message index. The Split Completion Message is created using information from the PCI Express Completion packet or using the information from the original PCI-X request when taking ownership of a PCI-X to PCI Express transaction. If the completion with Unsupported Request status is in response to a DWORD request, the error Split Completion Message replaces the normal Split Completion for this transaction. If the completion with Unsupported Request status is in response to a burst request, the bridge is permitted to send the error Split Completion Message in lieu of the first Split Completion for this Sequence or any continuation of the Sequence after a disconnection on an ADB. (Unlike conventional PCI, there is no way for PCI Express bridges to indicate on which data phase the Master-Abort occurred.)

10.1.7. PCI Express Completer Abort Completion Status

If the bridge receives a completion with Completer Abort status on the PCI Express primary interface in response to any forwarded non-posted PCI-X transaction, the bridge sets the Received Target-Abort bit in the bridge's Status register and creates a Split Completion Error Message. The Split Completion Message is created as described in PCI-X PT 2.0a with the PCI-X Bridge Error class code and Target-Abort error message index. The Split Completion Message is created using information from the PCI Express Completion packet or using the information from the original PCI-X request when the bridge has taken ownership of a PCI-X to PCI Express transaction. The bridge also set the Signaled Target-Abort bit in the Secondary Status register when it sends the Split Completion Message with the Target-Abort Status. If the completion with Completer Abort status is in response to a DWORD request, the error Split Completion Message replaces the normal Split Completion for this transaction. If the completion with Completer Abort status is in response to a burst request, the bridge is permitted to send the error Split Completion Message in lieu of the first Split Completion for this Sequence or any continuation of the Sequence after a disconnection on an

ADB. (Unlike conventional PCI, there is no way for PCI Express bridges to indicate on which data phase the Target-Abort occurred.)

10.1.8. Unexpected Completions

When a bridge receives an Unexpected Completion from PCI Express that matches the primary interface Requester ID, it is required to report the condition per the PCI Express Base 1.0a requirements and discard the Completion. In addition, bridges are also required to set the Unexpected Split Completion Status bit in the PCI-X Bridge Status register if the Split Completion is targeted at the bridge (i.e., the Requester ID carries the bridge's secondary interface Bus Number, and has a Device Number and Function Number set to zero, but the tag field does not match that of any transactions owned by the bridge).

10.2. Conventional PCI/PCI-X Originating Interface

This section describes the bridge error handling requirements for transactions that cross the bridge if the originating side of the bridge is operating in any of the conventional PCI/PCI-X modes and the destination interface is a PCI Express interface. The tables that follow summarize the error forwarding requirements (specific to transaction translation) described in the following sections. For the purpose of the error forwarding discussion, PCI-X Split Transactions (Split Completions or Split Completion Messages) are treated as originating from the PCI-X side of the bridge (see Figure 10-2) and are discussed in this section. PCI Express requests that result in immediate completion error cases on the conventional PCI/PCI-X interface are discussed in Section 10.1.2.

Bridges are required to forward Poisoned TLPs that are received on PCI Express, as described in Section 10.1.1.1. Bridges are further required to support TLP poisoning as a Transmitter to permit proper forwarding of parity and ECC errors that occur on the secondary interface.

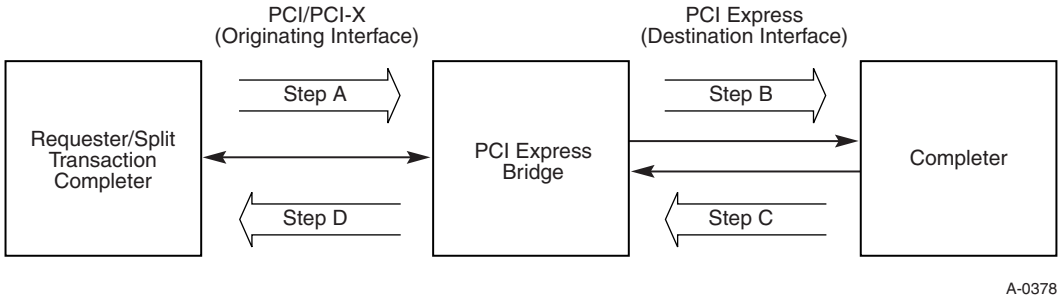

Figure 10-2: Transaction Error Forwarding Flow Diagram with conventional PCI/PCI-X as the Originating Interface

Posted and non-posted write and Split Completion data received on the secondary interface with bad parity or an uncorrectable ECC error are forwarded to PCI Express as Poisoned TLPs (i.e., the EP field is assigned a value of 1b). Split Completion Messages with uncorrectable data error during the data phase of the Split Completion message transaction result in CA status being returned on PCI Express.

Table 10-4 provides the error forwarding requirements for Uncorrectable data errors detected by a bridge when a transaction (request or a split completion) targets the PCI Express interface.

Table 10-4: Error Forwarding Requirements (Step A to Step B of Figure 10-2) for Received Conventional PCI/PCI-X Errors

Received Conventional PCI/PCI-X Error (Step A)		Forwarded PCI Express Error (Step B)
Conventional PCI/PCI-X Mode 1 (Parity)	**PCI-X Mode 1 (ECC) and Mode 2**	
Write with Uncorrectable Data Error	Write with Uncorrectable Data Error	Write Request with Poisoned TLP
Split Read Completion with Uncorrectable Data Error	Split Read Completion with Uncorrectable Data Error	Read Completion with Poisoned TLP
Split Completion Message with Uncorrectable Data Error in Data Phase	Split Completion Message with Uncorrectable Data Error in Data Phase	Read/Write Completion with CA Status

Table 10-5 provides the bridge behavior on a PCI delayed transaction that is forwarded by a bridge to PCI Express as a memory read request or an I/O read/write request and the PCI Express interface returns a completion with Unsupported Request or Completer Abort Completion Status.

Table 10-5: Error Forwarding Requirements (Step C to Step D of Figure 10-2) for PCI Express Completions that are Translated to a PCI Immediate Response

PCI Express Completion Status (Step C)	PCI Immediate Response (Step D)	
	MAM = 1	**MAM = 0**
Unsupported Request (on Memory Read, I/O Read)	Target-Abort	Normal, Return FFFF FFFFh
Unsupported Request (on I/O Write)	Target-Abort	Normal Completion
Completer Abort	Target-Abort	

Table 10-6 provides the bridge error forwarding requirements when receiving a PCI-X Split Completion Message indicating a PCI-X error condition.

Table 10-6: Error Forwarding Requirements (Step A to Step B of Figure 10-2) for PCI-X Split Completion Messages Originating from the PCI-X Interface

PCI-X Split Completion Message (Completer Error Code) (Step A)	Completer Error Code		PCI Express Completion Status (Step B)
	Class	Index	
Normal Completion	0h	00h	Successful Completion
Master-Abort	1h	00h	Unsupported Request
Target-Abort	1h	01h	Completer Abort
Uncorrectable Write Data Error	1h	02h	Unsupported Request
Byte Count Out of Range	2h	00h	Unsupported Request
Uncorrectable Split Write Data Error	2h	01h	Unsupported Request
Device-Specific Error	2h	8Xh	Completer Abort
Reserved/Illegal	Others		Completer Abort

10.2.1. Received Conventional PCI/PCI-X Errors

10.2.1.1. Uncorrectable Data Error on a Non-Posted/Posted Write

If a PCI Express bridge receives a posted or non-posted write transaction on its secondary interface that is addressed such that it would cross the bridge and the bridge detects an uncorrectable data error, the bridge sets the Detected Parity Error bit in the Secondary Status register. The bridge also asserts PERR# on the secondary interface, if enabled. Refer to PCI-X PT 2.0a for additional error logging requirements when the secondary interface is operating in ECC mode.

For a posted write, the bridge forwards the transaction to PCI Express as a request with a poisoned TLP (see Chapter 2 of PCI Express Base 1.0a for details).

For a non-posted write, if the secondary interface is operating in conventional PCI mode and the Parity Error Response Enable bit is set in the Bridge Control register, the bridge discards the transaction. No Delayed Write Request is enqueued. Refer to PCI Bridge 1.2 for additional details.

For a non-posted write, if the secondary interface is operating in PCI-X mode, the bridge optionally signals either Data Transfer or Split Response as described in PCI-X PT 2.0a. If the bridge signals Data Transfer, the bridge discards the transaction and does not forward it. If the bridge signals Split Response, the bridge forwards the transaction to PCI Express as a request with a poisoned TLP. The bridge must not signal Retry or Target-Abort solely because of an uncorrectable data error.

If Advanced Error Reporting is supported, bridge also sets the Uncorrectable Data Error Status bit in the Secondary Uncorrectable Error Status register and follows the rules in PCI Express Base 1.0a with respect to error logging and message generation. The default error severity is ERR_NONFATAL.

IMPLEMENTATION NOTE

Discarding or Forwarding a Non-Posted Write with an Uncorrectable Data Error

If the bridge's secondary interface responds to a non-posted write transaction with Data Transfer and the write contains an uncorrectable data error, the bridge is allowed to discard the transaction (this is consistent with conventional PCI behavior where non-posted write transactions with data parity errors are discarded). A bridge may alternatively respond to a non-posted write transaction with Split Response. If the bridge signals Split Response prior to sampling and checking write data parity or ECC, it is able to respond to all non-posted write transactions sooner, but it is required to forward the transaction even if a data error is encountered. Therefore, if discarding of the data on a data error is the desired behavior, the bridge must verify good data parity or ECC prior to signaling Split Response.

10.2.1.2. Uncorrectable Data Error on Split Read Completion Data

If a bridge detects an uncorrectable data parity/ECC error on the PCI-X bus for a split read completion that crosses the bridge, the bridge asserts PERR# if enabled and sets the Detected Parity Error bit in the Secondary Status register and the Master Data Parity Error bit in the Secondary Status register if the Parity Error Response Enable bit is set. The bridge forwards the read completion to PCI Express as poisoned TLP (irrespective of whether of PERR# was signaled).

If Advanced Error Reporting is supported, the bridge generates an ERR_FATAL or ERR_NONFATAL message (with selectable severity) on PCI Express if enabled via the Uncorrectable Data Error Mask bit in the Secondary Uncorrectable Error Mask register. Also, the bridge sets the Uncorrectable Data Error Status bit in the Secondary Uncorrectable Error Status register and follows the rules as noted in Section 10 with respect to advanced error logging.

10.2.1.3. Uncorrectable Data Error on PCI Delayed Read Completions

When a bridge forwards a non-poisoned read completion from PCI Express to PCI and detects PERR# asserted by the PCI master, the bridge continues with forwarding the remainder of the completion.

When supporting Advanced Error Reporting, a bridge generates an ERR_FATAL or ERR_NONFATAL message (with selectable severity) on PCI Express if enabled via the PERR# Assertion Mask bit in the Secondary Uncorrectable Error Mask register. Also, a bridge must set the PERR# Assertion Detected bit in the Secondary Uncorrectable Error Status register and follow the rules as noted in Section 10 with respect to advanced error logging.

When a bridge forwards a poisoned read completion from PCI Express to PCI (received via a poisoned TLP from PCI Express), the bridge proceeds with the above mentioned actions when it

detects the PERR# pin asserted by the PCI master, but no error message is generated on PCI Express.

10.2.1.4. Uncorrectable Address and Attribute Errors

Bridge behavior on uncorrectable address and attribute parity/ECC errors that are detected is similar to the bridge behavior described in PCI Bridge 1.2 and PCI-X PT 2.0a for PCI/PCI-X bridges. When bridges are enabled to detect parity/ECC errors (via the Parity Error Response Enable bit in the Bridge Control register) and an address or attribute parity/ECC error is detected by the bridge, it terminates the transaction as specified in PCI Bridge 1.2 and PCI-X PT 2.0a and generates an ERR_FATAL or ERR_NONFATAL message on PCI Express (if enabled). The error severity is selectable only if Advanced Error Reporting is supported and the default severity is ERR_FATAL.

If Advanced Error Reporting is supported, the error generation can be masked using the appropriate mask bit (i.e., either the Uncorrectable Address Error Mask bit or the Uncorrectable Attribute Error Mask bit) in the Secondary Uncorrectable Error Mask register. Also, the bridge must set the appropriated status bit (i.e., either the Uncorrectable Address Error Status bit or the Attribute Error Status bit) in the Secondary Uncorrectable Error Status register and follow the rules as noted in Section 10 with respect to advanced error logging.

Refer to PCI-X PT 2.0a for additional error logging required in the PCI-X Capability registers when the secondary interface is operating in ECC mode.

10.2.1.5. Correctable Errors

Correctable errors detected by a bridge when the secondary interface is operating in ECC mode are logged per PCI-X PT 2.0a. These correctable errors also cause the Correctable Error Detected Status bit in the PCI Express Device Status register to be set and an ERR_COR message to be generated on PCI Express if enabled.

10.2.2. Unsupported Request (UR) Completion Status

A PCI Express bridge provides two methods for handling a PCI Express completion received with Unsupported Request status in response to a request originated by a secondary interface in conventional PCI mode. The bridge's response to this completion is controlled by the Master-Abort Mode bit in the Bridge Control register.

In the default case, the Master-Abort Mode bit is cleared and an Unsupported Request is not considered to be an error. However, when the Master-Abort Mode bit is set, it is considered an error condition when the bridge receives a completion with Unsupported Request status in response to a conventional PCI request.

In all cases of receiving Unsupported Request completion status on PCI Express in response to a conventional PCI request initiated on the secondary interface, the bridge must set the Received Master-Abort bit in the Status register.

When the Master-Abort Mode bit is cleared, the bridge will operate in a PCI compatibility mode. When a read transaction initiated on an interface operating in conventional PCI mode results in the return of a completion with Unsupported Request status, the bridge will return FFFF FFFFh to the originating master and terminate the read transaction on the originating interface normally (by asserting TRDY#). When a non-posted write transaction results in a completion with Unsupported Request status, the bridge will complete the write transaction on the originating bus normally (by asserting TRDY#) and discard the write data.

When the Master-Abort Mode bit is set, the bridge must signal a Target-Abort to the originating master of an upstream read or a non-posted write transaction when the corresponding request on the PCI Express primary interface results in a completion with Unsupported Request status. Additionally, the bridge must set the Signaled Target-Abort bit in the Secondary Status register.

10.2.3. Completer Abort Completion Status

If the bridge receives a completion with Completer Abort status on the PCI Express primary interface in response to any forwarded non-posted PCI transaction, the bridge sets the Received Target-Abort bit in the bridge's Status register.

If the Secondary interface is operating in conventional PCI mode, a Completer Abort response on PCI Express translates to a Delayed Transaction Target-Abort. Bridges are required to continue forwarding completion data received on PCI Express for a particular PCI request to the requesting PCI agent in address order and up to (but not including) the first address associated with the PCI Express packet containing Completer Abort Completion Status. The bridge must then signal a Target-Abort. Bridges must set the Signaled Target-Abort bit in the Secondary Status register when signaling a Target-Abort to a PCI agent.

10.2.4. Split Completion Errors

10.2.4.1. Split Completion Message with Completer Errors

A transaction originating from the PCI Express primary interface of the bridge and requiring a completion may be forwarded to a PCI-X secondary interface where the target (or completer) responds with Split Response. After signaling Split Response, if the completer encounters an abnormal condition that prevents it from executing a Split Transaction, the completer must notify the requester of the abnormal condition by sending a Split Completion Message with the Completer Error class (see PCI-X PT 2.0a for details on valid completer responses). A PCI Express to PCI-X bridge translates a PCI-X Split Completion Message Status to a PCI Express Completion Status per Table 10-6 and when doing so sets the appropriate conventional PCI status bits associated with the secondary interface as described in PCI-X PT 2.0a. Target-Abort, Uncorrectable Write Data Error, and Device-Specific Errors also cause an ERR_FATAL/ERR_NONFATAL message to be generated on PCI Express, if enabled. The severity is selectable when supporting Advanced Error Reporting and the default severity is ERR_NONFATAL. Additionally, when a bridge responds with Completer Abort status, it sets the Signaled Target-Abort Status bit in the bridge's Status register.

If Advanced Error Reporting is supported, the error generation can be masked using the appropriate mask bit (i.e., the Received Target-Abort Mask bit, the Received Master-Abort Mask bit, or the PERR# Assertion Detected Mask bit) in the Secondary Uncorrectable Error Mask register. Also, a bridge that receives PCI-X Split Completion Message with Master-Abort, Target-Abort, or Uncorrectable Data Error status must set the Received Master-Abort, Received Target-Abort, and PERR# Assertion Detected Status bits in the Secondary Uncorrectable Error Status register, respectively, and follow the rules as noted in Section 10 with respect to advanced error logging.

10.2.4.1.1. Other Split Completion Errors

10.2.4.1.1.1. Corrupted or Unexpected Split Completions

Bridges are required to handle corrupted or unexpected Split Completions (as defined in Chapter 5 of PCI-X PT 2.0a) received on the secondary interface that include a Requester ID matching one used by the bridge for transactions that it generates or takes ownership of. When a corrupted or unexpected Split Completion is detected, the bridge is required to set the Unexpected Split Completion Status bit in the PCI-X Secondary Status register.

When Advanced Error Reporting is supported, the bridge generates an ERR_FATAL or ERR_NONFATAL message (with selectable severity) on PCI Express if enabled via the Unexpected Split Completion Mask bit in the Secondary Uncorrectable Error Mask register. Also, a bridge must set the Unexpected Split Completion Status bit in the Secondary Uncorrectable Error Status register and follow the rules as noted in Section 10 with respect to advanced error logging.

10.2.4.1.1.2. Data Parity Errors on Split Completion Messages

Data parity/ECC errors are also possible during the data phase of a Split Completion Message. Bridges return a PCI Express Completion with Completer Abort Completion Status, under this error condition, set the appropriate status bits per PCI-X PT 2.0a and generate an ERR_FATAL or ERR_NONFATAL message on PCI Express, if enabled. The error severity is selectable only when Advanced Error Reporting is supported and the default severity is ERR_FATAL.

When Advanced Error Reporting is supported, the error generation can be masked using the Uncorrectable Split Completion Message Data Error Mask bit in the Secondary Uncorrectable Error Mask register. Also, a bridge must set the Uncorrectable Split Completion Message Data Error Status bit in the Secondary Uncorrectable Error Status register and follow the rules as noted in Section 10 with respect to advanced error logging.

10.3. Timeout Errors

10.3.1. PCI Express Completion Timeout Errors

The PCI Express Completion Timeout Mechanism allows requesters to abort a non-posted request if completion does not arrive within a reasonable time. This mechanism allows for recovery of the requester in a scenario where there is no reasonable expectation that the completion will arrive. When a requester times out on a request, it retires the request, deallocates any buffer space corresponding to that request, and reports an error.

Bridges, when acting as initiators on PCI Express on behalf of internally-generated requests and requests forwarded from a secondary interface in conventional PCI mode, behave as Endpoints for requests that they take ownership of (i.e., bridges allocate header and data buffer space for requests that they own). These requests have the same chance of being lost in the PCI Express fabric as any other requests and the completion timeout mechanism applies to bridge requests as well under such a scenario. Bridges, when transparently forwarding requests from PCI-X to PCI Express (i.e., without taking ownership), do not behave as Endpoints and hence do not have the ability to retire a transaction autonomously. The completion timeout mechanism does not apply to such requests.

When bridges timeout on requests that they own, they treat the request as if it received an unsupported request completion from the PCI Express fabric and follow the rules for handling unsupported request completions as described in Section 10.2.2 and generate an error message on PCI Express if enabled (default severity of ERR_NONFATAL). Note that when a bridge forms the Split Completion Message on PCI-X for a Completion Timeout Error, it uses its primary interface Bus Number, Device Number, and Function Number for the Completer ID field on PCI-X. Also, if a bridge supports Advanced Error Reporting, it must set the appropriate AER status bit (i.e., Completion Timeout Status) and follow the rules as noted in Section 10 with respect to advanced error logging.

10.3.2. PCI Delayed Transaction Timeout Errors

PCI Express bridges are required to implement Delayed Transaction Timers for their secondary interface as PCI bridges do. The control and operation of the Delayed Transaction Timer is the same as for PCI bridges. In addition, bridges that implement Advanced Error Reporting log delayed transaction timer errors in the Delayed Transaction Discard Timer Expired Status bit in the Secondary Uncorrectable Error Status register. There is no header log associated with Delayed transaction timeout errors (software must ignore the header log) and error escalation on PCI Express is enabled either if the Discard Timer SERR# Enable bit in the Bridge Control register is set or the Delayed Transaction Discard Timer Expired Mask bit in the Secondary Uncorrectable Error Mask register is clear (when supported). The default severity for Delayed Transaction Timeout errors is ERR_NONFATAL and the severity is selectable if Advanced Error Reporting is supported.

10.4. Other Errors

10.4.1. SERR# Assertion Detected

PCI devices can assert SERR# when detecting errors that could compromise system integrity. Bridges are required to detect and log SERR# assertions in the conventional PCI registers as described in PCI Bridge 1.2. SERR# assertions on conventional PCI/PCI-X are enabled to be forwarded as PCI Express ERR_FATAL or ERR_NONFATAL messages, either if the SERR# Enable bit in the Bridge Control register is set or the SERR# Assertion Detected Mask bit in Advanced Error Reporting capability is clear (when supporting Advanced Error Reporting). Error severity is selectable only when supporting Advanced Error Reporting and the default severity is ERR_FATAL. Bridges that support Advanced Error Reporting are also required to log the detection of SERR# assertions in the SERR# Assertion Detected status bit in the Secondary Uncorrectable Error Status register and update the Secondary Uncorrectable Error Pointer field per rules referred to in Section 10. There is no header log associated with detection of SERR# assertions.

10.4.2. Internal Bridge Errors

Internal bridge errors that could compromise system integrity must be reported by bridges via an ERR_FATAL message on PCI Express, if enabled. The message severity is selectable if Advanced Error Reporting is supported.

If Advanced Error Reporting is supported, a bridge must also log this condition by setting the Internal Bridge Error Status bit and updating the Secondary Uncorrectable Error Pointer per rules referred to in Section 10. There is no header log associated with internal bridge errors.

11

11. Legacy Support

Bridges are required to support the ISA addressing mode and may optionally support VGA addressing for interoperability with legacy system software (or firmware) and with legacy devices that may reside downstream of the bridge. These features are reviewed in this chapter.

11.1. VGA Addressing (Optional)

The VGA Enable bit in the Bridge Control register (Section 5.1.2.13) is used to control response by the bridge to both VGA frame buffer addresses and to VGA register addresses. If the VGA Enable bit is 1b, the bridge will positively decode and forward memory accesses to VGA frame buffer addresses and I/O accesses to VGA registers from the primary interface to the secondary interface (and block forwarding from the secondary to primary interface of these same accesses). A bridge never forwards transactions that access VGA BIOS memory addresses (regardless of the setting of the VGA Enable bit). ROM code provided by PCI-compatible devices must be copied to system memory before execution and may be mapped to any address in PCI memory address space via the Expansion ROM Base Address register in the device's configuration header.

The VGA 16-Bit Decode bit in the Bridge Control register is used to select between 10-bit and 16-bit VGA I/O address decoding and is applicable when the VGA Enable bit is 1b.

For reference, the VGA addresses governed by the VGA Enable function are listed below.

VGA memory addresses:

- 0A 0000h through 0B FFFFh

VGA I/O addresses (Address bits 15:10 are not decoded when the VGA 16-Bit Decode bit is 0b):

- Address bits 9:0 = 3B0h through 3BBh and 3C0h through 3DFh (VGA 16-Bit Decode bit is 0b)
- Address bits 15:0 = 03B0h through 03BBh and 03C0h through 03DFh (VGA 16-bit Decode bit is 1b)

The VGA Palette Snoop Enable bit must be implemented as read-only with a value of zero for PCI Express bridges. A PCI Express bridge positively decodes palette reads and writes initiated from the PCI Express interface when the VGA Enable bit is set. When the VGA Enable bit is cleared, all VGA palette accesses are ignored.

Refer to Chapter 4 of PCI Bridge 1.2 for additional details.

11.2. ISA Addressing Mode

PCI Express bridges must implement the ISA Enable bit in the Bridge Control register and support the associated functionality.

The ISA Enable affects only I/O addresses that are in the bridge's I/O range (as defined by the I/O Base, I/O Base Upper 16 Bits, I/O Limit, and I/O Limit Upper 16 Bits) and in the first 64 KB of PCI I/O Space (0000 0000h to 0000 FFFFh). If this bit is 1b and the I/O address meets the stated constraints, the bridge will block forwarding of I/O transactions downstream (from the primary interface to the secondary interface) if the I/O address is in the top 768 bytes of each naturally aligned 1-KB block. If the ISA Enable bit is 0b, the bridge must forward downstream all I/O addresses in the address range defined by the I/O Base and I/O Limit registers.

If the ISA Enable bit is 1b, I/O transactions on the secondary bus in the top 768 bytes of any 1-KB address block within the first 64 KB of PCI I/O Space will be forwarded upstream to the primary bus, even if the address is between the I/O base and I/O limit addresses. Figure 11-1 illustrates this mapping for a 4-KB range. The combination of the 4-KB granularity and the ISA Enable bit results in I/O address decoding on the secondary interface of the bridge that is similar to EISA slot decoding. Devices on the secondary interface are permitted to be mapped to the first 256 bytes of each naturally aligned 1-KB block within the defined I/O address range.

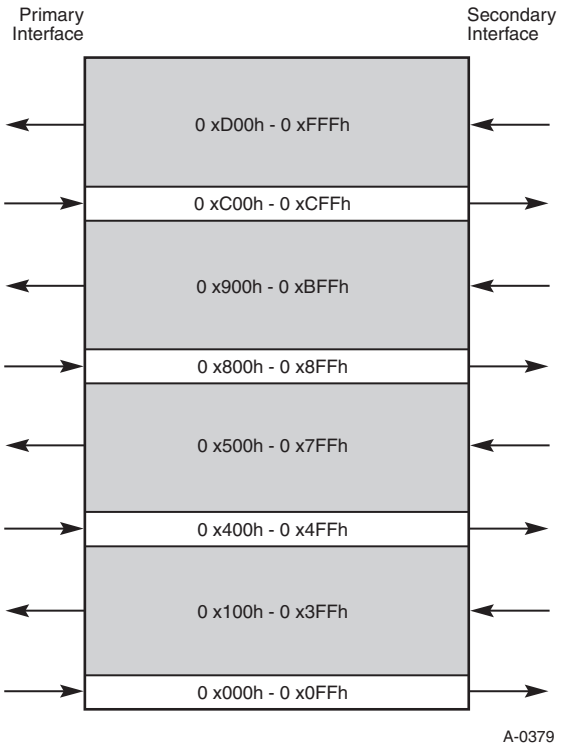

Figure 11-1: ISA Mode I/O Address Range Example

The ISA Enable bit only affects the I/O address decoding behavior of the bridge. It does not affect the bridge's prefetching, posting, ordering, or error handling behavior.

A

A. Conventional PCI/PCI-X to PCI Express Bridges (Reverse Bridges)

This appendix provides guidelines for implementing a bridge from a conventional PCI or PCI-X primary interface to a PCI Express secondary interface. This primary and secondary interface configuration is referred to as a reverse bridge. The guidelines in this appendix focus on the implementation of an integrated solution such as a PCI add-in card or system board that integrates a PCI Express device.

An example of the use of a reverse bridge as the interface device on a conventional PCI/PCI-X add-in card is shown in Figure A-1.

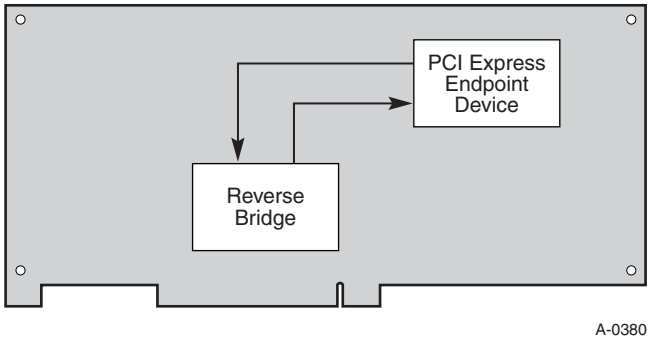

Figure A-1: Conventional PCI/PCI-X Add-In Card with Reverse Bridge Interface Device

A.1. Features Outside the Scope of this Appendix

The reverse bridge guidelines presented in this appendix are limited to the set of important recommendations that need to be followed in order to support a single PCI Express device on the downstream port of the reverse bridge. Because of the limited nature of these guidelines, the following PCI Express features are outside the scope of this appendix:

- End-to-End CRC
- Virtual Channels
- Isochronous Traffic
- PCI Express Hot Plug in-band signaling
- Extended Configuration Space (except in the case of PCI-X Mode 2)
- Beacon signaling mechanism
- Vendor-Defined Messages

In addition, from the system topology and usage model point of view, the following configurations are outside the scope of this appendix:

- Multiple PCI Express Links fanned out from the single bridge
- Bridging from PCI Express back to conventional PCI/PCI-X
- PCI Express standard add-in card/module expansion slots connected downstream of reverse bridge

A.2. Initialization Requirements

A.2.1. Link Initialization

A reverse bridge should adhere to the Link initialization requirements of a root port.

A.2.2. Slot Power Initialization

A reverse bridge should generate a Set Slot Power Limit message from its downstream port indicating the power limitations of the implementation. For example, a PCI add-in card should generate a Set Slot Power Limit message that sets the power limit such that the card does not exceed the power limits for a PCI add-in card. The power limit set for a PCI add-in card solution should be based on the value encoded on the PRSNT1# and PRSNT2# pins.

A.3. Configuration Transaction Support

A reverse bridge has the same requirements as a Root Complex or Switch (see Chapter 7 of PCI Express Base 1.0a) in terms of preventing Type 0 configuration transactions that do not specify Device Number 0 from being propagated to the PCI Express Link. The bridge should Master-Abort Type 1 configuration transactions that would be converted into a Type 0 configuration transaction not specifying Device Number 0.

A.4. Reset Requirements

A reverse bridge must generate a PCI Express hot reset on the PCI Express Link when RST# is asserted on the primary bus or when the Secondary Bus Reset bit is set in the Bridge Control register. It is recommended that the bridge generate a PCI Express hot reset when the bridge transitions from $D3_{hot}$ to D0 to mimic the PCI $D3_{hot}$ to D0 reset. It is also recommended that reverse bridges use RST# from the PCI bus to derive corresponding side-band reset signal for the PCI Express device.

A.5. Message Handling Requirements

A.5.1. Power Management Message Support

A reverse bridge should generate a PME Turn Off message when the bridge is placed in D3. The bridge should then wait for the PME Turn Off Acknowledge before entering D3. After entering

D3, the reverse bridge must wait for the downstream device to initiate the transition of the Link State to L2/L3. A reverse bridge has the same requirements for entering L2/L3 as the downstream port of a PCI Express switch.

A.5.1.1. PME Handling Requirements

A reverse bridge must translate PME messages from the PCI Express interface to the PME# signal on the conventional PCI/PCI-X bus. A reverse bridge must convert the edge triggered PME events on the PCI express interface to the level triggered PME# signal on PCI. A reverse bridge should assert PME# on the PCI bus when PCI Express WAKE# signal is asserted as well as when PCI Express PME Message is received.

For compatibility with existing software, the bridge should not signal PME unless the PME signaling is enabled by the PME_En bit in the Power Management Control/Status register (see PCI PM 1.1). The bridge should set its PME_Status bit when PME# is asserted and deassert PME# when the PME_Status bit or the PME_En bit is cleared. All PME messages received while the PME_En bit is cleared should be ignored and the PME_Status bit must not be set during this time.

A.5.2. Locked Transaction Support

A reverse bridge, like a PCI-to-PCI bridge, is allowed to pass locked transactions from the primary interface to the secondary interface. If a reverse bridge supports the PCI Lock Mechanism (supports LOCK# input signal), the bridge must use the Memory Read Locked command to initiate a locked sequence when a locked request is sent on the PCI bus. All subsequent locked read transactions targeting the bridge will use the Memory Read Locked command on PCI Express. All subsequent locked write transactions will use the Memory Write command on PCI Express. The bridge must send the Unlock message when the PCI Lock sequence is complete (see PCI 3.0, Appendix F).

A.5.3. Error Signaling Message Support

A reverse bridge is required to convert all Error Fatal and Error Non-Fatal messages to SERR# on the PCI interface.

A.5.4. INTx Interrupt Message Support

A reverse bridge should control the state of the corresponding PCI INTx# interrupt pins based on the Assert_INTx and Deassert_INTx messages the bridge receives.

A.6. Reverse Bridge Effects on the Configuration Header

As a baseline, PCI Express reverse bridges use the standard Type 01h Configuration Space header as defined in PCI Bridge 1.2. However, the usage of the following registers/bits is modified relative to their definition in PCI Bridge 1.2:

- Secondary Latency Timer register: This register is not used in PCI Express and is implemented as read only zero.
- Secondary Status register:
  - 66MHz Capable: This bit is not used in PCI Express and is implemented as read only zero.
  - Fast Back-to-Back Transactions Capable: This bit is not used in PCI Express and is implemented as read only zero.
  - DEVSEL Timing: This field is not used in PCI Express and is implemented as read only zero.
- Bridge Control register:
  - Fast Back-to-Back Enable: This bit is not used in PCI Express and is implemented as read only zero.
  - Primary Discard Timer: This bit selects the number of PCI clocks that the bridge will wait for a master on the primary interface to repeat a Delayed Transaction request. This bit is ignored in PCI-X mode.
  - Secondary Discard Timer: This bit is not used in PCI Express and is implemented as read only zero.

The usage of the following registers/bits is modified relative to a PCI Express bridge:

- PCI Express Capabilities register:
  - Device/Port Type: This field is encoded as 8h indicating this is a PCI/PCI-X to PCI Express Bridge.
- PCI-X Bridge ECC Control and Status register:
  - Select Secondary ECC Register: This bit is hardwired to a 0. All ECC Control and Status registers are associated with the primary interface.

A.7. PCI-X Support

PCI/PCI-X to PCI Express bridges that support PCI-X must include a PCI-X Capabilities List item as shown in Figure 5-3. If the bridge is installed on an add-in card, the connection of the PCIXCAP pin of the add-in card must be consistent with the presence of this Capabilities List item in the bridge that serves as the add-in card's system interface (i.e., the most upstream bridge). That is, the connection of the PCIXCAP pin must indicate the add-in card is capable of operating in PCI-X mode if, and only if, the PCI-X Capabilities List item is present in the bridge that connects to the PCI connector of the add-in card. Refer to PCI-X 2.0a for more information on PCIXCAP and PCI-X Capabilities List item requirements for non-bridge functions of multi-function devices.

Acknowledgements

The following persons were instrumental in the development of the *PCI Express to PCI/PCI-X Bridge Specification*:[19]

Jasmin Ajanovic	Intel Corporation	Isaac Livny	Agere Systems
Katsutoshi Akagi	NEC Corporation	Andrew Lueck	Texas Instruments Incorporated
Leonard Brugger	PLX Technology, Inc.	Fred Meschino	Dell Computer Corporation
Jung-Lin Chang	Sun Microsystems, Inc.	Sridhar Muthrasanallur	Intel Corporation
Weilong Chen	VIA Technologies, Inc.	Richard Reeves	Advanced Micro Devices, Inc.
Joe Cowan	Hewlett-Packard Company	Jack Regula	PLX Technology, Inc.
Ben Drerup	IBM Corporation	Fadi Saibi	Agere Systems
Nobuo Furuya	NEC Corporation	Prashant Sethi	Intel Corporation
Steve Glaser	Sun Microsystems, Inc.	Wesley Shao	Sun Microsystems, Inc.
Nils Graef	Agere Systems	Ron Simmons	PLX Technology, Inc.
Gabriel Higham	Dell Computer Corporation	Arie van der Hoeven	Microsoft Corporation
Paul Howard	Texas Instruments Incorporated	Davis Walker	Microsoft Corporation
Carl Jackson	Hewlett-Packard Company	Ted Willke	Intel Corporation
Mohan Kumar	Intel Corporation	Julie Wong	NVidia Corporation

[19] Company affiliation listed is at the time of specification contributions.