PCI Express™
Base Specification
Revision 1.0a

April 15, 2003

Revision	Revision History	DATE
1.0	Initial release.	7/22/02
1.0a	Incorporated Errata C1-C67 and E1-E4.17.	04/15/03

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 this specification.

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

DISCLAIMER

Contents

Figures

Tables

Objective of the Specification

This specification describes the PCI Express architecture, interconnect attributes, fabric management, and the programming interface required to design and build systems and peripherals that are compliant with the PCI Express specification.

5   The goal is to enable such devices from different vendors to inter-operate in an open architecture. The specification is intended as an enhancement to the PCI architecture spanning multiple market segments; Clients (Desktops and Mobile), Servers (Standard and Enterprise), and Embedded and Communication devices. The specification allows system OEMs and peripheral developers adequate room for product versatility and market differentiation without the burden of carrying
10   obsolete interfaces or losing compatibility.

Document Organization

The PCI Express specification is organized as a base specification and a set of companion documents. At this time, the *PCI Express Base Specification* and the *PCI Express Card Electromechanical Specification* are being published. As the PCI Express definition evolves, other companion
15   documents will be published.

The *PCI Express Base Specification* contains the technical details of the architecture, protocol, Link Layer, Physical Layer, and software interface. The PCI Express Base Specification is applicable to all variants of PCI Express.

The *PCI Express Card Electromechanical Specification* focuses on information necessary to implementing
20   an evolutionary strategy with the current PCI desktop/server mechanicals as well as electricals. The mechanical chapters of the specification contain a definition of evolutionary PCI Express card edge connectors while the electrical chapters cover auxiliary signals, power delivery, and the add-in card interconnect electrical budget.

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

8b/10b	The data encoding scheme[1] used in the PCI Express Version 1.0 Physical Layer.
advertise (Credits)	Used in the context of Flow Control, the act of a Receiver sending information regarding its Flow Control Credit availability.
asserted	The active logical state of a conceptual or actual signal.
attribute	Transaction handling preferences indicated by specified Packet header bits and fields (for example, non-snoop).
Beacon	An optional 30 kHz–500 MHz in-band signal used to exit the L2 Link power management state. One of two defined mechanisms for waking up a Link in L2 (see also wakeup).
Bridge	A device that virtually or actually connects a PCI/PCI-X segment or PCI Express Port with an internal component interconnect or with another PCI/PCI-X segment or PCI Express Port. A virtual *Bridge* in a Root Complex or Switch must use the software configuration interface described in this specification.
by-1, x1	A Link or Port with one Physical Lane.
by-8, x8	A Link or Port with eight Physical Lanes.
by-N, xN	A Link with "N" Physical Lanes.
Character	An 8-bit quantity treated as an atomic entity; a byte.
cold reset	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.
Configuration Space	One of the four address spaces within the PCI Express architecture. Packets with a *Configuration Space* address are used to configure a device.
conventional PCI	Behavior or features that conform to the *PCI Local Bus Specification, Rev. 2.3*.

---

[1] IBM Journal of Research and Development, Vol. 27, #5, September 1983 "A DC-Balanced, Partitioned-Block 8B/10B Transmission Code" by Widmer and Franaszek.

component	A physical device (a single package).
Data Link Layer	The intermediate Layer that is between the Transaction Layer and the Physical Layer.
Data Link Layer Packet, DLLP	A Packet generated in the Data Link Layer to support Link management functions.
data payload	Information following the header in some packets that is destined for consumption by the logical device receiving the Packet (for example, Write Requests or Read Completions).
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.
Downstream	1. The relative position of an interconnect/system element (Port/component) that is farther from the Root Complex. The Ports on a Switch that are not the Upstream Port are *Downstream* Ports. All Ports on a Root Complex are *Downstream* Ports. The *Downstream* component on a Link is the component farther from the Root Complex. 2. A direction of information flow where the information is flowing away from the Root Complex.
DFT	Design for Testability.
DWORD, DW	Four bytes. Used in the context of a data payload, the 4 bytes of data must be on a naturally aligned four-byte boundary (the least significant two bits of the byte address are 00b).
Egress Port	The transmitting Port; that is, the Port that sends outgoing traffic.
Electrical Idle	The state of the output driver in which both lines, D+ and D-, are driven to the DC common mode voltage.
Endpoint	A device with a Type 00h Configuration Space header.
Error Recovery, Error Detection	The mechanisms that ensure integrity of data transfer, including the management of the transmit side retry buffer(s).
Flow Control	The method for communicating 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 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.
in-band signaling	A method for signaling events and conditions using the Link between two components, as opposed to the use of separate physical (sideband) signals. All mechanisms defined in this document can be implemented using *in-band signaling*, although in some form factors sideband signaling may be used instead.
Ingress Port	Receiving Port; that is, the Port that accepts incoming traffic.
I/O Space	One of the four address spaces of the PCI Express architecture. Identical to the I/O space defined in PCI 2.3.
isochronous	Data associated with time-sensitive applications, such as audio or video applications.
invariant	A field of a TLP header that contains a value that cannot legally be modified as the TLP flows through the PCI Express fabric.
Lane	A set of differential signal pairs, one pair for transmission and one pair for reception. A by-N Link is composed of N *Lanes*.
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*.
Logical Idle	A period of one or more Symbol Times when no information (TLPs, DLLPs, or any special Symbol) is being transmitted or received. Unlike Electrical Idle, during *Logical Idle* the idle character is being transmitted and received.

Malformed Packet	A TLP that violates specific TLP formation rules as defined in this specification.
Message	A TLP used to communicate information outside of the Memory, I/O, and Configuration spaces.
Message Signaled Interrupt, MSI	An optional feature that enables a device to request service by writing a system-specified DW of data to a system-specified address using a Memory Write semantic Request.
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-byte *naturally aligned* means the least significant 6 bits of the byte address are 00 0000b.
Packet	A fundamental unit of information transfer consisting of a header that, in some cases, is followed by a data payload.
PCI bus	The PCI Local Bus, as specified in the PCI 2.3 and PCI-X 2.0 specifications.
PCI Software Model	The software model necessary to initialize, discover, configure, and use a PCI device, as specified in the PCI 2.3, PCI-X 2.0, and PCI BIOS specifications.
Phantom Function Number, PFN	An unclaimed function number that may be used to expand the number of outstanding transaction identifiers by logically combining the *PFN* with the Tag identifier to create a unique transaction identifier.
Physical Lane	See Lane.
Physical Layer	The Layer that directly interacts with the communication medium between two components.
Port	1. Logically, an interface between a component and a PCI Express Link. 2. Physically, a group of Transmitters and Receivers located on the same chip that define a Link.
PPM	Parts per Million. Applied to frequency, the difference, in millionths of a Hertz, between a stated ideal frequency, and the measured long-term average of a frequency.
Quality of Service, QoS	Attributes affecting the bandwidth, latency, jitter, relative priority, etc., for differentiated classes of traffic.

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).
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. Any implementation dependence on a *reserved* field value or encoding will result in an implementation that is not PCI Express-compliant. The functionality 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 that first introduces a transaction sequence into the PCI Express 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-PCI Bridge.
sideband signaling	A method for signaling events and conditions using physical signals separate from the signals forming the Link between two components. All mechanisms defined in this document can be implemented using in-band signaling, although in some form factors sideband signaling may be used instead.
Standard Hot-Plug Controller (SHPC)	A PCI Hot-Plug Controller compliant with SHPC 1.0.
Split Transaction	A single logical transfer containing an initial transaction (the Request) terminated at the target (the Completer), followed by one or more transactions initiated by the Completer in response to the Request.
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.
Symbol	A 10-bit quantity produced as the result of 8b/10b encoding.
Symbol Time	The period of time required to place a Symbol on a Lane (10 times the Unit Interval).
system element	Logical devices or groups of devices that operate according distinct sets of rules. The following *system elements* are defined: Root Complex, Endpoint, Switch, and Bridge.
Tag	A number assigned to a given Non-posted Request to distinguish Completions for that Request from other Requests.

Transaction Descriptor
> An element of a Packet header that, in addition to Address, Length, and Type, describes the properties of the Transaction.

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.

Transceiver
> The physical Transmitter and Receiver pair on a single chip.

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.

Unsupported Request, UR
> A Request Packet that specifies some action or access to some space that is not supported by the Completer.

Unit Interval, UI
> Given a data stream of a repeating pattern of alternating 1 and 0 values, the *Unit Interval* is the value measured by averaging the time interval between voltage transitions, over a time interval long enough to make all intentional frequency modulation of the source clock negligible.

Upstream
> 1. The relative position of an interconnect/system element (Port/component) that is closer to the Root Complex. The Port on a Switch that is closest topologically to the Root Complex is the *Upstream* Port. The Port on an Endpoint or Bridge component is an *Upstream* Port. The *Upstream* component on a Link is the component closer to the Root Complex. 2. A direction of information flow where the information is flowing towards the Root Complex.

variant
> A field of a TLP header that contains a value that is subject to possible modification according to the rules of this specification as the TLP flows through the PCI Express fabric.

wakeup
> An optional mechanism used by a component to request the reapplication of main power when in the L2 Link state. Two such mechanisms are defined: Beacon (using in-band signaling) and WAKE# (using sideband signaling).

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

Reference Documents

PCI Express Card Electromechanical Specification, Rev. 1.0

PCI Express to PCI/PCI-X Bridge Specification, Rev. 1.0

Mini PCI Express Card Electromechanical Specification, Rev. 1.0

PCI Local Bus Specification, Rev. 2.3

PCI-X Addendum to the PCI Local Bus Specification, Rev. 2.0

PCI Hot-Plug Specification, Rev. 1.1

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

PCI-to-PCI Bridge Architecture Specification, Rev. 1.1

PCI Power Management Interface Specification, Rev. 1.1

Advanced Configuration and Power Interface Specification, Rev. 2.0

Guidelines for 64-bit Global Identifier (EUI-64) Registration Authority

1

1.   Introduction

This chapter presents an overview of the PCI Express architecture and key concepts. PCI Express is a high performance, general purpose I/O interconnect defined for a wide variety of future computing and communication platforms. Key PCI attributes, such as its usage model, load-store
5   architecture, and software interfaces, are maintained, whereas its parallel bus implementation is replaced by a highly scalable, fully serial interface. PCI Express takes advantage of recent advances in point-to-point interconnects, Switch-based technology, and packetized protocol to deliver new levels of performance and features. Power Management, Quality Of Service (QoS), Hot-Plug/Hot-Swap support, Data Integrity, and Error Handling are among some of the advanced features
10   supported by PCI Express.

1.1.   A Third Generation I/O Interconnect

The high-level requirements for this third generation I/O interconnect are as follows:

❑ **Supports multiple market segments and emerging applications:**

- Unifying I/O architecture for desktop, mobile, workstation, server, communications
15   platforms, and embedded devices

❑ **Ability to deliver low cost, high volume solutions:**

- Cost at or below PCI cost structure at the system level

❑ **Support multiple platform interconnect usages:**

- Chip-to-chip, board-to-board via connector or cabling

20 ❑ **New mechanical form factors:**

- Mobile, PCI-like form factor and modular, cartridge form factor

❑ **PCI compatible software model:**

- Ability to enumerate and configure PCI Express hardware using PCI system configuration software implementations with no modifications

25 - Ability to boot existing operating systems with no modifications

- Ability to support existing I/O device drivers with no modifications

- Ability to configure/enable new PCI Express functionality by adopting the PCI configuration paradigm

❑ **Performance:**

- Low-overhead, low-latency communications to maximize application payload bandwidth and Link efficiency

- High-bandwidth per pin to minimize pin count per device and connector interface

- Scalable performance via aggregated Lanes and signaling frequency

❑ **Advanced features:**

- Comprehend different data types and ordering rules

- Power management and budgeting

  ♦ Ability to identify power management capabilities of a given function

  ♦ Ability to transition a function into a specific power state

  ♦ Ability to receive notification of the current power state of a function

  ♦ Ability to generate a request to wakeup from a power-off state of the main power supply

  ♦ Ability to sequence device power-up to allow graceful platform policy in power budgeting.

- Ability to support differentiated services, i.e., different qualities of service (QoS)

  ♦ Ability to have dedicated Link resources per QoS data flow to improve fabric efficiency and effective application-level performance in the face of head-of-line blocking

  ♦ Ability to configure fabric QoS arbitration policies within every component

  ♦ Ability to tag end-to-end QoS with each packet

  ♦ Ability to create end-to-end isochronous (time-based, injection rate control) solutions

- Hot-Plug and Hot-Swap support

  ♦ Ability to support existing PCI Hot-Plug and Hot-Swap solutions

  ♦ Ability to support native Hot-Plug and Hot-Swap solutions (no sideband signals required)

  ♦ Ability to support a unified software model for all form factors

- Data Integrity

    ♦ Ability to support Link-level data integrity for all types of transaction and Data Link packets

    ♦ Ability to support end-to-end data integrity for high availability solutions

₅
- Error Handling

    ♦ Ability to support PCI-level error handling

    ♦ Ability to support advanced error reporting and handling to improve fault isolation and recovery solutions

- Process Technology Independence

₁₀
    ♦ Ability to support different DC common mode voltages at Transmitter and Receiver

- Ease of Testing

    ♦ Ability to test electrical compliance via simple connection to test equipment

1.2.   PCI Express Link

A Link represents a dual-simplex communications channel between two components. The
₁₅ fundamental PCI Express Link consists of two, low-voltage, differentially driven signal pairs: a
Transmit pair and a Receive pair as shown in Figure 1-1.

Packet

Component A

Component B

Packet
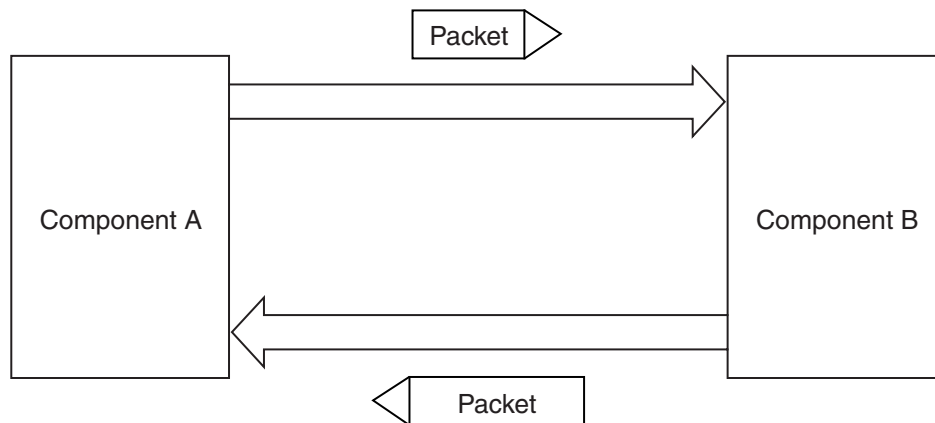
OM13750

Figure 1-1: PCI Express Link

The primary Link attributes are:

❑ The basic Link – PCI Express Link consists of dual unidirectional differential Links, implemented as a Transmit pair and a Receive pair. A data clock is embedded using an encoding scheme (refer to Chapter 4) to achieve very high data rates.

5 ❑ Signaling rate – Once initialized, each Link must only operate at one of the supported signaling levels. For the first generation of PCI Express technology, there is only one signaling rate defined, which provides an effective 2.5 Gigabits/second/Lane/direction of raw bandwidth. The data rate is expected to increase with technology advances in the future.

❑ Lanes – A Link must support at least one Lane – each Lane represents a set of differential signal
10 pairs (one pair for transmission, one pair for reception). To scale bandwidth, a Link may aggregate multiple Lanes denoted by xN where N may be any of the supported Link widths. An x8 Link represents an aggregate bandwidth of 20 Gigabits/second of raw bandwidth in each direction. This specification describes operations for x1, x2, x4, x8, x12, x16, and x32 Lane widths.

15 ❑ Initialization – During hardware initialization, each PCI Express Link is set up following a negotiation of Lane widths and frequency of operation by the two agents at each end of the Link. No firmware or operating system software is involved.

❑ Symmetry – Each Link must support a symmetric number of Lanes in each direction, i.e., a x16 Link indicates there are 16 differential signal pairs in each direction.

20 1.3.   PCI Express Fabric Topology

A fabric is composed of point-to-point Links that interconnect a set of components – an example fabric topology is shown in Figure 1-2. This figure illustrates a single fabric instance referred to as a hierarchy – composed of a Root Complex (RC), multiple Endpoints (I/O devices), a Switch, and a PCI Express-PCI Bridge, all interconnected via PCI Express Links. Each of the components of the
25 topology are mapped in a single flat address space and can be accessed using PCI-like load/store accesses transaction semantics.
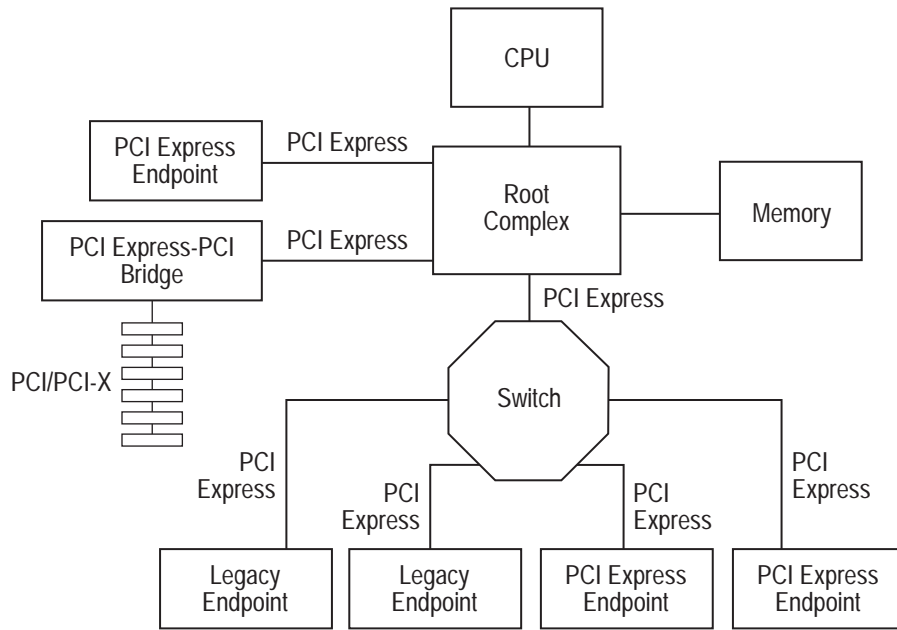
OM13751

Figure 1-2: Example Topology

1.3.1. Root Complex

❑ A Root Complex (RC) denotes the root of an I/O hierarchy that connects the CPU/memory subsystem to the I/O.

❑ As illustrated in Figure 1-2, a Root Complex may support one or more PCI Express Ports. Each interface defines a separate hierarchy domain. Each hierarchy domain may be composed of a single Endpoint or a sub-hierarchy containing one or more Switch components and Endpoints.

❑ The capability to route peer-to-peer transactions between hierarchy domains through a Root Complex is optional and implementation dependent. For example, an implementation may incorporate a real or virtual Switch internally within the Root Complex to enable full peer-to-peer support in a software transparent way.

❑ A Root Complex must support generation of configuration requests as a Requester.

❑ A Root Complex is permitted to support the generation of I/O requests as a Requester.

❑ A Root Complex must not support Lock semantics as a Completer.

❑ A Root Complex is permitted to support generation of Locked Requests as a Requester.

1.3.2.   Endpoints

Endpoint refers to a type of device that can be the Requester or Completer of a PCI Express transaction either on its own behalf or on behalf of a distinct non-PCI Express device (other than a PCI device or Host CPU), e.g., a PCI Express attached graphics controller or a PCI Express-USB host controller. Endpoints are classified as either legacy or PCI Express Endpoints.

1.3.2.1.   Legacy Endpoint Rules

❑ A legacy Endpoint must be a device with a Type 00h Configuration Space header.

❑ A legacy Endpoint must support Configuration Requests as a Completer

❑ A legacy Endpoint may support I/O Requests as a Completer.

❑ A legacy Endpoint may generate I/O Requests.

❑ A legacy Endpoint may support Lock memory semantics as a Completer if that is required by the device's legacy software support requirements.

❑ A legacy Endpoint must not issue a Locked Request.

❑ A legacy Endpoint may implement extended configuration space capabilities, but such capabilities may be ignored by software.

❑   A legacy Endpoint operating as the Requester of a Memory Transaction is not required to be capable of generating addresses 4 GB or greater.

❑   A legacy Endpoint is required to support MSI if an interrupt resource is requested, but is permitted to support either the 32-bit or 64-bit Message Address version of the MSI capability structure.

❑   A legacy Endpoint is permitted to support 32-bit addressing for Base Address registers that request memory resources.

❑ A legacy Endpoint must appear within one of the hierarchy domains originated by the Root Complex.

1.3.2.2.   PCI Express Endpoint Rules

❑ A PCI Express Endpoint must be a device with a Type 00h Configuration Space header.

❑ A PCI Express Endpoint must support Configuration Requests as a Completer.

❑ A PCI Express Endpoint must not depend on operating system allocation of I/O resources claimed through BAR(s).

❑ A PCI Express Endpoint must not generate I/O Requests.

❑ A PCI Express Endpoint must not support Locked Requests as a Completer or generate them as a Requestor. PCI Express-compliant software drivers and applications must be written to prevent the use of lock semantics when accessing a PCI Express Endpoint.

❑ A PCI Express Endpoint operating as the Requester of a Memory Transaction is required to be capable of generating addresses greater than 4 GB.

❑ A PCI Express Endpoint is required to support MSI if an interrupt resource is requested, and must support the 64-bit Message Address version of the MSI capability structure.

5 ❑ A PCI Express Endpoint requesting memory resources through a BAR must set the BAR's Prefetchable bit unless the range contains locations with read side-effects or locations in which the device does not tolerate write merging.

❑ For a PCI Express Endpoint, 64-bit addressing must be supported for all BARs that have the prefetchable bit set. 32-bit addressing is permitted for all BARs that do not have the prefetchable bit set.

10

❑ The minimum memory address range requested by a BAR is 128 bytes.

❑ A PCI Express Endpoint must appear within one of the hierarchy domains originated by the Root Complex.

1.3.3.　Switch

15 A Switch is defined as a logical assembly of multiple virtual PCI-to-PCI Bridge devices as illustrated in Figure 1-3. All Switches are governed by the following base rules.
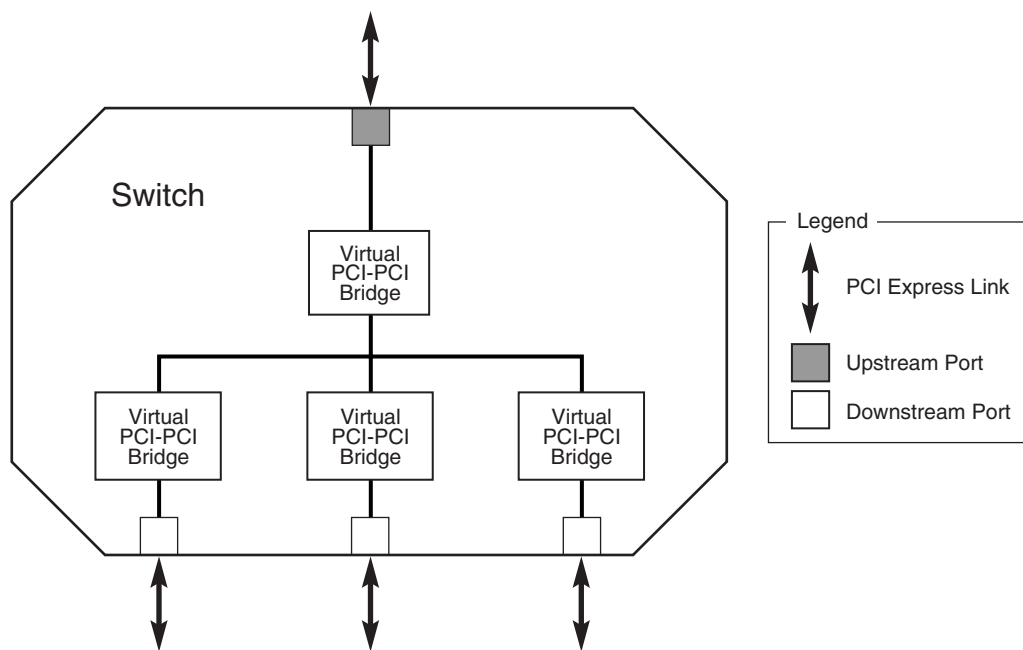
OM13752

Figure 1-3: Logical Block Diagram of a Switch

❑ Switches appear to configuration software as two or more logical PCI-to-PCI Bridges.

❑ A Switch forwards transactions using PCI Bridge mechanisms; e.g., address based routing.

❑ Except as noted in this document, a Switch must forward all types of Transaction Layer Packets
20　　between any set of Ports.

- ❑ Locked Requests must be supported as specified in Section 6.5. Switches are not required to support Downstream Ports as initiating Ports for Locked requests.

- ❑ Each enabled Switch Port must comply with the flow control specification within this document.

5 - ❑ A Switch is not allowed to split a packet into smaller packets, e.g., a single packet with a 256-byte payload must not be divided into two packets of 128 bytes payload each.

- ❑ Arbitration between Ingress Ports (inbound Link) of a Switch may be implemented using round robin or weighted round robin when contention occurs on the same Virtual Channel. This is described in more detail later within the specification.

10 - ❑ Endpoint devices (represented by Type 00h Configuration Space headers) may not appear to configuration software on the switch's internal bus as peers of the virtual PCI-to-PCI Bridges representing the Switch Downstream Ports.

1.3.4. PCI Express-PCI Bridge

- ❑ A PCI Express-PCI Bridge provides a connection between a PCI Express fabric and a
15 PCI/PCI-X hierarchy.

- ❑ PCI Express Port(s) of a PCI Express-PCI Bridge must comply with the requirements of this document.

1.4. PCI Express Fabric Topology Configuration

The PCI Express Configuration model supports two mechanisms:

20 - ❑ PCI compatible configuration mechanism: The PCI compatible mechanism supports 100% binary compatibility with PCI 2.3 or later operating systems and their corresponding bus enumeration and configuration software.

- ❑ PCI Express enhanced configuration mechanism: The enhanced mechanism is provided to increase the size of available configuration space and to optimize access mechanisms.

25 Each PCI Express Link is mapped through a virtual PCI-to-PCI Bridge structure and has a logical PCI bus associated with it. The virtual PCI-to-PCI Bridge structure may be part of a PCI Express Root Complex Port, a Switch Upstream Port, or a Switch Downstream Port. A Root Port is a virtual PCI-to-PCI Bridge structure that originates a PCI Express hierarchy domain from a PCI Express Root Complex. Logical devices are mapped into configuration space such that each will
30 respond to a particular device number.

1.5. PCI Express Layering Overview

This document specifies the architecture in terms of three discrete logical layers: the Transaction Layer, the Data Link Layer, and the Physical Layer. Each of these layers is divided into two sections: one that processes outbound (to be transmitted) information and one that processes inbound
35 (received) information, as shown in Figure 1-4.

The fundamental goal of this layering definition is to facilitate the reader's understanding of the specification. Note that this layering does not imply a particular PCI Express implementation.
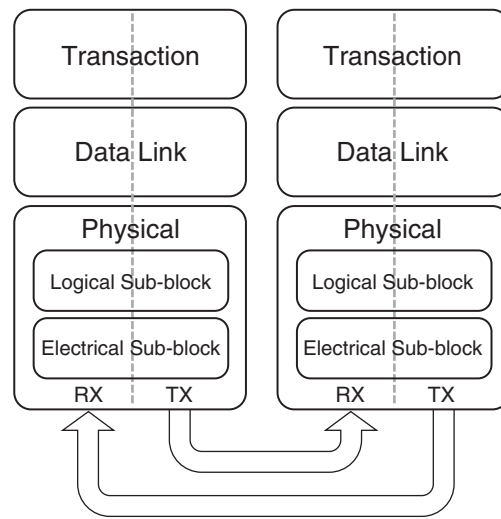
OM13753

Figure 1-4: High-Level Layering Diagram

PCI Express uses packets to communicate information between components. Packets are formed in the Transaction and Data Link Layers to carry the information from the transmitting component to the receiving component. As the transmitted packets flow through the other layers, they are extended with additional information necessary to handle packets at those layers. At the receiving side the reverse process occurs and packets get transformed from their Physical Layer representation to the Data Link Layer representation and finally (for Transaction Layer Packets) to the form that can be processed by the Transaction Layer of the receiving device. Figure 1-5 shows the conceptual flow of transaction level packet information through the layers.
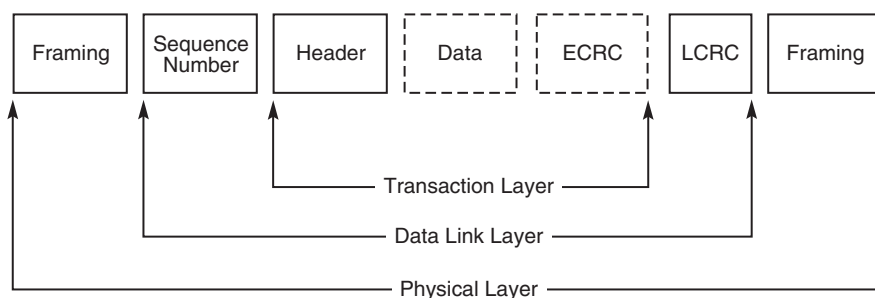
OM13754

Figure 1-5: Packet Flow Through the Layers

Note that a simpler form of packet communication is supported between two Data Link Layers (connected to the same Link) for the purpose of Link management.

1.5.1.   Transaction Layer

The upper Layer of the architecture is the Transaction Layer. The Transaction Layer's primary responsibility is the assembly and disassembly of Transaction Layer Packets (TLPs). TLPs are used to communicate transactions, such as read and write, as well as certain types of events. The
5   Transaction Layer is also responsible for managing credit-based flow control for TLPs.

Every request packet requiring a response packet is implemented as a split transaction. Each packet has a unique identifier that enables response packets to be directed to the correct originator. The packet format supports different forms of addressing depending on the type of the transaction (Memory, I/O, Configuration, and Message). The Packets may also have attributes such as No
10   Snoop and Relaxed Ordering.

The transaction Layer supports four address spaces: it includes the three PCI address spaces (memory, I/O, and configuration) and adds a Message Space. This specification uses Message Space to support all prior sideband signals, such as interrupts, power-management requests, and so on, as in-band Message transactions. You could think of PCI Express Message transactions as "virtual
15   wires" since their effect is to eliminate the wide array of sideband signals currently used in a platform implementation.

1.5.2.   Data Link Layer

The middle Layer in the stack, the Data Link Layer, serves as an intermediate stage between the Transaction Layer and the Physical Layer. The primary responsibilities of the Data Link Layer
20   include Link management and data integrity, including error detection and error correction.

The transmission side of the Data Link Layer accepts TLPs assembled by the Transaction Layer, calculates and applies a data protection code and TLP sequence number, and submits them to Physical Layer for transmission across the Link. The receiving Data Link Layer is responsible for checking the integrity of received TLPs and for submitting them to the Transaction Layer for
25   further processing. On detection of TLP error(s), this Layer is responsible for requesting retransmission of TLPs until information is correctly received, or the Link is determined to have failed.

The Data Link Layer also generates and consumes packets that are used for Link management functions. To differentiate these packets from those used by the Transaction Layer (TLP), the term
30   Data Link Layer Packet (DLLP) will be used when referring to packets that are generated and consumed at the Data Link Layer.

1.5.3.   Physical Layer

The Physical Layer includes all circuitry for interface operation, including driver and input buffers, parallel-to-serial and serial-to-parallel conversion, PLL(s), and impedance matching circuitry. It
35   includes also logical functions related to interface initialization and maintenance. The Physical Layer exchanges information with the Data Link Layer in an implementation-specific format. This Layer is responsible for converting information received from the Data Link Layer into an appropriate serialized format and transmitting it across the PCI Express Link at a frequency and width compatible with the device connected to the other side of the Link.

The PCI Express architecture has "hooks" to support future performance enhancements via speed upgrades and advanced encoding techniques. The future speeds, encoding techniques or media may only impact the Physical Layer definition.

1.5.4. Layer Functions and Services

1.5.4.1. Transaction Layer Services

The Transaction Layer, in the process of generating and receiving TLPs, exchanges Flow Control information with its complementary Transaction Layer on the other side of the Link. It is also responsible for supporting both software and hardware-initiated power management.

Initialization and configuration functions require the Transaction Layer to:

❑ Store Link configuration information generated by the processor or management device

❑ Store Link capabilities generated by Physical Layer hardware negotiation of width and operational frequency

A Transaction Layer's Packet generation and processing services require it to:

❑ Generate TLPs from device core Requests

❑ Convert received Request TLPs into Requests for the device core

❑ Convert received Completion Packets into a payload, or status information, deliverable to the core

❑ Detect unsupported TLPs and invoke appropriate mechanisms for handling them

❑ If end-to-end data integrity is supported, generate the end-to-end data integrity CRC and update the TLP header accordingly.

Flow control services:

❑ The Transaction Layer tracks flow control credits for TLPs across the Link.

❑ Transaction credit status is periodically transmitted to the remote Transaction Layer using transport services of the Data Link Layer.

❑ Remote Flow Control information is used to throttle TLP transmission.

Ordering rules:

❑ PCI/PCI-X compliant producer consumer ordering model

❑ Extensions to support relaxed ordering

Power management services:

❑ ACPI/PCI power management, as dictated by system software.

❑ Hardware-controlled autonomous power management minimizes power during full-on power states.

5   Virtual Channels and Traffic Class:

❑ The combination of Virtual Channel mechanism and Traffic Class identification is provided to support differentiated services and QoS support for certain classes of applications.

❑ Virtual Channels: Virtual Channels provide a means to support multiple independent logical data flows over given common physical resources of the Link. Conceptually this involves
10   multiplexing different data flows onto a single physical Link.

❑ Traffic Class: The Traffic Class is a Transaction Layer Packet label that is transmitted unmodified end-to-end through the fabric. At every service point (e.g., Switch) within the fabric, Traffic Class labels are used to apply appropriate servicing policies. Each Traffic Class label defines a unique ordering domain - no ordering guarantees are provided for packets that
15   contain different Traffic Class labels.

1.5.4.2.   *Data Link Layer Services*

The Data Link Layer is responsible for reliably exchanging information with its counterpart on the opposite side of the Link.

Initialization and power management services:

20   ❑ Accept power state Requests from the Transaction Layer and convey to the Physical Layer

❑ Convey active/reset/disconnected/power managed state to the Transaction Layer

Data protection, error checking, and retry services:

❑ CRC generation

❑ Transmitted TLP storage for Data Link level retry

25   ❑ Error checking

❑ TLP acknowledgment and retry Messages

❑ Error indication for error reporting and logging

1.5.4.3. Physical Layer Services

Interface initialization, maintenance control, and status tracking:

- ❑ Reset/Hot-Plug control/status
- ❑ Interconnect power management
- ❑ Width and Lane mapping negotiation
- ❑ Polarity reversal

Symbol and special ordered set generation:

- ❑ 8-bit/10-bit encoding/decoding
- ❑ Embedded clock tuning and alignment

Symbol transmission and alignment:

- ❑ Transmission circuits
- ❑ Reception circuits
- ❑ Elastic buffer at receiving side
- ❑ Multi-Lane de-skew (for widths > x1) at receiving side

System DFT support features

1.5.4.4. Inter-Layer Interfaces

1.5.4.4.1. Transaction/Data Link Interface

The Transaction to Data Link interface provides:

- ❑ Byte or multi-byte data to be sent across the Link
  - Local TLP-transfer handshake mechanism
  - TLP boundary information
- ❑ Requested power state for the Link

The Data Link to Transaction interface provides:

- ❑ Byte or multi-byte data received from the PCI Express Link
- ❑ TLP framing information for the received byte
- ❑ Actual power state for the Link
- ❑ Link status information

1.5.4.4.2. Data Link/Physical Interface

The Data Link to Physical interface provides:

❑ Byte or multi-byte wide data to be sent across the Link

   • Data transfer handshake mechanism

   • TLP and DLLP boundary information for bytes

❑ Requested power state for the Link

The Physical to Data Link interface provides:

❑ Byte or multi-byte wide data received from the PCI Express Link

❑ TLP and DLLP framing information for data

❑ Indication of errors detected by the Physical Layer

❑ Actual power state for the Link

❑ Connection status information

2

2.   Transaction Layer Specification

2.1.   Transaction Layer Overview
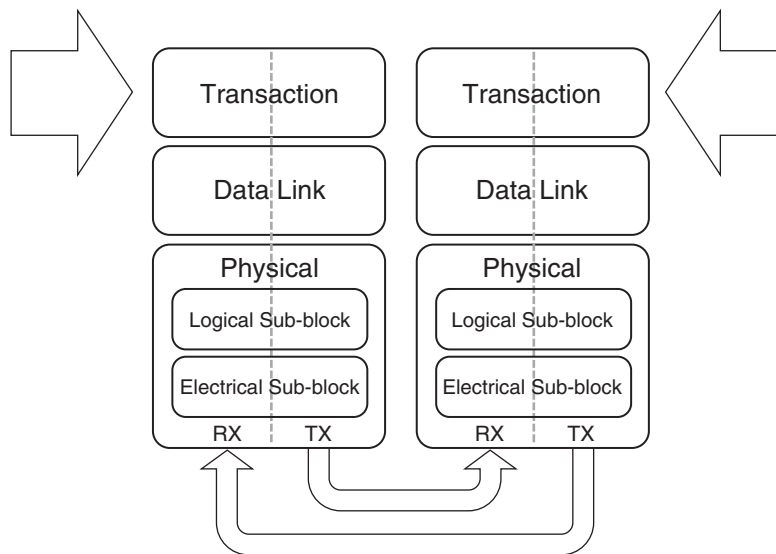
OM14295

Figure 2-1: Layering Diagram Highlighting the Transaction Layer

At a high level, the key aspects of PCI Express associated with the Transaction Layer are:

❑   A pipelined full split-transaction protocol

❑   Mechanisms for differentiating the ordering and processing requirements of Transaction Layer
Packets (TLPs)

❑   Credit-based flow control

❑   Optional support for data poisoning and end-to-end data integrity detection.

The Transaction Layer comprehends the following:

❑ TLP construction and processing

❑ Association of PCI Express transaction-level mechanisms with device resources including:

- Flow Control

5

- Virtual Channel management

❑ Rules for ordering and management of TLPs

- PCI/PCI-X compatible ordering

- Including Traffic Class differentiation

This chapter specifies the behaviors associated with the Transaction Layer.

10   2.1.1.   Address Spaces, Transaction Types, and Usage

Transactions form the basis for information transfer between a Requester and Completer. Four address spaces are defined within the PCI Express architecture, and different Transaction types are defined, each with its own unique intended usage, as shown in Table 2-1.

Table 2-1: Transaction Types for Different Address Spaces

Address Space	Transaction Types	Basic Usage
Memory	Read Write	Transfer data to/from a memory-mapped location.
I/O	Read Write	Transfer data to/from an I/O-mapped location
Configuration	Read Write	Device configuration/setup
Message	Baseline (including Vendor–defined)	From event signaling mechanism to general purpose messaging

Details about the rules associated with usage of these address formats and the associated TLP
15   formats are described later in this chapter.

2.1.1.1.   Memory Transactions

Memory Transactions include the following types:

❑ Read Request/Completion

❑ Write Request

5   Memory Transactions use two different address formats:

❑ Short Address Format: 32-bit address

❑ Long Address Format: 64-bit address

2.1.1.2.   I/O Transactions

PCI Express supports I/O Space for compatability with legacy devices which require their use.
10   Future revisions of this specification are expected to depreciate the use of I/O Space. I/O
Transactions include the following types:

❑ Read Request/Completion

❑ Write Request/Completion

I/O Transactions use a single address format:

15   ❑ Short Address Format: 32-bit address

2.1.1.3.   Configuration Transactions

Configuration Transactions are used to access configuration registers of PCI Express devices.

Configuration Transactions include the following types:

❑ Read Request/Completion

20   ❑ Write Request/Completion

2.1.1.4.   Message Transactions

The Message Transactions, or simply Messages, are used to support in-band communication of
events between PCI Express devices.

In addition to the specified Messages, PCI Express provides support for vendor-defined Messages
25   using specified Message codes. The definition of specific vendor-defined Messages is outside the
scope of this document.

This specification establishes a standard framework within which vendors can specify their own
vendor-defined Messages tailored to fit the specific requirements of their platforms (see Sections
2.2.8.5 and 2.2.8.7).

30   Note that these vendor-defined Messages are not guaranteed to be interoperable with components
from different vendors.

2.1.2. Packet Format Overview

Transactions consist of Requests and Completions, which are communicated using packets. Figure 2-2 shows a high level serialized view of a Transaction Layer Packet (TLP), consisting of a TLP header, a data payload (for some types of packets), and an optional TLP digest. Figure 2-3 shows a more detailed view of the TLP. The following sections of this chapter define the detailed structure of the packet headers and digest.
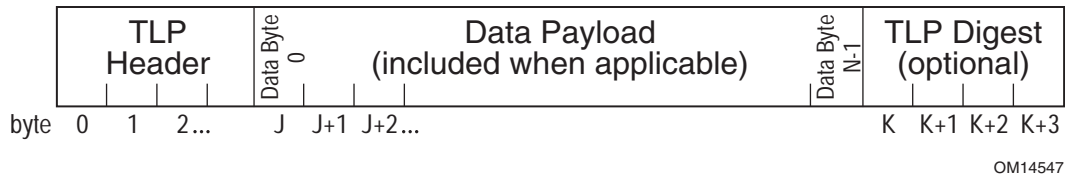
Figure 2-2: Serial View of a TLP

PCI Express conceptually transfers information as a serialized stream of bytes as shown in Figure 2-2. Note that at the byte level, information is transmitted/received over the interconnect with byte 0 being transmitted/received first. Refer to Section 4.2 for details on how individual bytes of the packet are encoded and transmitted over the physical media.

Detailed layouts of the TLP header and TLP digest (presented in generic form in Figure 2-3) are drawn with the lower numbered bytes on the left rather than on the right as has traditionally been depicted in other PCI specifications. The PCI Express header layout is optimized for performance on a serialized interconnect, driven by the requirement that the most time critical information be transferred first. For example, within the PCI Express TLP header, the most significant byte of the address field is transferred first so that it may be used for early address decode.
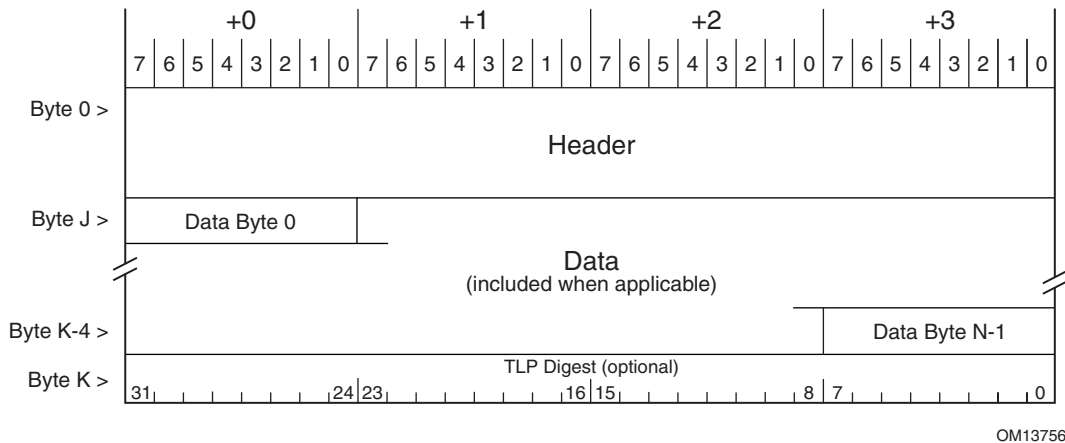
Figure 2-3: Generic TLP Format

Payload data within a TLP is depicted with the lowest addressed byte (byte J in Figure 2-3) shown to the upper left. Detailed layouts depicting data structure organization (such as the Configuration Space depictions in Chapter 7) retain the traditional PCI byte layout with the lowest addressed byte shown on the right. Regardless of depiction, all bytes are conceptually transmitted over the Link in increasing byte number order.

Depending on the type of a packet, the header for that packet will include some of the following types of fields:

❑ Format of the packet

❑ Type of the packet

5 ❑ Length for any associated data

❑ Transaction Descriptor, including:

- Transaction ID

- Attributes

- Traffic Class

10 ❑ Address/routing information

❑ Byte Enables

❑ Message encoding

❑ Completion status

2.2.   Transaction Layer Protocol - Packet
15   Definition

PCI Express uses a packet based protocol to exchange information between the Transaction Layers of the two components communicating with each other over the Link. PCI Express supports the following basic transaction types: Memory, I/O, Configuration, and Messages. Two addressing formats for Memory Requests are supported: 32 bit and 64 bit.

20 Transactions are carried using Requests and Completions. Completions are used only where required, for example, to return read data, or to acknowledge Completion of I/O and Configuration Write Transactions. Completions are associated with their corresponding Requests by the value in the Transaction ID field of the Packet header.

All TLP fields marked Reserved (sometimes abbreviated as R) must be filled with all 0's when a TLP
25 is formed. Values in such fields must be ignored by Receivers. Note that for certain fields there are both specified and reserved values – the handling of reserved values in these cases is specified separately for each case.

2.2.1.   Common Packet Header Fields

All Transaction Layer Packet (TLP) headers contain the following fields (see Figure 2-4):

30 ❑ Fmt[1:0] – Format of TLP (see Table 2-2) – bits 6:5 of byte 0

❑ Type[4:0] – Type of TLP – bits 4:0 of byte 0

The Fmt and Type fields provide the information required to determine the size of the remaining part of the header, and if the packet contains a data payload following the header. The Fmt, Type,

TD and Length fields contain all information necessary to determine the overall size of the TLP itself. The Type field, in addition to defining the type of the TLP also determines how the TLP is routed by a Switch. Different types of TLPs are discussed in more detail in the following sections.

❑ Permitted Fmt[1:0] and Type[4:0] field values are shown in Table 2-3.

• All other encodings are reserved.

❑ TC[2:0] – Traffic Class (see Section 2.4.2) – bits [6:4] of byte 1

❑ Attr[1:0] – Attributes (see Section 2.2.6.3) – bits [5:4] of byte 2

❑ TD – 1b indicates presence of TLP digest in the form of a single DW at the end of the TLP (see Section 2.2.3) – bit 7 of byte 2

❑ EP – indicates the TLP is poisoned (see Section 2.7) – bit 6 of byte 2

❑ Length[9:0] – Length of data payload in DW (see Table 2-4) – bits 1:0 of byte 2 concatenated with bits 7:0 of byte 3

• TLP data must be 4-byte naturally aligned and in increments of 4-byte Double Words (DW).

• Reserved for TLPs that do not contain or refer to data payloads, including Cpl, CplLk, and Messages (except as specified)
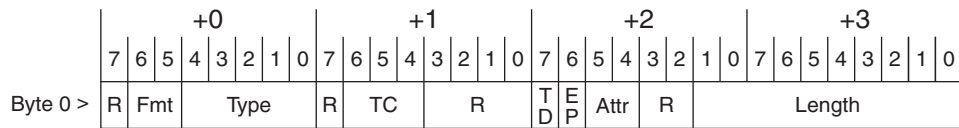
OM14540

Figure 2-4: Fields Present in all TLPs

Table 2-2: Fmt[1:0] Field Values

Fmt[1:0]	Corresponding TLP Format
00b	3 DW header, no data
01b	4 DW header, no data
10b	3 DW header, with data
11b	4 DW header, with data

Table 2-3: Fmt[1:0] and Type[4:0] Field Encodings

TLP Type	Fmt [1:0][2]	Type [4:0]	Description
MRd	00 01	0 0000	Memory Read Request
MRdLk	00 01	0 0001	Memory Read Request-Locked
MWr	10 11	0 0000	Memory Write Request
IORd	00	0 0010	I/O Read Request
IOWr	10	0 0010	I/O Write Request
CfgRd0	00	0 0100	Configuration Read Type 0
CfgWr0	10	0 0100	Configuration Write Type 0
CfgRd1	00	0 0101	Configuration Read Type 1
CfgWr1	10	0 0101	Configuration Write Type 1
Msg	01	1 0$r_2r_1r_0$	Message Request – The sub-field r[2:0] specifies the Message routing mechanism (see Table 2-11).
MsgD	11	1 0$r_2r_1r_0$	Message Request with data payload – The sub-field r[2:0] specifies the Message routing mechanism (see Table 2-11).
Cpl	00	0 1010	Completion without Data – Used for I/O and Configuration Write Completions and Read Completions (I/O, Configuration, or Memory) with Completion Status other than Successful Completion.
CplD	10	0 1010	Completion with Data – Used for Memory, I/O, and Configuration Read Completions.
CplLk	00	0 1011	Completion for Locked Memory Read without Data – Used only in error case.
CplDLk	10	0 1011	Completion for Locked Memory Read – otherwise like CplD.
			All encodings not shown above are Reserved.

---

[2] Requests with two Fmt[1:0] values shown can use either 32 bits (the first value) or 64 bits (the second value) Addressing Packet formats.

Table 2-4: Length[9:0] Field Encoding

Length[9:0]	Corresponding TLP Data Payload Size
00 0000 0001b	1 DW
00 0000 0010b	2 DW
...	...
11 1111 1111b	1023 DW
00 0000 0000b	1024 DW

2.2.2. TLPs with Data Payloads - Rules

❑ Length is specified as an integral number of DW

❑ Length[9:0] is reserved for all Messages except those which explicitly refer to a Data Length

- See Message Code table in Section 2.2.8.

5 ❑ The Transmitter of a TLP with a data payload must not allow the data payload length to exceed the length specified by the value in the Max_Payload_Size field of the Transmitter's Device Control register (see Section 7.8.4).

- Note: Max_Payload_Size applies only to TLPs with data payloads; Memory Read Requests are not restricted in length by Max_Payload_Size. The size of the Memory Read Request is 10 controlled by the Length field

❑ The data payload of a Received TLP must not exceed the length specified by the value in the Max_Payload_Size field of the Receiver's Device Control register (see Section 7.8.4).

- Receivers must check for violations of this rule. If a Receiver determines that a TLP violates this rule, the TLP is a Malformed TLP

15 ♦ This is a reported error associated with the Receiving Port (see Section 6.2)

❑ For TLPs, that include data, the value in the Length field and the actual amount of data included in the TLP must match.

- Receivers must check for violations of this rule. If a Receiver determines that a TLP violates this rule, the TLP is a Malformed TLP

20 ♦ This is a reported error associated with the Receiving Port (see Section 6.2)

❑ The value in the Length field applies only to data – the TLP Digest is not included in the Length

❑ When a data payload is included in a TLP, the first byte of data following the header corresponds to the byte address closest to zero and the succeeding bytes are in increasing byte address sequence.

25 - Example: For a 16-byte write to location 100h, the first byte following the header would be the byte to be written to location 100h, and the second byte would be written to location 101h, and so on, with the final byte written to location 10Fh.

🗒 IMPLEMENTATION NOTE

Maintaining Alignment in Data Payloads

Section 2.3.1.1 discusses rules for forming Read Completions respecting certain natural address boundaries. Memory Write performance can be significantly improved by respecting similar address boundaries in the formation of the Write Request. Specifically, forming Write Requests such that natural address boundaries of 64 or 128 bytes are respected will help to improve system performance.

2.2.3. TLP Digest Rules

❑ For any TLP, a value of 1b in the TD field indicates the presence of the TLP Digest field including an ECRC value at the end of the TLP

- A TLP where the TD field value does not correspond with the observed size (accounting for the data payload, if present) is a Malformed TLP

  ♦ This is a reported error associated with the Receiving Port (see Section 6.2)

❑ If the device at the ultimate destination of the TLP does not support ECRC checking, the device must ignore the TLP Digest

- If the device at the ultimate destination of the TLP supports ECRC checking, the device interprets the value in the TLP Digest field as an ECRC value, according to the rules in Section 2.7.1

2.2.4. Routing and Addressing Rules

There are three principal mechanisms for TLP routing: address, ID, and implicit. This section defines the rules for the address and ID routing mechanisms. Implicit routing is used only with Message Requests, and is covered in Section 2.2.8.

2.2.4.1. Address Based Routing Rules

❑ Address routing is used with Memory and I/O Requests.

❑ Two address formats are specified, a 64-bit format used with a 4 DW header (see Figure 2-5) and a 32-bit format used with a 3 DW header (see Figure 2-6).
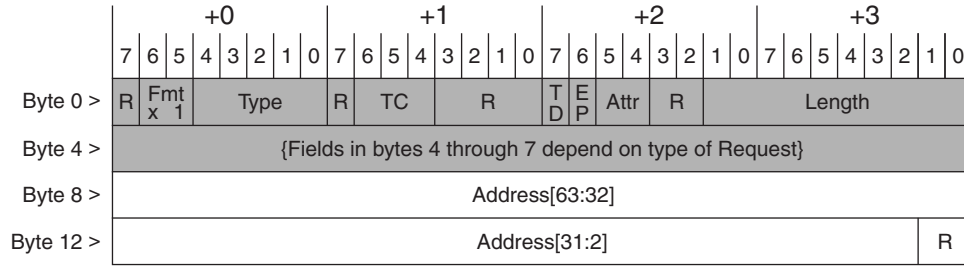
Figure 2-5: 64-bit Address Routing
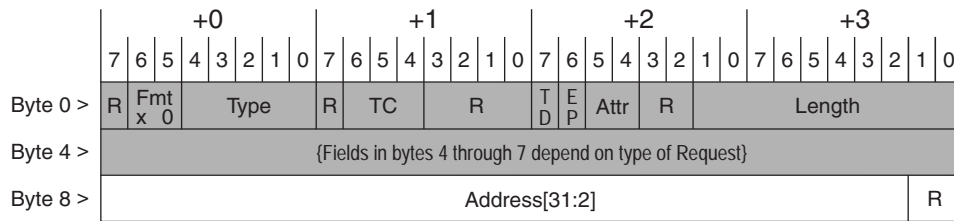
Figure 2-6: 32-bit Address Routing

❏ Address mapping to the TLP header is shown in Table 2-5.

Table 2-5: Address Field Mapping

Address Bits	32-bit Addressing	64-bit Addressing
63:56	Not Applicable	Bits 7:0 of Byte 8
55:47	Not Applicable	Bits 7:0 of Byte 9
47:40	Not Applicable	Bits 7:0 of Byte 10
39:32	Not Applicable	Bits 7:0 of Byte 11
31:24	Bits 7:0 of Byte 8	Bits 7:0 of Byte 12
23:16	Bits 7:0 of Byte 9	Bits 7:0 of Byte 13
15:8	Bits 7:0 of Byte 10	Bits 7:0 of Byte 14
7:2	Bits 7:2 of Byte 11	Bits 7:2 of Byte 15

❏ Memory Read Requests and Memory Write Requests can use either format.

  • For Addresses below 4 GB, Requesters must use the 32-bit format.

❏ I/O Read Requests and I/O Write Requests use the 32-bit format.

❏ All PCI Express Agents must decode all address bits in the header - address aliasing is not allowed.

> 📝 **IMPLEMENTATION NOTE**

Prevention of Address Aliasing

For correct software operation, full address decoding is required even in systems where it may be known to the system hardware architect/designer that fewer than 64 bits of address are actually meaningful in the system.

2.2.4.2.   ID Based Routing Rules

❑ ID routing is used with Configuration Requests, optionally with Vendor_Defined Messages (see Section 2.2.8.6), and with Completions

❑ ID routing uses the Bus, Device, and Function Numbers to specify the destination Device for the TLP

   • Bus, Device, and Function Number to TLP header mapping is shown in Table 2-6

❑ Two ID routing formats are specified, one used with a 4 DW header (see Figure 2-7) and one used with a 3 DW header (see Figure 2-8).

   • Header field locations are the same for both formats, and are given in Table 2-6

Table 2-6: Header Field Locations for ID Routing

Field	Header Location
Bus Number[7:0]	Bits 7:0 of Byte 8
Device Number[4:0]	Bits 7:3 of Byte 9
Function Number[2:0]	Bits 2:0 of Byte 9
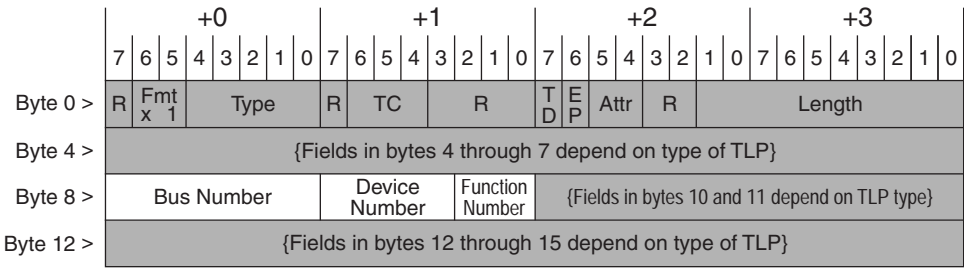
OM14542

Figure 2-7: ID Routing with 4 DW Header
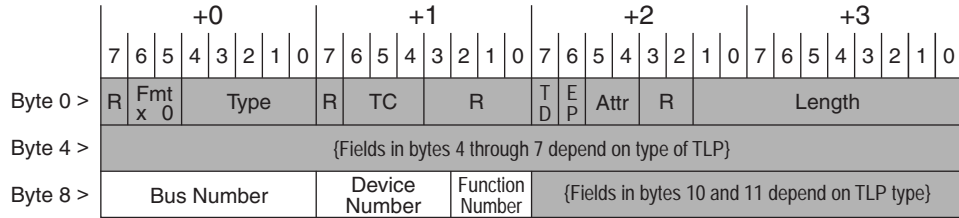
OM14541

Figure 2-8: ID Routing with 3 DW Header

2.2.5.   First/Last DW Byte Enables Rules

Byte Enables are included with Memory, I/O, and Configuration Requests. This section defines the corresponding rules. Byte Enables, when present in the Request header, are located in byte 7 of the header (see Figure 2-9).
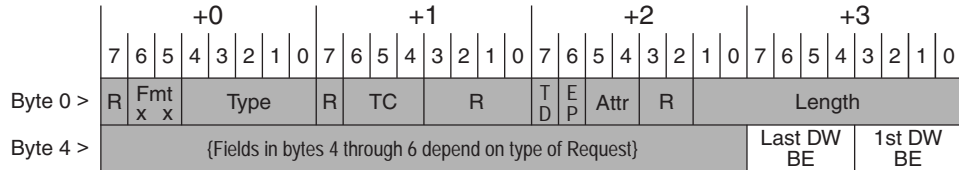
OM14545

Figure 2-9: Location of Byte Enables in TLP Header

5
❑   The 1st DW BE[3:0] field contains Byte Enables for the first (or only) DW referenced by a Request.

   • If the Length field for a Request indicates a length of greater than 1 DW, this field must not equal 0000b.

❑   The Last DW BE[3:0] field contains Byte Enables for the last DW of a Request.

10
   • If the Length field for a Request indicates a length of 1 DW, this field must equal 0000b.

   • If the Length field for a Request indicates a length of greater than 1 DW, this field must not equal 0000b.

❑   For each bit of the Byte Enables fields:

   • a value of 0b indicates that the corresponding byte of data must not be written or, if non-
15
prefetchable, must not be read at the Completer.

   • a value of 1b indicates that the corresponding byte of data must be written or read at the Completer.

❑   Non-contiguous Byte Enables (enabled bytes separated by non-enabled bytes) are permitted in the 1st DW BE field for all Requests with length of 1 DW.

20
   • Non-contiguous Byte Enable examples: 1010b, 0101b, 1001b, 1011b, 1101b

❑ Non-contiguous Byte Enables are permitted in both Byte Enables fields for QW aligned Memory Requests with length of 2 DW (1 QW).

❑ All Memory Requests with length of 3 DW or more must enable only bytes that are contiguous with the data between the first and last DW of the Request.

5 • Contiguous Byte Enables examples:
 1st DW BE: 1100, Last DW BE: 0011
 1st DW BE: 1000, Last DW BE: 0111

❑ Table 2-7 shows the correspondence between the bits of the Byte Enables fields, their location in the Request header, and the corresponding bytes of the referenced data.

Table 2-7: Byte Enables Location and Correspondence

Byte Enables	Header Location	Affected Data Byte[3]
1st DW BE[0]	Bit 0 of Byte 7	Byte 0
1st DW BE[1]	Bit 1 of Byte 7	Byte 1
1st DW BE[2]	Bit 2 of Byte 7	Byte 2
1st DW BE[3]	Bit 3 of Byte 7	Byte 3
Last DW BE[0]	Bit 4 of Byte 7	Byte N-4
Last DW BE[1]	Bit 5 of Byte 7	Byte N-3
Last DW BE[2]	Bit 6 of Byte 7	Byte N-2
Last DW BE[3]	Bit 7 of Byte 7	Byte N-1

10 ❑ A Write Request with a length of 1 DW with no bytes enabled is permitted, and has no effect at the Completer.

❑ If a Read Request of 1 DW specifies that no bytes are enabled to be read (1st DW BE[3:0] field = 0000b), the corresponding Completion must specify a Length of 1 DW, and include a data payload of 1 DW

15 • The contents of the data payload within the Completion packet is unspecified and may be any value

❑ Receiver/Completer behavior is undefined for a TLP violating the Byte Enables rules specified in this section.

❑ Receivers may optionally check for violations of the Byte Enables rules specified in this section.
20 If a Receiver implementing such checks determines that a TLP violates one or more Byte Enables rules, the TLP is a Malformed TLP

• If Byte Enables rules are checked, a violation is a reported error associated with the Receiving Port (see Section 6.2)

---

[3] Assuming the data referenced is N bytes in length (Byte 0 to Byte N-1). Note that last DW Byte Enables are used only if the data length is greater than one DW.

IMPLEMENTATION NOTE

Zero Length Read

A Memory Read Request of 1 DW with no bytes enabled, 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 flush semantic has wide application, and all Completers must implement the functionality associated with this semantic. Because a Requester may use the flush semantic without comprehending the characteristics of the Completer, Completers must ensure that zero length reads do not have side-effects. This is really just a specific case of the rule that in a non-prefetchable space, non-enabled bytes must not be read at the Completer. Note that the flush applies only to traffic in the same Traffic Class as the zero length Read.

2.2.6.   Transaction Descriptor

2.2.6.1.   Overview

The Transaction Descriptor is a mechanism for carrying Transaction information between the Requester and the Completer. Transaction Descriptors are composed of three fields:

❑ Transaction ID – identifies outstanding Transactions

❑ Attributes field – specifies characteristics of the Transaction

❑ Traffic Class (TC) field – associates Transaction with type of required service

Figure 2-10 shows the fields of the Transaction Descriptor. Note that these fields are shown together to highlight their relationship as parts of a single logical entity. The fields are not contiguous in the packet header.
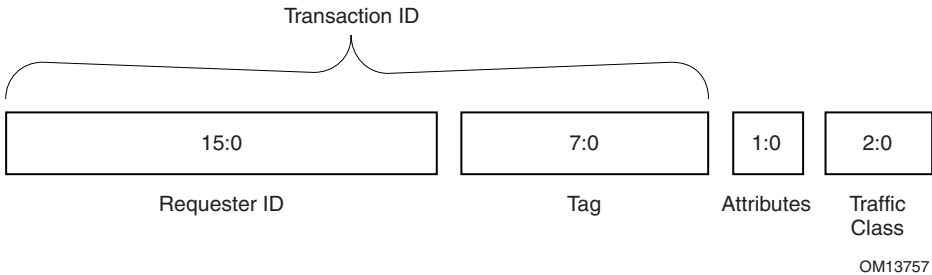
Figure 2-10: Transaction Descriptor

2.2.6.2. Transaction Descriptor – Transaction ID Field

The Transaction ID field consists of two major sub-fields: Requester ID and Tag as shown in Figure 2-11.
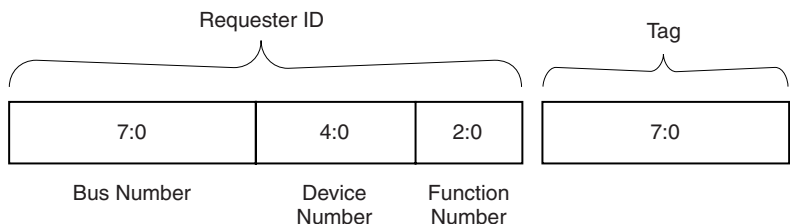
OM13758

Figure 2-11: Transaction ID

❑ Tag[7:0] is a 8-bit field generated by each Requestor, and it must be unique for all outstanding Requests that require a Completion for that Requester

- By default, the maximum number of outstanding Requests per device/function shall be limited to 32, and only the lower 5 bits of the Tag field are used with the remaining upper 3 bits required to be all 0's

- If the Extended Tag Field Enable bit (see Section 7.8.4) is set, the maximum is increased to 256, and the entire Tag field is used

- Receiver/Completer behavior is undefined if multiple Requests are issued non-unique Tag values

- If Phantom Function Numbers are used to extend the number of outstanding requests, the combination of the Phantom Function Number and the Tag field must be unique for all outstanding Requests that require a Completion for that Requester.

❑ For Requests which do not require Completion (Posted Requests), the value in the Tag[7:0] field is undefined and may contain any value

- For Posted Requests, the value in the Tag[7:0] field must not affect Receiver processing of the Request

❑ Requester ID and Tag combined form a global identifier, i.e., Transaction ID for each Transaction within a Hierarchy.

❑ Transaction ID is included with all Requests and Completions.

❑ The Requester ID is a 16-bit value that is unique for every PCI Express function within a Hierarchy.

❑ Functions must capture the Bus and Device Numbers supplied with all Configuration Write Requests (Type 0) completed by the function and supply these numbers in the Bus and Device Number fields of the Requester ID for all Requests initiated by the device/function.

₅　Exception: The assignment of Bus and Device Numbers to the logical devices within a Root Complex may be done in an implementation specific way.

Note that the Bus Number and Device Number may be changed at run time, and so it is necessary to re-capture this information with each and every Configuration Write Request.

Example: When a device (or function of a multi-function device) receives a Type 0 Configuration Write Request, the device comprehends that it is the intended recipient of the ₁₀ Request because it is a Type 0 Request. The routing information fields of the Write Request include the recipient's Bus Number and Device Number values (Figure 2-16). These values are captured by the device and used to generate the Requester and Completer ID field.

❑ Prior to the initial Configuration Write to a device, the device is not permitted to initiate Non-Posted Requests.

₁₅　● Exception: Logical devices within a Root Complex are permitted to initiate Requests prior to software-initiated configuration for accesses to system boot device(s).

Note that this rule and the exception are consistent with the existing PCI model for system initialization and configuration.

❑ Each function associated with a logical device must be designed to respond to a unique Function ₂₀ Number for Configuration Requests addressing that logical device.
Note: Each logical device may contain up to eight logical functions.

❑ A Switch must forward Requests without modifying the Transaction ID

❑ In some circumstances, a PCI Express-PCI Bridge is required to generate Transaction IDs for requests it forwards from a PCI or PCI-X bus.

₂₅ ## IMPLEMENTATION NOTE

Increasing the Number of Outstanding Requests

To increase the maximum possible number of outstanding Requests requiring Completion beyond 256, a device may, if the Phantom Function Number Enable bit is set (see Section 7.8.3), use Function Numbers not assigned to implemented functions to logically extend the Tag identifier. ₃₀ For a single function device, this can allow up to an 8-fold increase in the maximum number of outstanding Requests

Unclaimed function numbers are referred to as Phantom Function Numbers (PFN).

2.2.6.3. Transaction Descriptor – Attributes Field

The Attributes field is used to provide additional information that allows modification of the default handling of Transactions. These modifications apply to different aspects of handling the Transactions within the system, such as:

5 ❑ Ordering

❑ Hardware coherency management (snoop)

Note that attributes are hints that allow for optimizations in the handling of traffic. Level of support is dependent on target applications of particular PCI Express peripherals and platform building blocks. Refer to PCI-X 2.0 for additional details regarding these attributes.
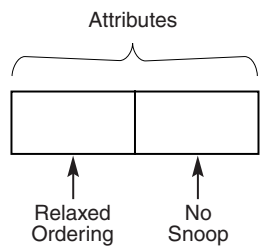
OM13759

Figure 2-12: Attributes Field of Transaction Descriptor

10 ## 2.2.6.4. Relaxed Ordering Attribute

Table 2-8 defines the states of the Relaxed Ordering attribute field. This attribute is discussed in Section 2.4.

Table 2-8: Ordering Attributes

Ordering Attribute	Ordering Type	Ordering Model
0	Default Ordering	PCI Strongly Ordered Model
1	Relaxed Ordering	PCI-X Relaxed Ordering Model

This attribute is not applicable and must be set to 0 for configuration requests, I/O requests, memory requests that are Message Signaled Interrupts, and Message requests (except where
15 specifically permitted).

2.2.6.5. No Snoop Attribute

Table 2-9 defines the states of the No Snoop attribute field. Note that the No Snoop attribute does not alter Transaction ordering.

Table 2-9: Cache Coherency Management Attribute

No Snoop Attribute	Cache Coherency Management Type	Coherency Model
0	Default	Hardware enforced cache coherency expected
1	No Snoop	Hardware enforced cache coherency not expected

This attribute is not applicable and must be set to 0 for configuration requests, I/O requests, memory requests that are Message Signaled Interrupts, and Message requests (except where specifically permitted).

2.2.6.6.   Transaction Descriptor – Traffic Class Field

The Traffic Class (TC) is a 3-bit field that allows differentiation of transactions into eight traffic classes.

Together with the PCI Express Virtual Channel support, the TC mechanism is a fundamental element for enabling differentiated traffic servicing. Every PCI Express Transaction Layer Packet uses TC information as an invariant label that is carried end to end within the PCI Express fabric. As the packet traverses across the fabric, this information is used at every Link and within each Switch element to make decisions with regards to proper servicing of the traffic. A key aspect of servicing is the routing of the packets based on their TC labels through corresponding Virtual Channels. Section 2.4.2 covers the details of the VC mechanism.

Table 2-10 defines the TC encodings.

Table 2-10: Definition of TC Field Encodings

TC Field Value	Definition
000	TC0: Best Effort service class (General Purpose I/O)
	(Default TC – must be supported by every PCI Express device)
001 – 111	TC1-TC7: Differentiated service classes
	(Differentiation based on Weighted-Round-Robin and/or Priority)

It is up to the system software to determine TC labeling and TC/VC mapping in order to provide differentiated services that meet target platform requirements.

The concept of Traffic Class applies only within the PCI Express interconnect fabric. Specific requirements of how PCI Express TC service policies are translated into policies on non-PCI

Express interconnects or within Root Complex or Endpoints is outside of the scope of this specification.

2.2.7.   Memory, I/O, and Configuration Request Rules

The following rule applies to all Memory, I/O and Configuration Requests. Additional rules specific to each type of Request follow.

❑ All Memory, I/O and Configuration Requests include the following fields in addition to the common header fields:

- Requester ID[15:0] and Tag[7:0], forming the Transaction ID

- Last DW BE[3:0] and 1st DW BE[3:0]

For Memory Requests, the following rules apply:

❑ Memory Requests route by address, using either 64-bit or 32-bit Addressing (see Figure 2-13 and Figure 2-14)

❑ For Memory Read Requests, Length must not exceed the value specified by Max_Read_Request_Size (see Section 7.8.4)

❑ Requests must not specify an Address/Length combination which causes a Memory Space access to cross a 4-KB boundary.

- Receivers may optionally check for violations of this rule. If a Receiver implementing this check determines that a TLP violates this rule, the TLP is a Malformed TLP

  ♦ If checked, this is a reported error associated with the Receiving Port (see Section 6.2)
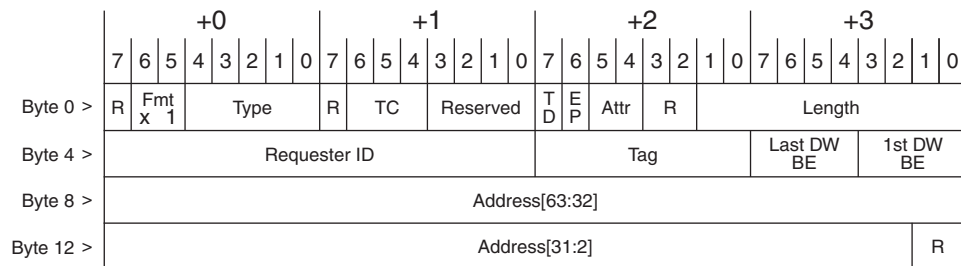
OM13764

Figure 2-13: Request Header Format for 64-bit Addressing of Memory
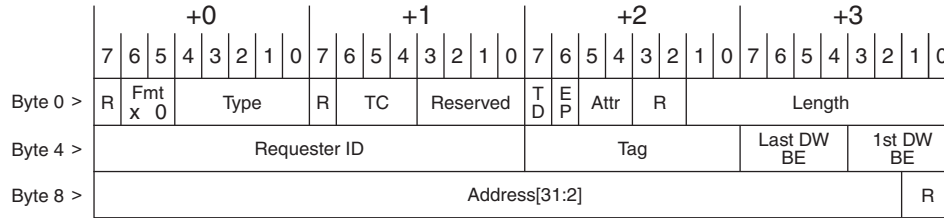
OM13763

Figure 2-14: Request Header Format for 32-bit Addressing of Memory

📝 IMPLEMENTATION NOTE

Generation of 64-bit Addresses

It is strongly recommended that PCI Express Endpoints be capable of generating the full range of 64-bit addresses. However, if a PCI Express Endpoint supports a smaller address range, and is unable to reach the full address range required by a given platform environment, the corresponding Device Driver must ensure that all Memory Transaction target buffers fall within the address range supported by the Endpoint. The exact means of ensuring this is platform and operating system specific, and beyond the scope of this specification.

For I/O Requests, the following rules apply:

❑ I/O Requests route by address, using 32-bit Addressing (see Figure 2-15)

❑ I/O Requests have the following restrictions:

- TC[2:0] must be 000b

- Attr[1:0] must be 00b

- Length[9:0] must be 00 0000 0001b

- Last DW BE[3:0] must be 0000b
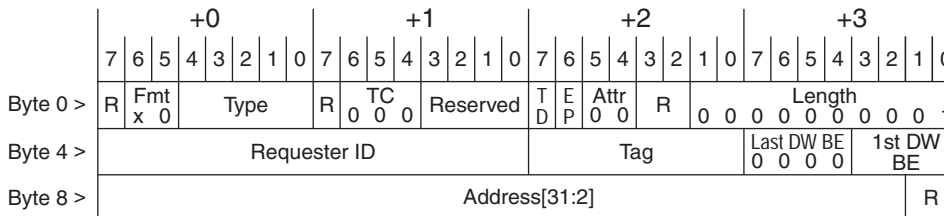
OM13765

Figure 2-15: Request Header Format for I/O Transactions

For Configuration Requests, the following rules apply:

❑ Configuration Requests route by ID, and use a 3 DW header

❑ In addition to the header fields included in all Memory, I/O, and Configuration Requests and the ID routing fields, Configuration Requests contain the following additional fields (see Figure 2-16)

- Register Number[5:0]

- Extended Register Number[3:0]

❑ Configuration Requests have the following restrictions:

- TC[2:0] must be 000b

- Attr[1:0] must be 00b

- Length[9:0] must be 00 0000 0001b

- Last DW BE[3:0] must be 0000b
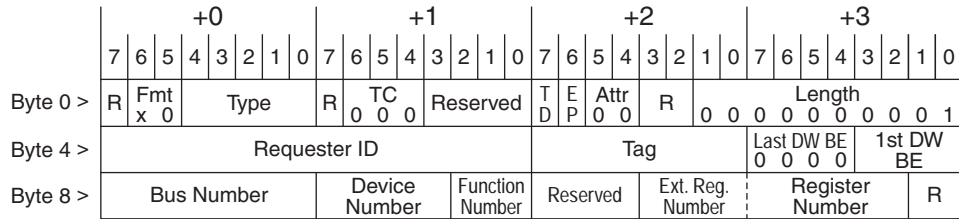
OM13766

Figure 2-16: Request Header Format for Configuration Transactions

Message Signaled Interrupts (MSIs) use Memory Write Requests to represent interrupt Messages (see Section 6.1.4). The Request format used for MSIs is identical to the Memory Write Request format defined above, and MSI Requests are indistinguishable from memory writes with regard to ordering, Flow Control, and data integrity.

2.2.8. Message Request Rules

This document defines the following groups of Messages:

❑ INTx Interrupt Signaling

❑ Power Management

❑ Error Signaling

❑ Locked Transaction Support

❑ Slot Power Limit Support

❑ Vendor-Defined Messages

❑ Hot-Plug Signaling

The following rules apply to all Message Requests. Additional rules specific to each type of Message follow.

❑ All Message Requests include the following fields in addition to the common header fields (see Figure 2-17):

- Requester ID[15:0] and Tag[7:0], forming the Transaction ID.

- Message Code[7:0] – Specifies the particular Message embodied in the Request.

❑ All Message Requests use the Msg Type field encoding, except for the Vendor_Defined Messages, which can use either Msg or MsgD, and the Set_Slot_Power_Limit Message, which uses MsgD.

❑ The Message Code field must be fully decoded (Message aliasing is not permitted).

❑ Except as noted, the Attr[1:0] field is reserved.

❑ Message Requests are posted and do not require Completion.

❑ Message Requests follow the same ordering rules as Memory Write Requests.
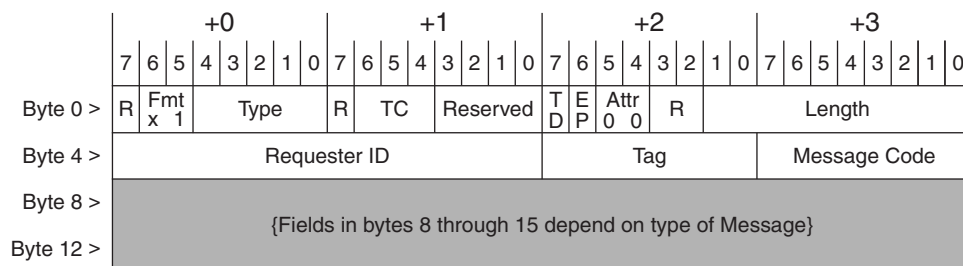
OM14539

Figure 2-17: Message Request Header

In addition to address and ID routing, Messages support several other routing mechanisms. These mechanisms are referred to as "implicit" because no address or ID specifies the destination device, but rather the destination is implied by the routing type. The following rules cover Message routing mechanisms:

5   ❑ Message routing is determined using the r[2:0] sub-field of the Type field

     • Message Routing r[2:0] values are defined in Table 2-11

     • Permitted values are defined in the following sections for each Message

Table 2-11: Message Routing

r[2:0]	Description
000	Routed to Root Complex
001	Routed by Address[4]
010	Routed by ID[5]
011	Broadcast from Root Complex
100	Local - Terminate at Receiver
101	Gathered and routed to Root Complex[6]
110-111	Reserved - Terminate at Receiver

2.2.8.1. INTx Interrupt Signaling - Rules

The Message Signaled Interrupt (MSI) mechanism is the preferred interrupt signaling mechanism in
10 PCI Express (see Section 6.1). However, in some systems, there may be devices which cannot support the MSI mechanism. The INTx virtual wire interrupt signaling mechanism is used to support legacy Endpoints and PCI Express/PCI(-X) Bridges in cases where the MSI mechanism cannot be used. Switches must support this mechanism. The following rules apply to the INTx interrupt signaling mechanism:

15   ❑ The INTx mechanism uses eight distinct Messages (see Table 2-12)

   ❑ Assert_INTx/Deassert_INTx Messages do not include a data payload (TLP Type is Msg).

   ❑ The Length field is reserved.

   ❑ Assert_INTx/Deassert_INTx Messages are only issued by Upstream Ports

     • Receivers may optionally check for violations of this rule. If a Receiver implementing this
20   check determines that an Assert_INTx/Deassert_INTx violates this rule, it must handle the TLP as a Malformed TLP.

       ♦ This is a reported error associated with the Receiving Port (see Section 6.2)

---

[4] Note that no Messages defined in this document use Address routing.

[5] See Section 2.2.4.

[6] This routing type is used only for PME_TO_Ack, and is described in Section 5.3.3.2.1.

❑ Assert_INTx and Deassert_INTx interrupt Messages must use the default Traffic Class designator (TC0). Receivers must check for violations of this rule. If a Receiver determines that a TLP violates this rule, it must handle the TLP as a Malformed TLP.

- This is a reported error associated with the Receiving Port (see Section 6.2)

Table 2-12: INTx Mechanism Messages

Name	Code[7:0]	Routing r[2:0]	Support[7]				Req ID[8]	Description/Comments
			RC	Ep	Sw	Br		
Assert_INTA	0010 0000	100	All:				BD	Assert INTA virtual wire. Note: These Messages are used for PCI 2.3 compatible INTx emulation
			r		tr			
			As Required:					
				t		t		
Assert_INTB	0010 0001	100	All:				BD	Assert INTB virtual wire
			r		tr			
			As Required:					
				t		t		
Assert_INTC	0010 0010	100	All:				BD	Assert INTC virtual wire
			r		tr			
			As Required:					
				t		t		
Assert_INTD	0010 0011	100	All:				BD	Assert INTD virtual wire
			r		tr			
			As Required:					
				t		t		

---

[7] Abbreviations:
RC = Root Complex
Sw = Switch (only used with "Link" routing)
Ep = Endpoint
Br = PCI Express (primary) to PCI/PCI-X (secondary) Bridge

r = Supports as Receiver
t = Supports as Transmitter

Note that Switches must support passing Messages on all legal routing paths. Only Messages specifying Local (0100b) routing or a reserved field value are terminated locally at the Receiving Port on a Switch.

[8] The Requester ID includes sub-fields for Bus Number, Device Number, and Function Number. Some Messages are not associated with specific Functions in a component, and for such Messages this field is Reserved; this is shown in this column using a code. Some messages can be used in more than one context and, therefore, more than one code may be listed. The codes in this column are:

BD   = Bus Number and Device Number included; Function Number is Reserved
BDF   = Bus Number, Device Number, and Function Number are included

Name	Code[7:0]	Routing r[2:0]	Support				Req ID	Description/Comments
			R C	E p	S w	B r		
Deassert_INTA	0010 0100	100	All: r		tr		BD	De-assert INTA virtual wire
			As Required:	t		t		
Deassert_INTB	0010 0101	100	All: r		tr		BD	De-assert INTB virtual wire
			As Required:	t		t		
Deassert_INTC	0010 0110	100	All: r		tr		BD	De-assert INTC virtual wire
			As Required:	t		t		
Deassert_INTD	0010 0111	100	All: r		tr		BD	De-assert INTD virtual wire
			As Required:	t		t		

The Assert_INTx/Deassert_INTx Message pairs constitute four "virtual wires" for each of the legacy PCI interrupts designated A, B, C, and D. The following rules describe the operation of these virtual wires:

❑ The components at both ends of each Link must track the logical state of the four virtual wires using the Assert/Deassert Messages to represent the active and inactive transitions (respectively) of each corresponding virtual wire.

- An Assert_INTx represents the active going transition of the INTx (x = A, B, C, or D) virtual wire

- A Deassert_INTx represents the inactive going transition of the INTx (x = A, B, C, or D) virtual wire

❑ When the local logical state of an INTx virtual wire changes at an Upstream Port, the Port must communicate this change in state to the Downstream Port on the other side of the same Link using the appropriate Assert_INTx or Deassert_INTx Message

Note: Duplicate Assert_INTx/Deassert_INTx Messages have no effect, but are not errors.

❑ INTx Interrupt Signaling is disabled when the Interrupt Disable bit of the Command register (see Section 7.5.1.1) is set to 1b.

- Any INTx virtual wires that are active when the Interrupt Disable bit is set must be deasserted by transmitting the appropriate Deassert_INTx Message(s)

❑ Virtual and actual PCI to PCI Bridges must map the virtual wires tracked on the secondary side of the Bridge according to the Device Number of the Device on the secondary side of the Bridge, as shown in Table 2-13

❑ Switches must track the state of the four virtual wires independently for each Downstream Port, and present a "collapsed" set of virtual wires on its Upstream Port

❑ If a Downstream Port goes to DL_Down status, the INTx virtual wires associated with that Port must be deasserted, and the Upstream Port virtual wire state updated accordingly.

• If this results in de-assertion of any Upstream INTx virtual wires, the appropriate Deassert_INTx Message(s) must be sent by the Upstream Port.

❑ The Root Complex must track the state of the four INTx virtual wires independently for each of its Downstream Ports, and map these virtual signals to system interrupt resources.

• Details of this mapping are system implementation specific.

❑ If a Downstream Port of the Root Complex goes to DL_Down status, the INTx virtual wires associated with that Port must be deasserted, and any associated system interrupt resource request(s) must be discarded.

Table 2-13: Bridge Mapping for INTx Virtual Wires

Device Number for Device on Secondary Side of Bridge (Interrupt Source)	INTx Virtual Wire on Secondary Side of Bridge	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

Note that the Requester ID of an Assert_INTx/Deassert_INTx Message will correspond to the Transmitter of the Message on that Link, and not necessarily to the original source of the interrupt.

📝 IMPLEMENTATION NOTE

System Interrupt Mapping

Note that system software (including BIOS and operating system) needs to comprehend the remapping of legacy interrupts (INTx mechanism) in the entire topology of the system (including hierarchically connected Switches and subordinate PCI Express/PCI Bridges) to establish proper correlation between PCI Express device interrupt and associated interrupt resources in the system interrupt controller. The remapping described by Table 2-13 is applied hierarchically at every Switch. In addition, PCI Express/PCI and PCI/PCI Bridges perform a similar mapping function.

2.2.8.2. *Power Management Messages*

These Messages are used to support PCI Express power management, which is described in detail in Chapter 5. The following rules define the Power Management Messages:

❑ Table 2-14 defines the Power Management Messages.

❑ Power Management Messages do not include a data payload (TLP Type is Msg).

❑ The Length field is reserved.

❑ Power Management Messages must use the default Traffic Class designator (TC0). Receivers must check for violations of this rule. If a Receiver determines that a TLP violates this rule, it must handle the TLP as a Malformed TLP.

- This is a reported error associated with the Receiving Port (see Section 6.2)

Table 2-14: Power Management Messages

Name	Code[7:0]	Routing r[2:0]	Support				Req ID	Description/Comments
			R C	E p	S w	B r		
PM_Active_State_Nak	0001 0100	100	t	r	tr	r	BD	Terminate at Receiver
PM_PME	0001 1000	000	All:				BDF	Sent Upstream by PME-requesting component. Propagates Upstream.
			r		tr	t		
			If PME supported:					
				t				
PME_Turn_Off	0001 1001	011	t	r		r	BDF	Broadcast Downstream
PME_TO_Ack	0001 1011	101	r	t		t	BDF	Sent Upstream by Endpoint. Sent Upstream by Switch when received on all Downstream Ports.
			(Note: Switch handling is special)					

2.2.8.3. *Error Signaling Messages*

Error Signaling Messages are used to signal errors that occur on specific transactions and errors that are not necessarily associated with a particular transaction (e.g., Link training fails). These Messages are initiated by the agent that detected the error.

5 ❑ Table 2-15 defines the Error Signaling Messages.

❑ Error Signaling Messages do not include a data payload (TLP Type is Msg).

❑ The Length field is reserved.

❑ Error Signaling Messages must use the default Traffic Class designator (TC0) Receivers must check for violations of this rule. If a Receiver determines that a TLP violates this rule, it must 10 handle the TLP as a Malformed TLP.

- This is a reported error associated with the Receiving Port (see Section 6.2)

Table 2-15: Error Signaling Messages

Name	Code[7:0]	Routing r[2:0]	Support				Req ID	Description/Comments
			R C	E p	S w	B r		
ERR_COR	0011 0000	000	r	t		t	BD BDF	This Message is issued when the component or device detects a correctable error on the PCI Express interface.
ERR_NONFATAL	0011 0001	000	r	t		t	BD BDF	This Message is issued when the component or device detects a Non-fatal, uncorrectable error on the PCI Express interface.
ERR_FATAL	0011 0011	000	r	t		t	BD BDF	This Message is issued when the component or device detects a Fatal, uncorrectable error on the PCI Express interface.

The initiator of the Message is identified with the Requester ID of the Message header. The Root Complex translates these error Messages into platform level events. Refer to Section 6.2 for details on uses for these Messages.

2.2.8.4. Locked Transactions Support

The Unlock Message is used to support Lock Transaction sequences. See Section 6.5 for details on Lock Transaction sequences. The following rules apply to the formation of the Unlock Message:

❑ Table 2-16 defines the Unlock Messages.

5 ❑ The Unlock Message does not include a data payload (TLP Type is Msg).

❑ The Length field is reserved.

❑ The Unlock Message must use the default Traffic Class designator (TC0) Receivers must check for violations of this rule. If a Receiver determines that a TLP violates this rule, it must handle the TLP as a Malformed TLP.

10 • This is a reported error associated with the Receiving Port (see Section 6.2)

Table 2-16 Unlock Message

Name	Code[7:0]	Routing r[2:0]	Support				Req ID	Description/Comments
			R C	E p	S w	B r		
Unlock	0000 0000	011	t	r		r	BD	Unlock Completer

2.2.8.5. Slot Power Limit Support

This Message is used to convey a slot power limitation value from a Downstream Port (of a Root Complex or a Switch) to an Upstream Port of a component (Endpoint, Switch, or a PCI Express-PCI Bridge) attached to the same Link.

15 ❑ Table 2-17 defines the Set_Slot_Power_Limit Message.

❑ The Set_Slot_Power_Limit Message includes a 1 DW data payload (TLP Type is MsgD).

❑ The Set_Slot_Power_Limit Message must use the default Traffic Class designator (TC0) Receivers must check for violations of this rule. If a Receiver determines that a TLP violates this rule, it must handle the TLP as a Malformed TLP.

20 • This is a reported error associated with the Receiving Port (see Section 6.2)

Table 2-17: Set_Slot_Power_Limit Message

Name	Code[7:0]	Routing r[2:0]	Support				Req ID	Description/Comments
			R C	E p	S w	B r		
Set_Slot_Power_Limit	0101 0000	100	t	r	tr	r	BDF	Set Slot Power Limit in Upstream Port

The Set_Slot_Power_Limit Message includes a one DW data payload. The data payload is copied from the Slot Capabilities register of the Downstream Port and is written into the Device Capabilities register of the Upstream Port on the other side of the Link. Bits 9:8 of the data payload

map to the Slot Power Limit Scale field and bits 7:0 map to the Slot Power Limit Value field. Bits 31:10 of the data payload must be set to all 0's by the Transmitter and ignored by the Receiver. This Message is sent automatically by the Downstream Port (of a Root Complex or a Switch) when one of the following events occurs:

5 ❑ On a Configuration Write to the Slot Capabilities register (see Section 7.8.9) when the Data Link Layer reports DL_Up status.

❑ Anytime when a Link transitions from a non-DL_Up status to a DL_Up status (see Section 2.9.2).

The component on the other side of the Link (Endpoint, Switch, or Bridge) that receives
10 Set_Slot_Power_Limit Message must copy the values in the data payload into the Device Capabilities register associated with the component's Upstream Port. PCI Express components that are targeted exclusively for integration on the system planar (e.g., system board) as well as components 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
15 defined in the corresponding electromechanical specification) are permitted to hardwire the value 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.

For more details on Power Limit control mechanism see Section 6.9.

2.2.8.6.   Vendor_Defined Messages

20 The Vendor_Defined Messages allow expansion of PCI Express messaging capabilities, either as a general extension to the PCI Express specification or a vendor-specific extension. Such extensions are not covered specifically in this document, although future revisions of this specification may use this mechanism to define new Messages (see below). This section defines the rules associated with these Messages generically.

25 ❑ The Vendor_Defined Messages (see Table 2-18) use the header format shown in Figure 2-18.

- If the Route by ID routing is used, bytes 8 and 9 form a 16-bit field for the destination ID

♦ otherwise these bytes are Reserved

- Bytes 10 and 11 form a 16-bit field for the Vendor ID, as defined by PCI-SIG, of the vendor defining the Message

30 - Bytes 12 through 15 are available for vendor definition

Table 2-18: Vendor_Defined Messages

Name	Code[7:0]	Routing r[2:0]	Support				Req ID	Description/Comments
			R C	E p	S w	B r		
Vendor_Defined Type 0	0111 1110	000, 010, 011, 100	See note.				See note.	Triggers detection of UR by Receiver if not implemented.
Vendor_Defined Type 1	0111 1111	000, 010, 011, 100	See note.				See note.	Silently discarded by Receiver if not implemented.

Note: Implementation specific.
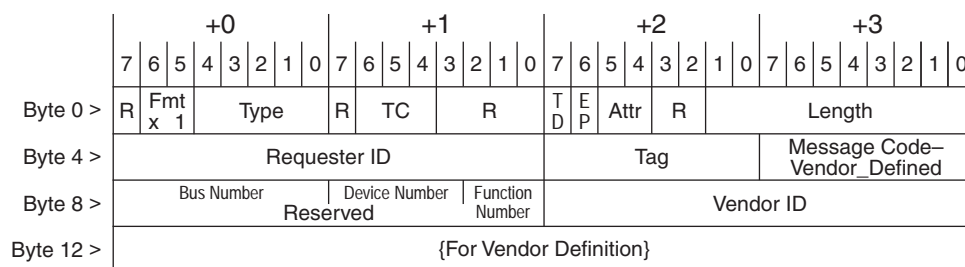
Figure 2-18: Header for Vendor-Defined Messages

❑ A data payload may be included with either type of Vendor_Defined Message (TLP type is Msg if no data payload is included and MsgD if a data payload is included)

❑ For both types of Vendor_Defined Messages, the Attr[1:0] field is not reserved.

❑ Messages defined by different vendors or by PCI-SIG are distinguished by the value in the Vendor ID field.

• The further differentiation of Messages defined by a particular vendor is beyond the scope of this document.

• Support for Messages defined by a particular vendor is implementation specific, and beyond the scope of this document.

❑ Receivers silently discard Vendor_Defined Type 1 Messages which they are not designed to receive – this is not an error condition.

❑ Receivers handle the receipt of an unsupported Vendor_Defined Type 0 Message as an Unsupported Request, and the error is reported according to Section 6.2.

2.2.8.7.   Hot-Plug Signaling Messages

The Hot-Plug Signaling Messages (Table 2-19) are virtual signals between Switches/Root Ports used to support Hot-Plug Event signaling and devices on cards that support Removal Request functionality (doorbell mechanism) on the card. For more information see Section 6.7.

Table 2-19: Hot-Plug Signaling Messages

Name	Code[7:0]	Routing r[2:0]	Support				Req ID	Description/Comments
			R C	E p	S w	B r		
Attention_Indicator_On	0100 0001	100	t	r	tr	r	BDF	This Message is issued by the Switch/Root Port when the Attention Indicator Control is set to 01b. The end device receiving the Message will terminate the Message and initiate appropriate action to cause the Attention Indicator located on the card to turn on. See note.
Attention_Indicator_Blink	0100 0011	100	t	r	tr	r	BDF	This Message is issued by the Switch/Root Port when the Attention Indicator Control is set to 10b. The end device receiving the Message will terminate the Message and initiate appropriate action to cause the Attention Indicator located on the card to blink. See note.
Attention_Indicator_Off	0100 0000	100	t	r	tr	r	BDF	This Message is issued by the Switch/Root Port when the Attention Indicator Control is set to 11b. The end device receiving the Message will terminate the Message and initiate appropriate action to cause the Attention Indicator located on the card to turn off. See note.

Name	Code[7:0]	Routing r[2:0]	Support				Req ID	Description/Comments
			RC	Ep	Sw	Br		
Power_Indicator_On	0100 0101	100	t	r	tr	r	BDF	This Message is issued by the Switch/Root Port when the Power Indicator Command is set to 01b. The end device receiving the Message will terminate the Message and initiate appropriate action to cause the Power Indicator located on the card to turn on. See note.
Power_Indicator_Blink	0100 0111	100	t	r	tr	r	BDF	This Message is issued by the Switch/Root Port when the Power Indicator Control is set to 10b. The end device receiving the Message will terminate the Message and initiate appropriate action to cause the Power Indicator located on the card to blink. See note.
Power_Indicator_Off	0100 0100	100	t	r	tr	r	BDF	This Message is issued by the Switch/Root Port when the Power Indicator Command is set to 11b. The end device receiving the Message will terminate the Message and initiate appropriate action to cause the Power Indicator located on the card to turn off. See note.
Attention_Button_Pressed	0100 1000	100	All: r		r		BDF	This Message is issued by a device in a slot that implements an Attention Button on the card to signal the Switch/Root Port to generate the Attention Button Pressed Event. The Switch Switch/Root Port terminates the Message and sets the Attention Button Pressed register to 1b which may result in an interrupt being generated.
			As Required:	t	t	t		

Note: If no indicators are present on the card, the Message is ignored.

Components must accept received Hot-Plug Signaling Messages as defined in the Support column of Table 2-19 without indicating an error. However, only components which implement or support the associated action must process the received Message and take the corresponding action. Components which do not implement or support the associated action discard the received
5 Message.

2.2.9. Completion Rules

All Read Requests and Non-Posted Write Requests require Completion. Completions include a Completion header that, for some types of Completions, will be followed by some number of DW of data. The rules for each of the fields of the Completion header are defined in the following
10 sections.

❑ Completions route by ID, and use a 3 DW header

- Note that the routing ID fields correspond directly to the Requester ID supplied with the corresponding Request. Thus for Completions these fields will be referred to collectively as the Requester ID instead of the distinct fields used generically for ID routing.

15 ❑ In addition to the header fields included in all TLPs and the ID routing fields, Completions contain the following additional fields (see Figure 2-19)

- Completer ID[15:0] – Identifies the Completer – described in detail below

- Completion Status[2:0] – Indicates the status for a Completion (see Table 2-20)

  ♦ Rules for determining the value in the Completion Status[2:0] field are in Section 2.3.1

20 - BCM – Byte Count Modified – this bit must not set by PCI Express Completers, and may only be set by PCI-X completers

- Byte Count[11:0] – The remaining byte count for Request

  ♦ The Byte Count value is specified as a binary number, with 0000 0000 0001b indicating 1 byte, 1111 1111 1111b indicating 4095 bytes, and 0000 0000 0000b indicating 4096 bytes

25 ♦ For Memory Read Completions, Byte Count[11:0] is set according to the rules in Section 2.3.1.1.

  ♦ For all other types of Completions, the Byte Count field must be 4.

- Tag[7:0] – in combination with the Requester ID field, corresponds to the Transaction ID

- Lower Address[6:0] – lower byte address for starting byte of Completion

30 ♦ For Memory Read Completions, the value in this field is the byte address for the first enabled byte of data returned with the Completion (see the rules in Section 2.3.1.1)

  ♦ This field is set to all 0's for all types of Completions other than Memory Read Completions

	+0	+1	+2	+3
	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Byte 0 >	R Fmt Type x 0	R TC Reserved	T E D P Attr R	Length
Byte 4 >	Completer ID		Compl. B Status C M	Byte Count
Byte 8 >	Requester ID		Tag	R Lower Address
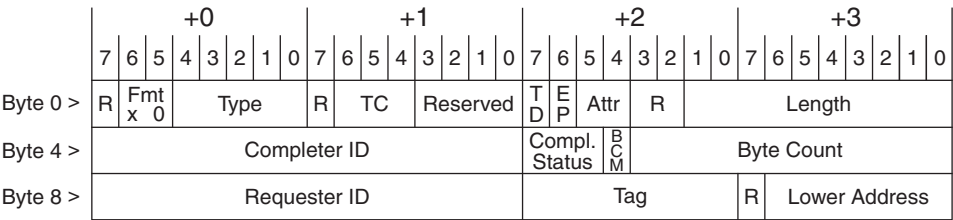
OM13769

Figure 2-19: Completion Header Format

Table 2-20: Completion Status Field Values

Completion Status[2:0] Field Value	Completion Status
000b	Successful Completion (SC)
001b	Unsupported Request (UR)
010b	Configuration Request Retry Status (CRS)
100b	Completer Abort (CA)
all others	Reserved

❑ The Completer ID[15:0] is a 16-bit value that is unique for every PCI Express function within a Hierarchy (see Figure 2-20)

Completer ID

7:0	4:0	2:0
Bus Number	Device Number	Function Number
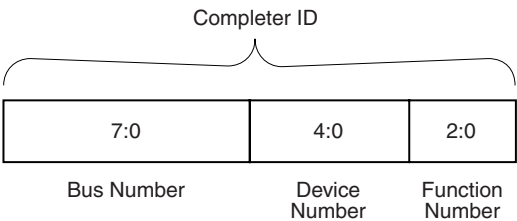
OM13770

Figure 2-20: Completer ID

❑ Functions must capture the Bus and Device Numbers supplied with all Type 0 Configuration Write Requests completed by the function, and supply these numbers in the Bus and Device Number fields of the Completer ID for all Completions generated by the device/function.

- If a function must generate a Completion prior to the initial device Configuration Write Request, 0's must be entered into the Bus Number and Device Number fields

- Note that Bus Number and Device Number may be changed at run time, and so it is necessary to re-capture this information with each and every Configuration Write Request.

- Exception: The assignment of bus numbers to the logical devices within a Root Complex may be done in an implementation specific way.

❑ In some cases, a Completion with the UR status may be generated by a multi-function device without associating the Completion with a specific function within the device – in this case, the Function Number field is Reserved.

- Example: A multi-function device receives a Read Request which does not target any resource associated with any of the functions of the device – the device generates a Completion with UR status and sets a value of all 0's in the Function Number field of the Completer ID

❑ Completion headers must supply the same values for the Requester ID, Tag, Attribute and Traffic Class as were supplied in the header of the corresponding Request.

❑ The Completion ID field is not meaningful prior to the software initialization and configuration of the completing device (using at least one Configuration Write Request), and the Requestor must ignore the value returned in the Completer ID field.

❑ A Completion including data must specify the actual amount of data returned in that Completion, and must include the amount of data specified.

- It is a TLP formation error to include more or less data than specified in the Length field, and the resulting TLP is a malformed TLP.

Note: This is simply a specific case of the general rule requiring TLP data payload length match the value in the Length field.

2.3. Handling of Received TLPs

This section describes how all Received TLPs are handled when they are delivered to the Receive Transaction Layer from the Receive Data Link Layer, after the Data Link Layer has validated the integrity of the received TLP. The rules are diagramed in the flowchart shown in Figure 2-21.

❑ Values in Reserved fields must be ignored by the Receiver.

❑ All Received TLPs which fail the required (and implemented optional) checks of TLP formation rules described in this chapter, or which use undefined Type field values, are Malformed TLPs (MP) and must be discarded without updating Receiver Flow Control information

- This is a reported error associated with the Receiving Port (see Section 6.2)

❑ Otherwise, update Receiver Flow Control tracking information (see Section 2.6)

❑ If the value in the Type field indicates the TLP is a Request, handle according to Request Handling Rules, otherwise, the TLP is a Completion – handle according to Completion Handling Rules (following sections)
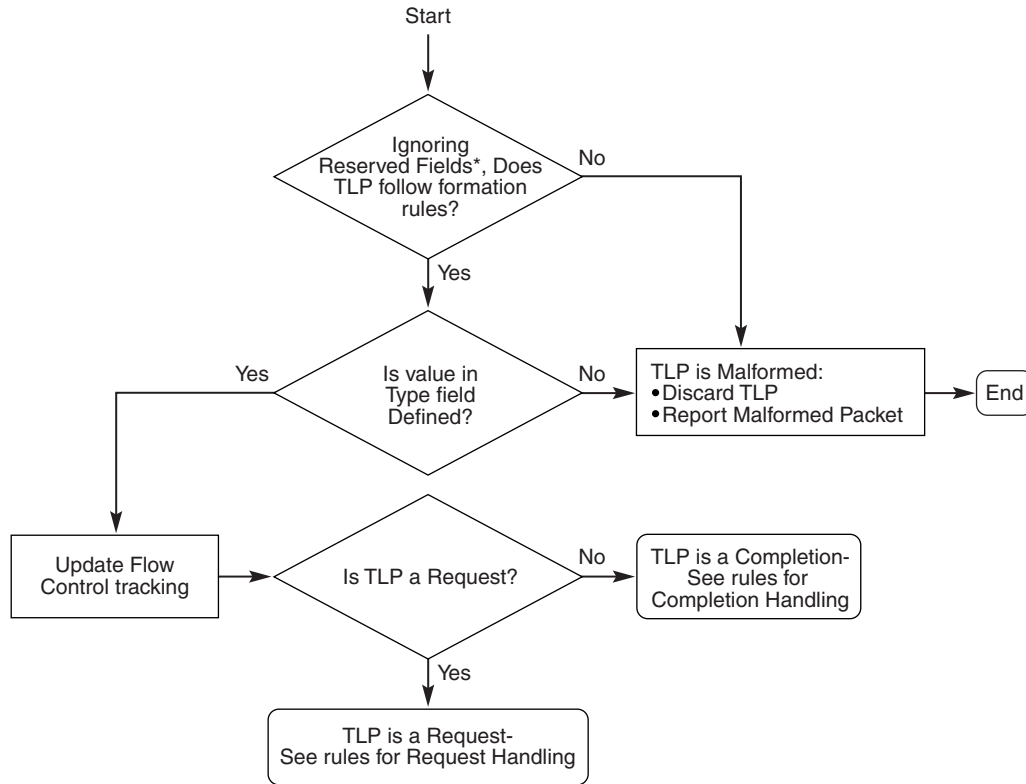
*TLP Header fields which are marked Reserved are not checked at the Receiver

OM13771

Figure 2-21: Flowchart for Handling of Received TLPs

Switches must process both TLPs which address resources within the Switch as well as TLPs which address resources residing outside the Switch. Switches handle all TLPs which address internal resources of the Switch according to the rules above. TLPs which pass through the Switch, or which address the Switch as well as passing through it, are handled according to the following rules (see Figure 2-22):

❑ If the value in the Type field indicates the TLP is not a Msg or MsgD Request, the TLP must be routed according to the routing mechanism used (see Sections 2.4.1 and 2.2.4.2)

❑ Switches route Completions using the information in the Requester ID field of the Completion.

❑ If the value in the Type field indicates the TLP is a Msg or MsgD Request, route the Request according to the routing mechanism indicated in the r[2:0] sub-field of the Type field

• If the value in r[2:0] indicates the Msg/MsgD is routed to the Root Complex (000b), the Switch must route the Msg/MsgD to the Upstream Port of the Switch

♦ It is an error to receive a Msg/MsgD Request specifying 000b routing at the Upstream Port of a Switch. Switches may check for violations of this rule – TLPs in violation are Malformed TLPs. If checked, this is a reported error associated with the Receiving Port (see Section 6.2)

• If the value in r[2:0] indicates the Msg/MsgD is routed by address (001b), the Switch must route the Msg/MsgD in the same way it would route a Memory Request by address

77

- If the value in r[2:0] indicates the Msg/MsgD is routed by ID (010b), the Switch must route the Msg/MsgD in the same way it would route a Completion by ID

- If the value in r[2:0] indicates the Msg/MsgD is a broadcast from the Root Complex (011b), the Switch must route the Msg/MsgD to all Downstream Ports of the Switch

    ♦ It is an error to receive a Msg/MsgD Request specifying 011b routing at the Downstream Port of a Switch. Switches may check for violations of this rule – TLPs in violation are Malformed TLPs. If checked, this is a reported error associated with the Receiving Port (see Section 6.2)

- If the value in r[2:0] indicates the Msg/MsgD terminates at the Receiver (100b or a reserved value), or if the Message Code field value is defined and corresponds to a Message which must be comprehended by the Switch, the Switch must process the Message according to the Message processing rules
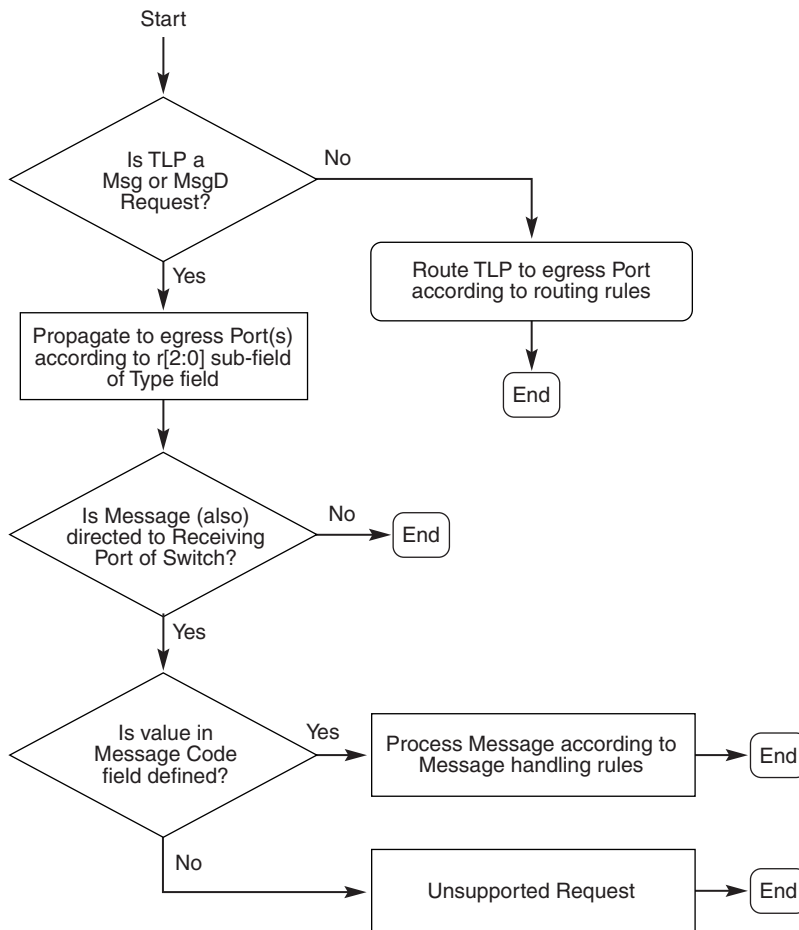
Figure 2-22: Flowchart for Switch Handling of TLPs

2.3.1. Request Handling Rules

This section describes how Received Requests are handled, following the initial processing done with all TLPs. The rules are diagramed in the flowchart shown in Figure 2-23.

❑ If the Request Type is not supported by the device, the Request is an Unsupported Request, and
⁵   is reported according to Section 6.2

   • If the Request requires Completion, a Completion Status of UR is returned (see Section 2.2.9)

❑ If the Request is a Message, and the Message Code specifies an undefined or unsupported value (other than Vendor_Defined Type 1 – see Section 2.2.8.6), the Request is an Unsupported
¹⁰   Request, and is reported according to Section 6.2

   • If the Message Code is a supported value, process the Message according to the corresponding Message processing rules

If the Request is not a Message, and is a supported Type, specific implementations may be optimized based on a defined programming model which ensures that certain types of (otherwise
¹⁵ legal) Requests will never occur. Such implementations may take advantage of the following rule:

❑ If the Request violates the programming model of the device, the device may optionally treat the Request as a Completer Abort, instead of handling the Request normally

   • If the Request is treated as a Completer Abort, this is a reported error associated with the device/function (see Section 6.2)

²⁰   • If the Request requires Completion, a Completion Status of CA is returned (see Section 2.2.9)

📝 IMPLEMENTATION NOTE

Optimizations Based on Restricted Programming Model

When a device's programming model restricts (vs. what is otherwise permitted in PCI Express) the
²⁵ characteristics of a Request, that device is permitted to "Completer Abort" any Requests which violate the programming model. Examples include unaligned or wrong-size access to a register block and unsupported size of request to a memory space.

Generally, devices are able to assume a restricted programming model when all communication will be between the device's driver software and the device itself. Devices which may be accessed
³⁰ directly by operating system software or by applications which may not comprehend the restricted programming model of the device (typically devices which implement legacy capabilities) should be designed to support all types of Requests which are possible in the existing usage model for the device. If this is not done, the device may fail to operate with existing software.

❑ Otherwise (supported Request Type, not a Message), process the Request

- If the Completer is permanently unable to process the Request due to a device-specific error condition the Completer must, if possible, handle the Request as a Completer Abort

  ♦ This is a reported error associated with the Receiving device/function, if the error can be isolated to a specific device/function in the component, or to the Receiving Port if the error cannot be isolated (see Section 6.2)

- For Configuration Requests only, following reset it is possible for a device to terminate the request but indicate that it is temporarily unable to process the Request, but will be able to process the Request in the future – in this case, the Configuration Request Retry Status Completion Status is used (see Section 6.6)

- In the process of servicing the Request, the Completer may determine that the (otherwise acceptable) Request must be handled as an error, in which case the Request is handled according to the type of the error

  ♦ Example: A PCI Express/PCI Bridge may initially accept a Request because it specifies a memory range mapped to the secondary side of the Bridge, but the Request may Master Abort or Target Abort on the PCI side of the Bridge. From the PCI Express perspective, the status of the Request in this case is UR (for Master Abort) or CA (for Target Abort). If the Request requires Completion on PCI Express, the corresponding Completion Status is returned.

❑ If the Request is a type which requires a Completion to be returned, generate a Completion according to the rules for Completion Formation (see Section 2.2.9)

- The Completion Status is determined by the result of handling the Request
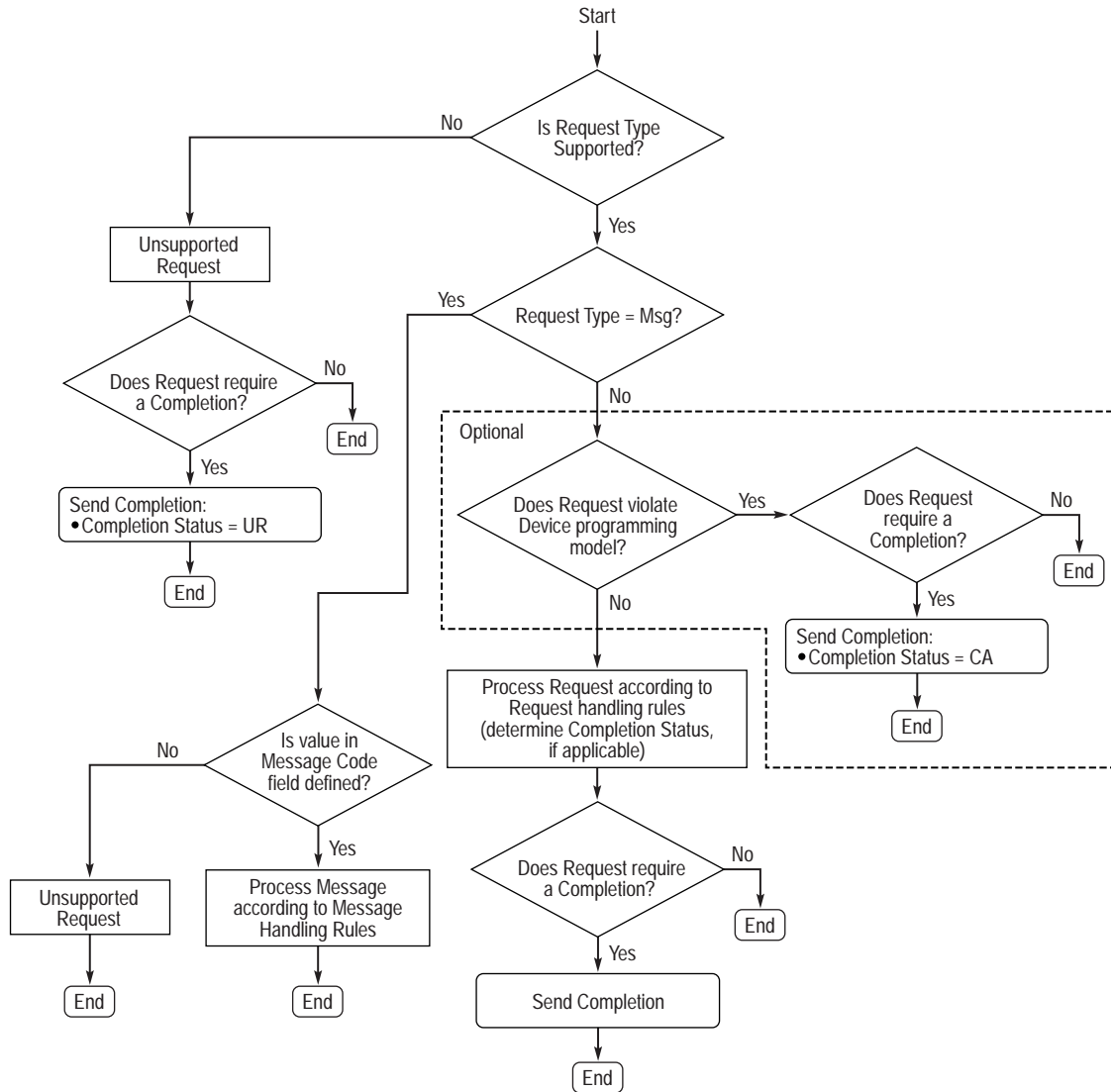
OM13773

Figure 2-23: Flowchart for Handling of Received Request

🖊 IMPLEMENTATION NOTE

Configuration Request Retry Status

Some devices require a lengthy self-initialization sequence to complete before they are able to service Configuration Requests (common with intelligent I/O solutions on PCI). PCI/PCI-X architecture has specified a 2^{25} (PCI) or 2^{26} (PCI-X) clock "recovery time" following reset to provide the required self-initialization time for such devices. PCI Express "softens" the need for this time based recovery period by implementing a Configuration Request Retry Completion Status (CRS). A device in receipt of a Configuration Request may respond with a Configuration Request Retry Completion Status to terminate the Request, and thus effectively stall the Configuration Request until such time that the subsystem has completed local initialization and is ready to communicate with the host. Note that it is only legal to respond with a configuration retry completion status in response to a Configuration Request. Sending this Completion Status in response to any other Request type will result in the generation of an error condition (Malformed TLP – see Section 6.2).

Receipt by the Requestor of a Completion with Configuration Request Retry Status terminates the Configuration Request on PCI Express. Further action by the Root Complex regarding the original Configuration Request is implementation specific, except during the period immediately following system reset (see Section 6.6). A Root Complex may choose to re-issue the Configuration Request as a new Request on PCI Express or complete the Request to the host as a failed transaction. Root Complex implementations may further choose to only allow a fixed number of Configuration Request/Retry Completion Status loops before determining that something is wrong with the target of the Request and taking appropriate action.

When used in systems including PCI Express to PCI/PCI-X Bridges, the Root Complex must comprehend the limit T_{rhfa} for PCI/PCI-X agents. Similarly, systems using PCI Express components which require additional self initialization time beyond the minimum guaranteed must provide some mechanism for re-issuing Configuration Requests terminated with CRS status. In systems running legacy PCI/PCI-X based software, the Root Complex must re-issue the Configuration Request using a hardware mechanism to ensure proper enumeration of the system.

See Section 6.6 for more information on reset.

2.3.1.1. Data Return for Read Requests

❑ Individual Completions for Memory Read Requests may provide less than the full amount of data Requested so long as all Completions for a given Request when combined return exactly the amount of data Requested in the Read Request.

₅　　• Completions for different Requests cannot be combined.

　　• I/O and Configuration Reads must be completed with exactly one Completion.

　　• The Completion Status for a Completion corresponds only to the status associated with the data returned with that Completion

₁₀　　♦ A Completion with status other than Successful Completion terminates the Completions for a single Read Request

　　　■ In this case, the value in the Length field is undefined, and must be ignored by the Receiver

❑ Completions must not include more data than permitted by the Max_Payload_Size parameter.

　　• Receivers must check for violations of this rule – TLPs in violation are Malformed TLPs

₁₅　　♦ This is a reported error associated with the Receiving Port (see Section 6.2)

　　Note: This is simply a special case of the rules which apply to all TLPs with data payloads

❑ Memory Read Requests may be completed with one, or in some cases, multiple Completions

❑ The Read Completion Boundary (RCB) parameter determines the naturally aligned address boundaries on which a Read Request may be serviced with multiple Completions

₂₀　　• For a Root Complex, RCB is 64 btyes or 128 bytes

　　♦ This value is reported through a configuration register (see Section 7.8)

　　Note: Bridges and Endpoints may implement a corresponding command bit which may be set by system software to indicate the RCB value for the Root Complex, allowing the Bridge/Endpoint to optimize its behavior when the Root Complex's RCB is 128 bytes.

₂₅　　• For all other system elements, RCB is 128 bytes

❑ Completions for Requests which do not cross the naturally aligned address boundaries at integer multiples of RCB bytes must include all data specified in the Request

❑ Requests which do cross the address boundaries at integer multiples of RCB bytes may be completed using more than one Completion, but the data must not be fragmented except along the address boundaries.

- The first Completion must start with the address specified in the Request, and must end at one of the following:

  ♦ the address specified in the Request plus the length specified by the Request (i.e., the entire Request)

  ♦ an address boundary between the start and end of the Request at an integer multiple of RCB bytes

- The final Completion must end with the address specified in the Request plus the length specified by the Request

- All Completions between, but not including, the first and final Completions must be an integer multiple of RCB bytes in length

❑ Receivers may optionally check for violations of RCB. If a Receiver implementing this check determines that a Completion violates this rule, it must handle the Completion as a Malformed TLP

- This is a reported error associated with the Receiving Port (see Section 6.2)

❑ Multiple Memory Read Completions for a single Read Request must return data in increasing address order.

❑ For each Memory Read Completion, the Byte Count field must indicate the remaining number of bytes required to complete the Request including the number of bytes returned with the Completion, except when the BCM field is 1b.[9]

- The total number of bytes required to complete a Memory Read Request is calculated as shown in Table 2-21

- If a Memory Read Request is completed using multiple Completions, the Byte Count value for each successive Completion is the value indicated by the preceding Completion minus the number of bytes returned with the preceding Completion.

---

[9] Only PCI-X completers set the BCM bit to 1b.

🖑 IMPLEMENTATION NOTE

BCM Bit Usage

To satisfy certain PCI-X protocol constraints, a PCI-X Bridge or PCI-X Completer for a PCI-X burst read in some cases will set the Byte Count field in the first PCI-X transaction of the Split Completion sequence to indicate the size of just that first transaction instead of the entire burst read. When this occurs, the PCI-X Bridge/PCI-X Completer will also set the BCM bit in that first PCI-X transaction, to indicate that the Byte Count field has been modified from its normal usage. Refer to the PCI-X 2.0 specification for further details.

A PCI Express Memory Read Requester needs to correctly handle the case when a PCI-X Bridge/PCI-X Completer sets the BCM bit. When this occurs, the first Read Completion packet returned to the Requestor will have the BCM bit set, indicating that the Byte Count field reports the size of just that first packet instead of the entire remaining byte count. The Requester should not conclude at this point that other packets of the Read Completion are missing.

The BCM bit will never be set in subsequent packets of the Read Completion, so the Byte Count field in those subsequent packets will always indicate the remaining byte count in each instance. Thus, the Requester can use the Byte Count field in these packets to determine if other packets of the Read Completion are missing.

PCI Express Completers will never set the BCM bit.

Table 2-21: Calculating Byte Count from Length and Byte Enables

1st DW BE[3:0]	Last DW BE[3:0]	Total Byte Count
1xx1	0000[10]	4
01x1	0000	3
1x10	0000	3
0011	0000	2
0110	0000	2
1100	0000	2
0001	0000	1
0010	0000	1
0100	0000	1
1000	0000	1
0000	0000	1
xxx1	1xxx	Length[11] * 4
xxx1	01xx	(Length * 4) - 1
xxx1	001x	(Length * 4) - 2
xxx1	0001	(Length * 4) - 3
xx10	1xxx	(Length * 4) - 1
xx10	01xx	(Length * 4) - 2
xx10	001x	(Length * 4) - 3
xx10	0001	(Length * 4) - 4
x100	1xxx	(Length * 4) - 2
x100	01xx	(Length * 4) - 3
x100	001x	(Length * 4) - 4
x100	0001	(Length * 4) - 5
1000	1xxx	(Length * 4) - 3
1000	01xx	(Length * 4) - 4
1000	001x	(Length * 4) - 5
1000	0001	(Length * 4) - 6

---

[10] Note that Last DW BE of 0000 is permitted only with a Length of 1 DW.

[11] Length is the number of DW as indicated by the value in the Length field, and is multiplied by 4 to yield a number in bytes.

❑ For all Memory Read Completions, the Lower Address field must indicate the lower bits of the byte address for the first enabled byte of data returned with the Completion

- For the first (or only) Completion, the Completer can generate this field from the least significant five bits of the address of the Request concatenated with two bits of byte-level address formed as shown in Table 2-22.

- For any subsequent Completions, the Lower Address field will always be zero except for Completions generated by a Root Complex with an RCB value of 64 bytes. In this case the least significant 6 bits of the Lower Address field will always be zero and the most significant bit of the Lower Address field will toggle according to the alignment of the 64-byte data payload.

Table 2-22: Calculating Lower Address from 1st DW BE

1st DW BE[3:0]	Lower Address[1:0]
0000	00b
xxx1	00b
xx10	01b
x100	10b
1000	11b

❑ When a Read Completion is generated with a Completion Status other than Successful Completion:

- No data is included with the Completion
  - ♦ The Cpl (or CplLk) encoding is used instead of CplD (or CplDLk)

- This Completion is the final Completion for the Request.
  - ♦ The Completer must not transmit additional Completions for this Request.
    - ■ Example: Completer split the Request into four parts for servicing; the second Completion had a Completer Abort Completion Status; the Completer terminated servicing for the Request, and did not Transmit the remaining two Completions.

- The Byte Count field must indicate the remaining number of bytes that would be required to complete the Request (as if the Completion Status were Successful Completion)

- The Lower Address field must indicate the lower bits of the byte address for the first enabled byte of data that would have been returned with the Completion if the Completion Status were Successful Completion

📝 IMPLEMENTATION NOTE

Restricted Programming Model

When a device's programming model restricts (vs. what is otherwise permitted in PCI Express) the size and/or alignment of Read Requests directed to the device, that device is permitted to use a Completer Abort Completion Status for Read Requests which violate the programming model. An implication of this is that such devices, generally devices where all communication will be between the device's driver software and the device itself, need not necessarily implement the buffering required to generate Completions of length RCB. However, in all cases, the boundaries specified by RCB must be respected for all reads which the device will Complete with Successful Completion status.

Examples:

1. Memory Read Request with Address of 1 0000h and Length of C0h bytes (192 decimal) could be completed by a Root Complex with an RCB value of 64 bytes with one of the following combinations of Completions (bytes):
192 –or– 128, 64 –or– 64, 128 –or– 64, 64, 64

2. Memory Read Request with Address of 10000h and Length of C0h bytes (192 decimal) could be completed by a Root Complex with an RCB value of 128 bytes in one of the following combinations of Completions (bytes):
192 –or– 128, 64

3. Memory Read Request with Address of 10020h and Length of 100h bytes (256 decimal) could be completed by a Root Complex with an RCB value of 64 bytes in one of the following combinations of Completions (bytes):
256 –or–
32, 224 –or– 32, 64, 160 –or– 32, 64, 64, 96 –or– 32, 64, 64, 64, 32 –or–
32, 64, 128, 32 –or– 32, 128, 96 –or– 32, 128, 64, 32 –or–
96, 160 –or– 96, 128, 32 –or– 96, 64, 96 –or– 96, 64, 64, 32 –or–
160, 96 –or– 160, 64, 32 –or– 224, 32

4. Memory Read Request with Address of 10020h and Length of 100h bytes (256 decimal) could be completed by an Endpoint in one of the following combinations of Completions (bytes):
256 –or– 96, 160 –or– 96, 128, 32 –or– 224, 32

2.3.2. Completion Handling Rules

❑ When a device receives Completion which does not correspond to any of the outstanding Requests issued by that device, the Completion is called an "Unexpected Completion."

❑ Receipt of an Unexpected Completion is an error and must be handled according to the following rules:

- The Agent receiving an Unexpected Completion must discard the Completion.

- An Unexpected Completion is a reported error associated with the Receiving Port (see Section 6.2)

Note: Unexpected Completions are assumed to occur mainly due to Switch misrouting of the Completion. The Requester of the Request may not receive a Completion for its Request in this case, and the Requester's Completion Timeout mechanism (see Section 2.8) will terminate the Request.

❑ Completions with a Completion Status other than Successful Completion, or Configuration Request Retry Status (in response to Configuration Request only) must cause the Requester to:

- Free Completion buffer space and other resources associated with the Request.

- Report the error according to the rules in Section 6.2.

❑ Root Complex handling of a Completion with Configuration Request Retry Status for a Configuration Request is implementation specific, except for the period following system reset (see Section 6.6).

❑ Completions with a Configuration Request Retry Status in response to a Request other than a Configuration Request are Malformed TLPs

- This is a reported error associated with the Receiving Port (see Section 6.2)

❑ Completions with a Reserved Completion Status value are treated as if the Completion Status was Unsupported Request (UR)

- This is a reported error associated with the Receiving device/function, normally the same as the Requestor (see Section 6.2)

❑ When a Read Completion is received with a Completion Status other than Successful Completion:

- No data is included with the Completion
    - ♦ The Cpl (or CplLk) encoding is used instead of CplD (CplDLk)

5
- This Completion is the final Completion for the Request.
    - ♦ The Requester must consider the Request terminated, and not expect additional Completions.
        - ■ Handling of partial Completions Received earlier is implementation specific.

Example: The Requester received 32 bytes of Read data for a 128-byte Read Request it
10 had issued, then a Completion with the Completer Abort Completion Status. The Requester then must free the internal resources which had been allocated for that particular Read Request.

📎 IMPLEMENTATION NOTE

Read Data Values with UR Completion Status

15 Some system configuration software depends on reading a data value of all 1's when a Configuration Read Request is terminated as an Unsupported Request, particularly when probing to determine the existence of a device in the system. A Root Complex intended for use with software that depends on a read-data value of all 1's must synthesize this value when UR Completion Status is returned for a Configuration Read Request.

---

20 2.4.   Transaction Ordering

2.4.1.   Transaction Ordering Rules

Table 2-23 defines the ordering requirements for PCI Express Transactions. The rules defined in this table apply uniformly to all types of Transactions on PCI Express including Memory, I/O, Configuration, and Messages. The ordering rules defined in this table apply within a single Traffic
25 Class (TC). There is no ordering requirement among transactions with different TC labels. Note that this also implies that there is no ordering required between traffic that flows through different Virtual Channels since transactions with the same TC label are not allowed to be mapped to multiple VCs on any PCI Express Link.

For Table 2-23, 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 (row) must be allowed to pass the first (column) to avoid deadlock.
⁵ (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
¹⁰ support the Producer-Consumer strong ordering model.

Table 2-23: Ordering Rules Summary Table

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
Non-Posted Request	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
Completion	I/O or Configuration Write Completion (Row E)	Y/N	Yes	Yes	Y/N	Y/N

Explanation of the entries in Table 2-23:

A2 a   A Memory Write or Message Request with the Relaxed Ordering Attribute bit clear (0b) 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 (1b)
¹⁵ 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   Endpoints, Switches, and Root Complex may allow Memory Write and Message Requests to pass Completions or be blocked by Completions.

A5, A6 b   PCI Express to PCI Bridges and PCI Express to PCI-X Bridges, when operating PCI segment in conventional mode, must allow Memory Write and Message Requests to pass Completions traveling in the PCI Express to PCI direction (Primary side of Bridge to Secondary side of Bridge) 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, then a Read Completion cannot pass a previously enqueued Memory Write or Message Request.

D2 b   If the Relaxed Ordering attribute bit is set, then 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.

Additional Rules:

❑ PCI Express Switches are permitted to allow a Memory Write or Message Request with the Relaxed Ordering bit set to pass any previously posted Memory Write or Message Request moving in the same direction. Switches must forward the Relaxed Ordering attribute unmodified. The Root Complex is also permitted to allow data bytes within the Request to be written to system memory in any order. (The bytes must be written to the correct system memory locations. Only the order in which they are written is unspecified).

❑ For Root Complex and Switch, Memory Write combining (as defined in the PCI Specification) is prohibited.

❑ Note: This is required so that devices can be permitted to optimize their receive buffer and control logic for Memory Write sizes matching their natural expected sizes, rather than being required to support the maximum possible Memory Write payload size.

❑ Combining of Memory Read Requests, and/or Completions for different Requests is prohibited.

5 ❑ The No Snoop bit does not affect the required ordering behavior.

📝 IMPLEMENTATION NOTE

Large Memory Reads vs. Multiple Smaller Memory Reads

Note that the rule associated with entry D5b in Table 2-23 ensures that for a single Memory Read Request serviced with multiple Completions, the Completions will be returned in address order.
10 However, the rule associated with entry D5a permits that different Completions associated with distinct Memory Read Requests may be returned in a different order than the issue order for the Requests. For example, if a device issues a single Memory Read Request for 256 bytes from location 1000h, and the Request is returned using two Completions (see Section 2.3.1.1) of 128 bytes each, it is guaranteed that the two Completions will return in the following order:

15 1st Completion returned: Data from 1000h to 107Fh.

2nd Completion returned: Data from 1080h to 10FFh.

However, if the device issues two Memory Read Requests for 128 bytes each, first to location 1000h, then to location 1080h, the two Completions may return in either order:

1st Completion returned: Data from 1000h to 107Fh.

20 2nd Completion returned: Data from 1080h to 10FFh.

– or –

1st Completion returned: Data from 1080h to 10FFh.

2nd Completion returned: Data from 1000h to 107Fh.

2.4.2. Update Ordering and Granularity Observed by a Read Transaction

If a Requester using a single transaction reads a block of data from a Completer, and the Completer's data buffer is concurrently being updated, the ordering of multiple updates and granularity of each update reflected in the data returned by the read is outside the scope of this specification. This applies both to updates performed by PCI Express write transactions and
30 updates performed by other mechanisms such as host CPUs updating host memory.

If a Requester using a single transaction reads a block of data from a Completer, and the Completer's data buffer is concurrently being updated by one or more entities not on the PCI Express fabric, the ordering of multiple updates and granularity of each update reflected in the data returned by the read is outside the scope of this specification.

As an example of update ordering, assume that the block of data is in host memory, and a host CPU writes first to location A and then to a different location B. A Requester reading that data block with a single read transaction is not guaranteed to observe those updates in order. In other words, the Requester may observe an updated value in location B and an old value in location A, regardless of the placement of locations A and B within the data block. Unless a Completer makes its own guarantees (outside this specification) with respect to update ordering, a Requester that relies on update ordering must observe the update to location B via one read transaction before initiating a subsequent read to location A to return its updated value.

As an example of update granularity, if a host CPU writes a QWORD to host memory, a Requester reading that QWORD from host memory may observe a portion of the QWORD updated and another portion of it containing the old value.

While not required by this specification, it is strongly recommended that host platforms guarantee that when a host CPU writes aligned DWORDs or aligned QWORDs to host memory, the update granularity observed by a PCI Express read will not be smaller than a DWORD.

IMPLEMENTATION NOTE

No Ordering Required Between Cachelines

A Root Complex serving as a Completer to a single Memory Read that requests multiple cachelines from host memory is permitted to fetch multiple cachelines concurrently, to help facilitate multi-cacheline completions, subject to Max_Payload_Size. No ordering relationship between these cacheline fetches is required.

2.4.3. Update Ordering and Granularity Provided by a Write Transaction

If a single write transaction containing multiple DWORDs and the Relaxed Ordering bit clear is accepted by a Completer, the observed ordering of the updates to locations within the Completer's data buffer must be in increasing address order. This semantic is required in case a PCI or PCI-X Bridge along the path combines multiple write transactions into the single one. However, the observed granularity of the updates to the Completer's data buffer is outside the scope of this specification.

While not required by this specification, it is strongly recommended that host platforms guarantee that when a PCI Express write updates host memory, the update granularity observed by a host CPU will not be smaller than a DWORD.

As an example of update ordering and granularity, if a Requester writes a QWORD to host memory, in some cases a host CPU reading that QWORD from host memory could observe the first DWORD updated and the second DWORD containing the old value.

2.5.   Virtual Channel (VC) Mechanism

The PCI Express Virtual Channel (VC) mechanism provides support for carrying throughout the PCI Express fabric traffic that is differentiated using TC labels. The foundation of VCs are independent fabric resources (queues/buffers and associated control logic). These resources are used to move information across PCI Express Links with fully independent flow-control between different VCs. This is key to solving the problem of flow-control induced blocking where a single traffic flow may create a bottleneck for all traffic within the system.

Traffic is associated with VCs by mapping packets with particular TC labels to their corresponding VCs. The PCI Express VC mechanism allows flexible mapping of TCs onto the VCs. In the simplest form, TCs can be mapped to VCs on a 1:1 basis. To allow performance/cost tradeoffs, PCI Express provides the capability of mapping multiple TCs onto a single VC. Section 2.5.2 covers details of TC to VC mapping.

A Virtual Channel is established when one or multiple TCs are associated with physical VC resource designated by VC ID. This process is controlled by the PCI Express configuration software as described in Sections 6.3 and 7.11.

Support for TCs and VCs beyond default TC0/VC0 pair is optional. The association of TC0 with VC0 is fixed, i.e., "hardwired," and must be supported by all PCI Express components. Therefore the baseline TC/VC setup does not require any VC-specific hardware or software configuration. In order to ensure interoperability, PCI Express components that do not implement the optional PCI Express Virtual Channel Capability structure must obey the following rules:

❑ A Requester must only generate requests with TC0 label. (Note that if the Requester initiates requests with a TC label other than TC0, the requests may be treated as malformed by the component on the other side of the Link that implements the extended VC capability and applies TC filtering.)

❑ A Completer must accept requests with TC label other than TC0, and must preserve the TC label, i.e., any completion that it generates must have the same TC label as the label of the request.

❑ A Switch must map all TCs to VC0 and must forward all transactions regardless of the TC label.

A PCI Express Endpoint or Root Complex that intends to be a Requester that can issue requests with TC label other than TC0 must implement the PCI Express Virtual Channel Capability structure, even if it only supports the default VC. This is required in order to enable mapping of TCs beyond the default configuration. It must follow the TC/VC mapping rules according to the software programming of the VC Capability structure.

Figure 2-24 illustrates the concept of Virtual Channel. Conceptually, traffic that flows through VCs is multiplexed onto a common physical Link resource on the Transmit side and de-multiplexed into separate VC paths on the Receive side.
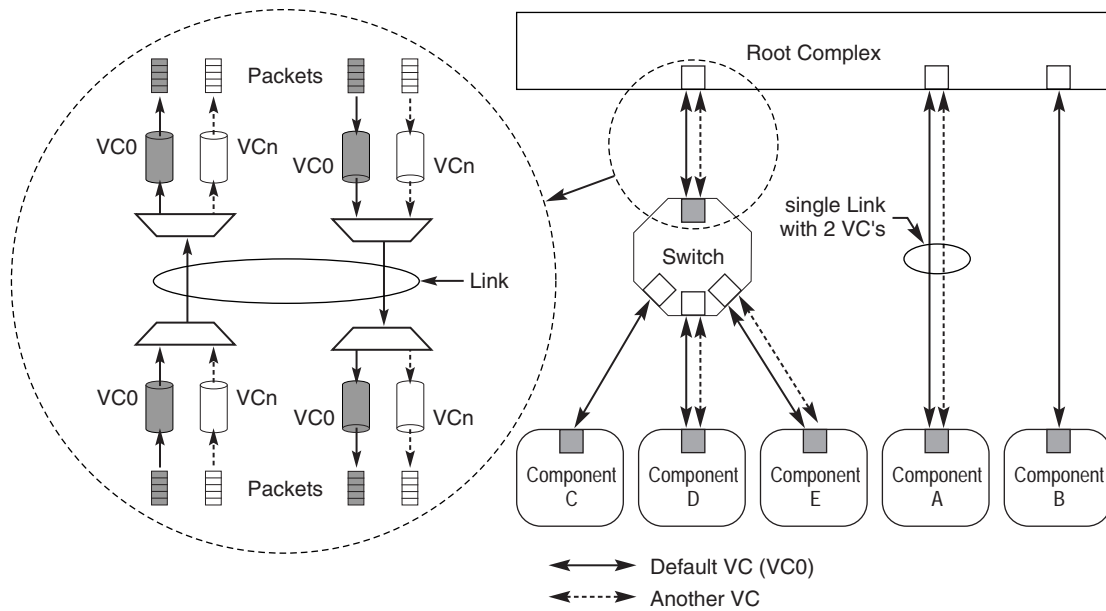
OM13760

Figure 2-24: Virtual Channel Concept – An Illustration

Internal to the Switch, every Virtual Channel requires dedicated physical resources (queues/buffers and control logic) that support independent traffic flows inside the Switch. Figure 2-25 shows conceptually the VC resources within the Switch (shown in Figure 2-24) that are required to support traffic flow in the upstream direction.
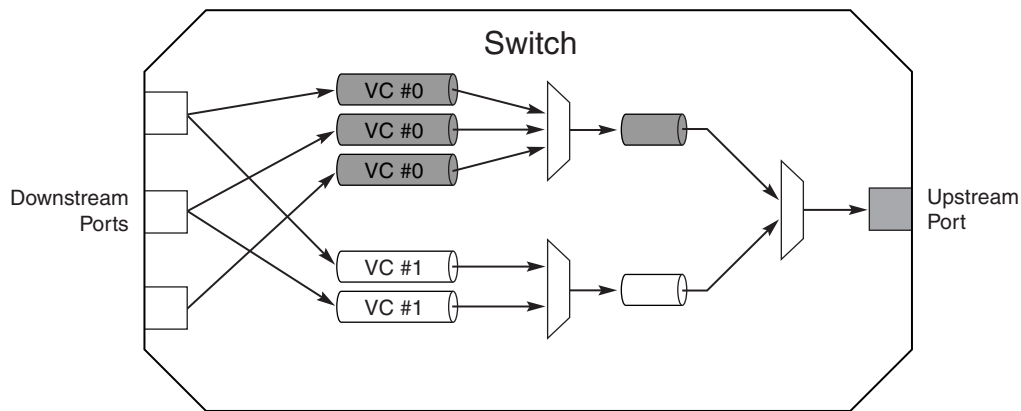
OM13761

Figure 2-25: Virtual Channel Concept – Switch Internals (Upstream Flow)

📝 IMPLEMENTATION NOTE

VC and VC Buffering Considerations

1. The amount of buffering beyond the architectural minimums per supported VC is implementation-specific.

2. Buffering beyond the architectural minimums is not required to be identical across all VCs on a given Link, i.e., an implementation may provide greater buffer depth for selected VCs as a function of implementation usage models and other Link attributes, e.g., Link width and signaling.

3. Implementations may adjust their buffering per VC based on implementation-specific policies derived from PCI Express configuration and VC enablement, e.g., if a four VC implementation has only two VCs enabled, the implementation may assign the non-enabled VC buffering to the enabled VCs to improve fabric efficiency/performance by reducing the probability of fabric backpressure due to Link-level flow control.

4. The number of VCs supported, and the associated buffering per VC per Port, are not required to be the same for all Ports of a multi-Port component (a Switch or Root Complex).

2.5.1. Virtual Channel Identification (VC ID)

Individual PCI Express Ports on a Root Complex, Switch, or Device can support 1-8 Virtual Channels – each Port is independently configured/managed therefore allowing implementations to vary the number of VCs supported per Port based on usage model-specific requirements. These VCs are uniquely identified using the Virtual Channel Identification (VC ID) mechanism.

Note that TLPs do not include VC ID information. The association of TLPs with VC ID for the purpose of Flow Control accounting is done at each Port of the Link using TC to VC mapping as discussed in Section 2.5.2.

All PCI Express Ports that support more than VC0 must provide the VC Capability structure according to the definition in Section 7.11. Providing this extended structure is optional for Ports that support only the default TC0/VC0 configuration. PCI Express configuration software is responsible for configuring Ports on both sides of the Link for a matching number of VCs. This is accomplished by scanning the PCI Express hierarchy and using VC Capability registers associated with Ports (that support more than default VC0) to establish number of VCs for the Link. Rules for assigning VC ID to VC hardware resources within a PCI Express Port are as follows:

❑ VC ID assignment must be unique per PCI Express Port – The same VC ID cannot be assigned to different VC hardware resources within the same Port.

❑ VC ID assignment must be the same (matching in the terms of numbers of VCs and their IDs) for the two PCI Express Ports on both sides of a PCI Express Link.

❑ VC ID 0 is assigned and fixed to the default VC.

2.5.2.   TC to VC Mapping

5   Every Traffic Class that is supported must be mapped to one of the Virtual Channels. The mapping of TC0 to VC0 is fixed.

The mapping of TCs other than TC0 is system software specific. However, the mapping algorithm must obey the following rules:

❑ One or multiple TCs can be mapped to a VC.

10   ❑ One TC must not be mapped to multiple VCs in any PCI Express Port.

❑ TC/VC mapping must be identical for PCI Express Ports on both sides of a PCI Express Link.

Table 2-24 provides an example of TC to VC mapping.

Table 2-24: TC to VC Mapping Example

Supported VC Configurations	TC/VC Mapping Options
VC0	TC(0-7)/VC0
VC0, VC1	TC(0-6)/VC0, TC7/VC1
VC0-VC3	TC(0-1)/VC0, TC(2-4)/VC1, TC(5-6)/VC2, TC7/VC3
VC0-VC7	TC[0:7]/VC[0:7]

Notes on conventions:

❑ TCn/VCk = TCn mapped to VCk

15   ❑ TC(n-m)/VCk = all TCs in the range n-m mapped to VCk (i.e., to the same VC)

❑ TC[n:m]/VC[n:m] = TCn/VCn, TCn_{+1}/ $VCn_{+1,}$..., TCm/VCm

Figure 2-26 provides a graphical illustration of TC to VC mapping in several different Link configurations. For additional considerations on TC/VC, refer to Section 6.3.
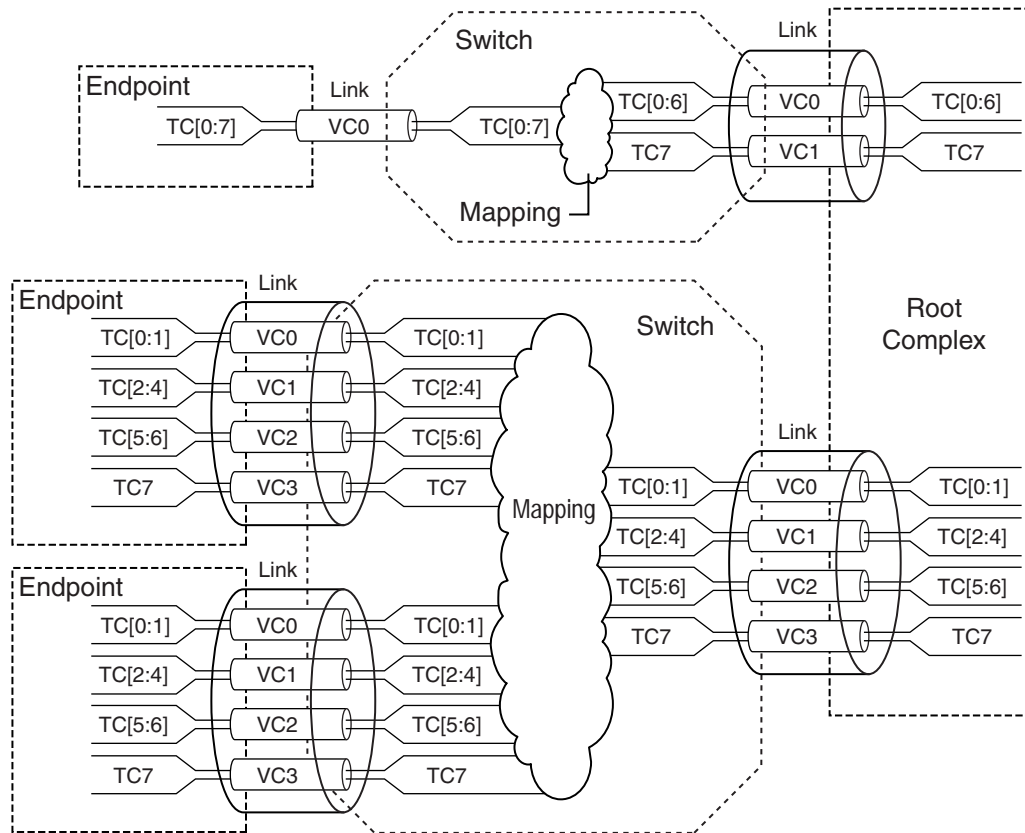
OM13762

Figure 2-26: An Example of TC/VC Configurations

2.5.3.   VC and TC Rules

Here is a summary of key rules associated with the TC/VC mechanism:

❑ All PCI Express devices must support the general purpose I/O Traffic Class, i.e., TC0 and must implement the default VC0.

5 ❑ Each Virtual Channel (VC) has independent Flow Control.

❑ There are no ordering relationships required between different TCs

❑ There are no ordering relationships required between different VCs

❑ A Switch's peer-to-peer capability applies to all Virtual Channels supported by the Switch.

❑ Transactions with a TC that is not mapped to any enabled VC in a PCI Express Ingress Port are
10 treated as malformed transactions by the receiving device.

❑ For Switches, transactions with TC that is not mapped to any of enabled VCs in the target Egress Port are treated as malformed TLPs.

❑ For a Root Port, transactions with a TC that is not mapped to any of enabled VCs in the target RCRB are treated as malformed TLPs.

❑ Switches must support independent TC/VC mapping configuration for each Port.

❑ Root Complex must support independent TC/VC mapping configuration for each RCRB and the associated Root Ports.

For more details on the VC and TC mechanisms, including configuration, mapping, and arbitration, refer to Chapter 6.3.

2.6. Ordering and Receive Buffer Flow Control

Flow Control (FC) is used to prevent overflow of Receiver buffers and to enable compliance with the ordering rules defined in Section 2.4. Note that the Flow Control mechanism is used by the Requester to track the queue/buffer space available in the Agent across the Link as shown in Figure 2-27. That is, Flow Control is point-to-point (across a Link) and not end-to-end. Flow Control does not imply that a Request has reached its ultimate Completer.
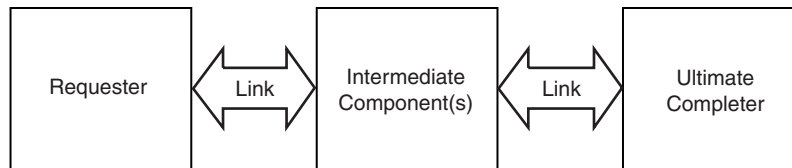
OM13776

Figure 2-27: Relationship Between Requester and Ultimate Completer

Flow Control is orthogonal to the data integrity mechanisms used to implement reliable information exchange between Transmitter and Receiver. Flow Control can treat the flow of TLP information from Transmitter to Receiver as perfect, since the data integrity mechanisms ensure that corrupted and lost TLPs are corrected through retransmission (see Section 3.5).

Each Virtual Channel maintains an independent Flow Control credit pool. The FC information is conveyed between two sides of the Link using DLLP packets. The VC ID field of the DLLP is used to carry the Virtual Channel Identification that is required for proper flow-control credit accounting.

Flow Control is handled by the Transaction Layer in cooperation with the Data Link Layer. The Transaction Layer performs Flow Control accounting functions for Received TLPs and "gates" TLP Transmissions based on available credits for transmission.

Note: Flow Control is a function of the Transaction Layer and, therefore, the following types of information transmitted on the interface are not associated with Flow Control Credits: LCRC, Packet Framing Symbols, other Special Symbols, and Data Link Layer to Data Link Layer inter-communication packets. An implication of this fact is that these types of information must be processed by the Receiver at the rate they arrive (except as explicitly noted in this specification).

Also, any TLPs transferred from the Transaction Layer to the Data Link and Physical Layers must have first passed the Flow Control "gate." Thus, both Transmit and Receive Flow Control mechanisms are unaware if the Data Link Layer transmits a TLP repeatedly due to errors on the Link.

2.6.1. Flow Control Rules

In this and other sections of this specification, rules are described using conceptual "registers" that a PCI Express device could use in order to implement a PCI Express compliant design. This description does not imply or require a particular implementation and is used only to clarify the requirements.

❏ Flow Control information is transferred using Flow Control Packets (FCPs), which are a type of DLLP (see Section 3.4)

❏ The unit of Flow Control credit is 4 DW for data

❏ For headers, the unit of Flow Control credit is one maximum-size header plus TLP digest

❏ Each Virtual Channel has independent Flow Control

❏ Flow Control distinguishes three types of TLPs (note relationship to ordering rules – see Section 2.4):

- Posted Requests (P) – Messages and Memory Writes

- Non-Posted Requests (NP) – All Reads, I/O, and Configuration Writes

- Completions (CPL) – Associated with corresponding NP Requests

❏ In addition, Flow Control distinguishes the following types of TLP information within each of the three types:

- Headers (H)

- Data (D)

❏ Thus, there are six types of information tracked by Flow Control for each Virtual Channel, as shown in Table 2-25.

Table 2-25: Flow Control Credit Types

Credit Type	Applies to This Type of TLP Information
PH	Posted Request headers
PD	Posted Request Data payload
NPH	Non-Posted Request headers
NPD	Non-Posted Request Data payload
CPLH	Completion headers
CPLD	Completion Data payload

❏ TLPs consume Flow Control credits as shown in Table 2-26.

Table 2-26: TLP Flow Control Credit Consumption

TLP	Credit Consumed[12]
Memory, I/O, Configuration Read Request	1 NPH unit
Memory Write Request	1 PH + n PD units[13]
I/O, Configuration Write Request	1 NPH + 1 NPD Note: size of data written is never more than 1 (aligned) DW
Message Requests without data	1 PH unit
Message Requests with data	1 PH + n PD units
Memory Read Completion	1 CPLH + n CPLD units
I/O, Configuration Read Completions	1 CPLH unit + 1 CPLD unit
I/O, Configuration Write Completions	1 CPLH unit

❑ Components must implement independent Flow Control for all Virtual Channels that are supported by that component.

❑ Flow Control is initialized autonomously by hardware only for the default Virtual Channel (VC0)

- VC0 is initialized when the Data Link Layer is in the DL_Init state following reset (see Sections 3.2 and 3.3)

❑ When other Virtual Channels are enabled by software, each newly enabled VC will follow the Flow Control initialization protocol (see Section 3.3)

- Software enables a Virtual Channel by setting the VC Enable bits for that Virtual Channel in both components on a Link (see Section 7.11)

Note: It is possible for multiple VCs to be following the Flow Control initialization protocol simultaneously – each follows the initialization protocol as an independent process

❑ Software disables a Virtual Channel by clearing the VC Enable bits for that Virtual Channel in both components on a Link

- Disabling a Virtual Channel for a component resets the Flow Control tracking mechanisms for that Virtual Channel in that component

❑ InitFC1 and InitFC2 FCPs are used only for Flow Control initialization (see Section 3.3)

❑ An InitFC1, InitFC2, or UpdateFC FCP that specifies a Virtual Channel that is disabled is discarded without effect

---

[12] Each header credit implies the ability to accept a TLP Digest along with the corresponding TLP.

[13] For all cases where "n" appears, n = Roundup(Length/FC unit size).

❑ During FC initialization for any Virtual Channel, including the default VC initialized as a part of Link initialization, Receivers must initially advertise VC credit values equal to or greater than those shown in Table 2-27.

- Components may optionally check for violations of this rule. If a component implementing this check determines a violation of this rule, the violation is a Flow Control Protocol Error (FCPE)

    ♦ If checked, this is a reported error associated with the Receiving Port (see Section 6.2)

Table 2-27: Minimum Initial Flow Control Advertisements[14]

Credit Type	Minimum Advertisement
PH	1 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 1024 bytes, the smallest permitted initial credit value would be 040h.
NPH	1 unit – credit value of 01h
NPD	1 unit – credit value of 01h
CPLH	Root Complex (supporting peer-to-peer traffic between all Root Ports) and Switch: 1 FC unit - credit value of 01h
	Root Complex (not supporting peer-to-peer traffic between all Root Ports) and Endpoint: infinite FC units - initial credit value of all 0s[15]
CPLD	Root Complex (supporting peer-to-peer traffic between all Root Ports) and Switch: 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 will ever generate, whichever is smaller.
	Root Complex (not supporting peer-to-peer traffic between all Root Ports) and Endpoint: infinite FC units - initial credit value of all 0s

❑ A Root Complex that does not support peer-to-peer traffic between all Root Ports must advertise infinite Completion credits.

❑ A Root Complex that supports peer-to-peer traffic between all Root Ports may optionally advertise non-infinite Completion credits. In this case, the Root Complex must ensure that deadlocks are avoided and forward progress is maintained for completions directed towards the Root Complex. Note that temporary stalls of completion traffic (due to a temporary lack of credit) are possible since Non-Posted requests forwarded by the RC may not have explicitly allocated completion buffer space.

---

[14] PCI Express to PCI/PCI-X Bridge requirements are addressed in the *PCI Express to PCI/PCI-X Bridge Specification, Rev. 1.0.*

[15] This value is interpreted as infinite by the Transmitter, which will, therefore, never throttle.

❑ A Receiver must never cumulatively issue more than 2048 outstanding unused credits to the Transmitter for data payload or 128 for header.

- Components may optionally check for violations of this rule. If a component implementing this check determines a violation of this rule, the violation is a Flow Control Protocol Error (FCPE).

  ♦ If checked, this is a reported error associated with the Receiving Port (see Section 6.2)

❑ If an Infinite Credit advertisement (value of 00h or 000h) has been made during initialization, no Flow Control updates are required following initialization.

- If UpdateFC DLLPs are sent, the credit value fields must be set to zero and must be ignored by the Receiver. The Receiver may optionally check for non-zero update values (in violation of this rule). If a component implementing this check determines a violation of this rule, the violation is a Flow Control Protocol Error (FCPE)

  ♦ If checked, this is a reported error associated with the Receiving Port (see Section 6.2)

❑ If only the Data or header advertisement (but not both) for a given type (N, NP, or CPL) has been made with infinite credits during initialization, the transmission of UpdateFC DLLPs is still required, but the credit field corresponding to the Data/header (advertised as infinite) must be set to zero and must be ignored by the Receiver.

- The Receiver may optionally check for non-zero update values (in violation of this rule). If a Receiver implementing this check determines a violation of this rule, the violation is a Flow Control Protocol Error (FCPE)

  ♦ If checked, this is a reported error associated with the Receiving Port (see Section 6.2)

❑ A TLP using an uninitialized VC is a Malformed TLP

- This is a reported error associated with the Receiving Port (see Section 6.2)

2.6.1.1. FC Information Tracked by Transmitter

❑ For each type of information tracked, there are two quantities tracked for Flow Control TLP Transmission gating:

- CREDITS_CONSUMED

  ♦ Count of the total number of FC units consumed by TLP Transmissions made since Flow Control initialization, modulo $2^{[Field\ Size]}$ (where [Field Size] is 8 for PH, NPH, and CPLH and 12 for PD, NPD and CPLD).

  ♦ Set to all 0's at interface initialization

♦ Updated for each TLP the Transaction Layer allows to pass the Flow Control gate for Transmission

■ Updated as shown:

$$CREDITS\_CONSUMED :=$$

$$(CREDITS\_CONSUMED + Increment) \bmod 2^{[Field\ Size]}$$

Where Increment is the size in FC credits of the corresponding part of the TLP passed through the gate, and [Field Size] is 8 for PH, NPH, and CPLH and 12 for PD, NPD, and CPLD

● CREDIT_LIMIT

♦ The most recent number of FC units legally advertised by the Receiver. This quantity represents the total number of FC credits made available by the Receiver since Flow Control initialization, , modulo $2^{[Field\ Size]}$ (where [Field Size] is 8 for PH, NPH, and CPLH and 12 for PD, NPD and CPLD).

♦ Undefined at interface initialization

♦ Set to the value indicated during Flow Control initialization

♦ For each FC update received,

■ if CREDIT_LIMIT is not equal to the update value, set CREDIT_LIMIT to update value

❑ The Transmitter gating function must determine if sufficient credits have been advertised to permit the transmission of a given TLP. If the Transmitter does not have enough credits to transmit the TLP, it must block the transmission of the TLP, possibly stalling other TLPs that are using the same Virtual Channel. The Transmitter must follow the ordering and deadlock avoidance rules specified in Section 2.4, which require that certain types of TLPs must bypass other specific types of TLPs when the latter are blocked. Note that TLPs using different Virtual Channels have no ordering relationship, and must not block each other.

❑ The Transmitter gating function test is performed as follows:

● For each required type of credit, the number of credits required is calculated as:

$$CUMULATIVE\_CREDITS\_REQUIRED =$$

$$(CREDITS\_CONSUMED +$$
$$<credit\ units\ required\ for\ pending\ TLP>) \bmod 2^{[Field\ Size]}$$

● Unless CREDIT_LIMIT was specified as "infinite" during Flow Control initialization, the Transmitter is permitted to Transmit a TLP if, for each type of information in the TLP, the following equation is satisfied (using unsigned arithmetic):

$$(CREDIT\_LIMIT -$$
$$CUMULATIVE\_CREDITS\_REQUIRED) \bmod 2^{[Field\ Size]}$$

$$<= 2^{[Field\ Size]} / 2$$

If CREDIT_LIMIT was specified as "infinite" during Flow Control initialization, then the gating function is unconditionally satisfied for that type of credit.

Note that some types of Transactions require more than one type of credit. (For example, Memory Write requests require PH and PD credits.)

❑ When accounting for credit use and return, information from different TLPs is never mixed within one credit.

❑ When some TLP is blocked from Transmission by a lack of FC Credit, Transmitters must follow the ordering rules specified in Section 2.4 when determining what types of TLPs must be permitted to bypass the stalled TLP.

❑ The return of FC credits for a Transaction must not be interpreted to mean that the Transaction has completed or achieved system visibility.

- Flow Control credit return is used for receive buffer management only, and Agents must not make any judgment about the Completion status or system visibility of a Transaction based on the return or lack of return of Flow Control information.

❑ When a Transmitter sends a nullified TLP (with inverted LCRC and using EDB as the end Symbol), the Transmitter does not modify CREDITS_CONSUMED for that TLP (see Section 3.5.2.1)

2.6.1.2.   FC Information Tracked by Receiver

❑ For each type of information tracked, the following quantities are tracked for Flow Control TLP Receiver accounting:

- CREDITS_ALLOCATED

  ♦ Count of the total number of credits granted to the Transmitter since initialization, modulo $2^{[Field\ Size]}$ (where [Field Size] is 8 for PH, NPH, and CPLH and 12 for PD, NPD and CPLD)

  ♦ Initially set according to the buffer size and allocation policies of the Receiver

  ♦ This value is included in the InitFC and UpdateFC DLLPs (see Section 3.4)

  ♦ Incremented as the Receiver Transaction Layer makes additional receive buffer space available by processing Received TLPs

    ■ Updated as shown:

    CREDITS_ALLOCATED :=

    (CREDITS_ALLOCATED + Increment) mod $2^{[Field\ Size]}$

    Where Increment corresponds to the credits made available, and [Field Size] is 8 for PH, NPH, and CPLH and 12 for PD, NPD, and CPLD

- CREDITS_RECEIVED (Optional – for optional error check described below)

  ♦ Count of the total number of FC units consumed by valid TLPs Received since Flow Control initialization, modulo $2^{[Field\ Size]}$ (where [Field Size] is 8 for PH, NPH, and CPLH and 12 for PD, NPD and CPLD)

  ♦ Set to all 0's at interface initialization

  ♦ Updated as shown:

  CREDITS_RECEIVED :=

  (CREDITS_ RECEIVED + Increment) mod $2^{[Field\ Size]}$

  (Where Increment corresponds to the credits made available, and [Field Size] is 8 for PH, NPH and CPLH and 12 for PD, NPD, and CPLD)

  for each Received TLP, provided that TLP:

  ■ passes the Data Link Layer integrity checks

  ■ is not malformed

  ■ does not consume more credits than have been allocated (see following rule)

- ❑ If a Receiver implements the CREDITS_RECEIVED counter, then when a nullified TLP (with inverted LCRC and using EDB as the end Symbol) is received, the Receiver does not modify CREDITS_RECEIVED for that TLP (see Section 3.5.2.1)

- ❑ A Receiver may optionally check for Receiver Overflow errors (TLPs exceeding CREDITS_ALLOCATED), by checking the following equation, using unsigned arithmetic:

  (CREDITS_ALLOCATED - CREDITS_RECEIVED) mod $2^{[Field\ Size]}$

  $>= 2^{[Field\ Size]} / 2$

  If the check is implemented and this equation evaluates as true, the Receiver must:

  - discard the TLP(s) without modifying the CREDITS_RECEIVED

  - de-allocate any resources which it had allocated for the TLP(s)

  If checked, this is a reported error associated with the Receiving Port (see Section 6.2)

  Note: Following a Receiver Overflow error, Receiver behavior is undefined, but it is encouraged that the Receiver continues to operate, processing Flow Control updates and accepting any TLPs which do not exceed allocated credits.

- ❑ For non-infinite NPH, NPD, PH, and CPLH types, an UpdateFC FCP must be scheduled for Transmission each time the following sequence of events occurs:

  - all advertised FC units for a particular type of credit are consumed by TLPs received

  - one or more units of that type are made available by TLPs processed

- ❑ For non-inifinite PD and CPLD types, when the number of available credits is less than Max_Payload_Size, an UpdateFC FCP must be scheduled for Transmission each time one or more units of that type are made available by TLPs processed

❑ UpdateFC FCPs may be scheduled for Transmission more frequently than is required

❑ When the Link is in the L0 or L0s Link state, Update FCPs for each enabled type of non-infinite FC credit must be scheduled for transmission at least once every 30 μs (-0%/+50%), except when the Extended Sync bit of the Control Link register is set, in which case the limit is 120 μs (-0%/+50%).

- A timeout mechanism may optionally be implemented. If implemented, such a mechanism must:

  ♦ be active only when the Link is in the L0 or L0s Link state

  ♦ use a timer with a limit of 200 μs (-0%/+50%), where the timer is reset by the receipt of any Init or Update FCP. Alternately, the timer may be reset by the receipt of any DLLP (See Section 3.4)

  ♦ upon timer expiration, instruct the Physical Layer to retrain the Link (via the LTSSM Recovery state, Section 4.2.6.4)

Note: The implementation of this optional mechanism is strongly encouraged. It is anticipated that future revisions of this specification may change this mechanism from optional to required.

🖊 IMPLEMENTATION NOTE

Use of "Infinite" FC Advertisement

For a given implementation it is possible that not all of the queue types need to be physically implemented in hardware for all Virtual Channels. For example, since Non-Posted Writes are only allowed on Virtual Channel 0, there is no need to implement a Non-Posted Data queue for Virtual Channels other than VC0. For unimplemented queues, the Receiver can eliminate the need to present the appearance of tracking Flow Control credits by advertising infinite Flow Control credits during initialization.

📝 IMPLEMENTATION NOTE

Flow Control Update Frequency

For components subject to receiving streams of TLPs, it is desirable to implement receive buffers larger than the minimum size required to prevent Transmitter throttling due to lack of available credits. Likewise, UpdateFC FCPs must be returned such that the time required to send, receive and process the UpdateFC is sufficient. Table 2-28 shows recommended values for the frequency of transmission during normal operation based on Link Width and Max_Payload_Size values. Note that the values given in Table 2-28 do not account for any delays caused by the Receiver or Transmitter being in L0s. For improved performance and/or power-saving, it may be desirable to use a Flow Control update policy that is more sophisticated than a simple timer. Any such policy is implementation specific, and beyond the scope of this document.

The values are calculated as a function of the largest TLP payload size and Link width. The values are measured at the Port of the TLP Receiver, starting with the time the last Symbol of a TLP in received to the first Symbol of the UpdateFC DLLP being transmitted. The values are calculated using the formula:

$$\frac{(Max\_Payload\_Size + TLPOverhead) * UpdateFactor}{LinkWidth} + InternalDelay$$

where:

Max_Payload_Size	The value in the Max_Payload_Size field of the Device Control register
TLP Overhead	Represents the additional TLP components which consume Link bandwidth (header, LCRC, framing Symbols) and is treated here as a constant value of 28 Symbols
UpdateFactor	Represents the number of maximum size TLPs sent during the time between UpdateFC receptions, and is used to balance Link bandwidth efficiency and receive buffer sizes – the value varies according to Max_Payload_Size and Link width, and is included in Table 2-28
LinkWidth	The operating width of the Link
InternalDelay	Represents the internal processing delays for received TLPs and transmitted DLLPs, and is treated here as a constant value of 19 Symbol Times

Table 2-28: UpdateFC Transmission Latency Guidelines by Link Width and Max Payload (Symbol Times)

		Link Operating Width						
		x1	**x2**	**x4**	**x8**	**x12**	**x16**	**x32**
Max_Payload_Size (bytes)	**128**	237 UF = 1.4	128 UF = 1.4	73 UF = 1.4	67 UF = 2.5	58 UF = 3.0	48 UF = 3.0	33 UF = 3.0
	256	416 UF = 1.4	217 UF = 1.4	118 UF = 1.4	107 UF = 2.5	90 UF = 3.0	72 UF = 3.0	45 UF = 3.0
	512	559 UF = 1.0	289 UF = 1.0	154 UF = 1.0	86 UF = 1.0	109 UF = 2.0	86 UF = 2.0	52 UF = 2.0
	1024	1071 UF = 1.0	545 UF = 1.0	282 UF = 1.0	150 UF = 1.0	194 UF = 2.0	150 UF = 2.0	84 UF = 2.0
	2048	2095 UF = 1.0	1057 UF = 1.0	538 UF = 1.0	278 UF = 1.0	365 UF = 2.0	278 UF = 2.0	148 UF = 2.0
	4096	4143 UF = 1.0	2081 UF = 1.0	1050 UF = 1.0	534 UF = 1.0	706 UF = 2.0	534 UF = 2.0	276 UF = 2.0

2.7.   Data Integrity

The basic data reliability mechanism in PCI Express is contained within the Data Link Layer, which uses a 32-bit CRC (LCRC) code to detect errors in TLPs on a Link-by-Link basis, and applies a Link-by-Link retransmit mechanism for error recovery. A TLP is a unit of data and transaction control that is created by a data-source at the "edge" of the PCI Express domain (such as an Endpoint or Root Complex), potentially routed through intermediate components (i.e., Switches) and consumed by the ultimate PCI Express recipient. As a TLP passes through a Switch, the Switch may need to change some control fields without modifying other fields that should not change as the packet traverses the path. Therefore, the LCRC is regenerated by Switches. Data corruption may occur internally to the Switch, and the regeneration of a good LCRC for corrupted data masks the existence of errors. To ensure end-to-end data integrity detection in systems that require high data reliability, a Transaction Layer end-to-end 32-bit CRC (ECRC) can be placed in the TLP Digest field at the end of a TLP. The ECRC covers all fields that do not change as the TLP traverses the path (invariant fields). The ECRC is generated by the Transaction Layer in the source component, and checked in the destination component. A Switch that supports ECRC checking checks ECRC on TLPs that are destined to a destination within the Switch itself. On all other TLPs a Switch must preserve the ECRC (forward it untouched) as an integral part of the TLP.

In some cases, the data in a TLP payload is known to be corrupt at the time the TLP is generated, or may become corrupted while passing through an intermediate component, such as a Switch. In these cases, error forwarding, also known as data poisoning, can be used to indicate the corruption to the device consuming the data.

2.7.1.   ECRC Rules

The capability to generate and check ECRC is reported to software, and the ability to do so is enabled by software (see Section 7.10.7).

❑   If a device is enabled to generate ECRC, it must calculate and apply ECRC for all TLPs originated by the device

❑   Switches must pass TLPs with ECRC unchanged from the Ingress Port to the Egress Port

❑ If a device reports the capability to check ECRC, it must support Advanced Error Reporting (see Section 6.2)

❑ If a device is enabled to check ECRC, it must do so for all TLPs received by the device including ECRC

5 • Note that it is still possible for the device to receive TLPs without ECRC, and these are processed normally – this is not an error

Note that a Switch may perform ECRC checking on TLPs passing through the Switch. ECRC Errors detected by the Switch are reported in the same way any other device would report them, but do not alter the TLPs passage through the Switch.

10 A 32-bit ECRC is calculated for the entire TLP (header and data payload) using the following algorithm and appended to the end of the TLP (see Figure 2-3):

❑ The ECRC value is calculated using the following algorithm (see Figure 2-28).

❑ The polynomial used has coefficients expressed as 04C1 1DB7h

❑ The seed value (initial value for ECRC storage registers) is FFFF FFFFh

15 ❑ All invariant fields of the TLP header and the entire data payload (if present) are included in the ECRC calculation, all bits in variant fields must be set to 1 for ECRC calculations.

• Bit 0 of the Type field is variant[16]

• The EP field is variant

• all other fields are invariant

20 ❑ ECRC calculation starts with bit 0 of byte 0 and proceeds from bit 0 to bit 7 of each byte of the TLP

❑ The result of the ECRC calculation is complemented, and the complemented result bits are mapped into the 32-bit TLP Digest field as shown in Table 2-29.

Table 2-29: Mapping of Bits into ECRC Field

ECRC Result Bit	Corresponding Bit Position in the 32-bit TLP Digest Field
0	7
1	6
2	5
3	4
4	3
5	2
6	1
7	0
8	15

[16] Bit 0 of the Type field changes when a Configuration Request is changed from Type 1 to Type 0.

ECRC Result Bit	Corresponding Bit Position in the 32-bit TLP Digest Field
9	14
10	13
11	12
12	11
13	10
14	9
15	8
16	23
17	22
18	21
19	20
20	19
21	18
22	17
23	16
24	31
25	30
26	29
27	28
28	27
29	26
30	25
31	24

❑ The 32-bit ECRC value is placed in the TLP Digest field at the end of the TLP (see Figure 2-3)

❑ For TLPs including a TLP Digest field used for an ECRC value, Receivers which support end-to-end data integrity checking, check the ECRC value in the TLP Digest field by:

• applying the same algorithm used for ECRC calculation (above) to the received TLP, not including the 32-bit TLP Digest field of the received TLP

• comparing the calculated result with the value in the TLP Digest field of the received TLP

How the Receiver makes use of the end-to-end data integrity check provided through the ECRC is beyond the scope of this document.

OM13777A
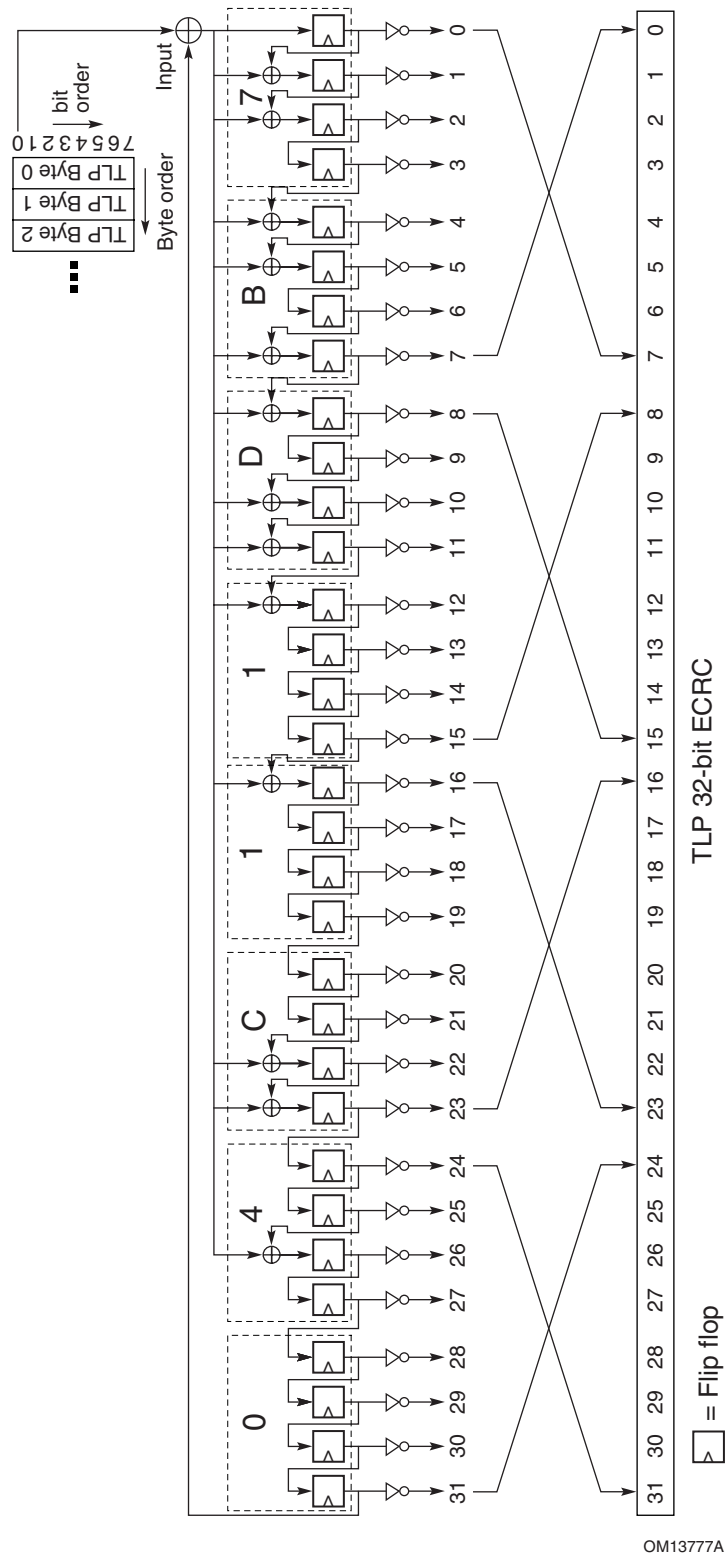
Figure 2-28: Calculation of 32-bit ECRC for TLP End to End Data Integrity Protection

IMPLEMENTATION NOTE

Protection of TD Bit Inside Switches

It is of utmost importance that Switches insure and maintain the integrity of the TD bit in TLPs that they receive and forward (i.e., by applying a special internal protection mechanism), since corruption of the TD bit will cause the ultimate target device to misinterpret the presence or absence of the TLP digest field.

Similarly, it is highly recommended that Switches provide internal protection to other variant fields in TLPs that they receive and forward, as the end-to-end integrity of variant fields is not sustained by the ECRC.

IMPLEMENTATION NOTE

Data Link Layer Does Not Have Internal TLP Visibility

Since the Data Link Layer does not process the TLP header (it determines the start and end of the TLP based on indications from the Physical Layer), it is not aware of the existence of the TLP Digest field, and simply passes it to the Transaction Layer as a part of the TLP.

2.7.2. Error Forwarding

Error Forwarding (also known as data poisoning), is indicated by setting the EP field to 1b. The rules for doing this are specified in Section 2.7.2.2. Here are some examples of cases where Error Forwarding might be used:

❑ Example #1: A read from main memory encounters uncorrectable error

❑ Example #2: Parity error on a PCI write to main memory

❑ Example #3: Data integrity error on an internal data buffer or cache.

2.7.2.1. Error Forwarding Usage Model

❑ Error Forwarding is only used for Read Completion Data or Write Data, never for the cases when the error is in the "header" (request phase, address/command, etc.). Requests/Completions with header errors cannot be forwarded in general since true destination cannot be positively known and, therefore, forwarding may cause direct or side effects such as data corruption, system failures, etc.

❑ Used for controlled propagation of errors through the system, system diagnostics, etc.

❑ Does not cause Link Layer Retry – Poisoned TLPs will be retried only if there are transmission errors on PCI Express as determined by the TLP error detection mechanisms in the Data Link Layer. The Poisoned TLP may ultimately cause the originator of the request to re-issue it (at the Transaction Layer of above) in the case of read operation or to take some other action. Such use of Error Forwarding information is beyond the scope of this specification.

2.7.2.2.   Rules For Use of Data Poisoning

❑ Support for TLP poisoning in a Transmitter is optional.

❑ Data poisoning applies only to the data within a Write Request (Posted or Non-Posted) or a Read Completion.

• Poisoning of a TLP is indicated by a 1b value in the EP field

❑ If a Transmitter supports data poisoning, TLPs that are known to the Transmitter to include bad data must use the poisoning mechanism defined above.

❑ Receipt of a Poisoned TLP is a reported error associated with the Receiving device/function (see Section 6.2)

❑ A Poisoned Configuration Write Request must be discarded by the Completer, and a Completion with a Completion Status of UR is returned (see Section 2.2.9).

• A Switch must route a Poisoned Configuration Write Request in the same way it would route a non-Poisoned Configuration Write Request, unless the Request addresses the configuration space of the Switch itself, in which case the Switch is the Completer for the Request and must follow the above rule.

❑ A Poisoned I/O or Memory Write Request, or a Message with data (except for vendor-defined Messages), that addresses a control register or control structure in the Completer must be handled as an Unsupported Request (UR) by the Completer (see Section 2.2.9).

• A Switch must route a Poisoned I/O or Memory Write Request or Message with data in the same way it would route the same Request if it were not poisoned, unless the Request addresses a control register or control structure of the Switch itself, in which case the Switch is the Completer for the Request and must follow the above rule.

For some applications it may be desirable for the Completer to use poisoned data in Write Requests which do not target control registers or control structures – such use is not forbidden. Similarly, it may be desirable for the Requester to use data marked poisoned in Completions – such use is also not forbidden. The appropriate use of poisoned information is application specific, and is not discussed in this document.

This document does not define any mechanism for determining which part or parts of the data payload of a Poisoned TLP are actually corrupt and which, if any, are not corrupt.

2.8. Completion Timeout Mechanism

In any split transaction protocol, there is a risk associated with the failure of a Requester to receive an expected Completion. To allow Requesters to attempt recovery from this situation in a standard manner, the Completion Timeout mechanism is defined. This mechanism is intended to be activated only when there is no reasonable expectation that the Completion will be returned, and should never occur under normal operating conditions. Note that the values specified here do not reflect expected service latencies, and must not be used to estimate typical response times.

PCI Express devices that issue Requests requiring Completions must implement the Completion Timeout mechanism. An exception is made for Configuration Requests (see below). The Competion Timout mechanism is activated for each Request that requires one or more Completions when the Request is transmitted. Since PCI Express Switches do not autonomously initiate Requests that need Completions, the requirement for Completion Timeout support is limited only to Root Complexes, PCI Express-PCI Bridges, and Endpoint devices.

This specification defines the following range for the minimum/maximum acceptable timer values for the Completion Timeout mechanism:

❑ The Completion Timeout timer must not expire (i.e., cause a timeout event) in less than 50 µs. It is strongly recommend that unless an application requires this level of timer granularity the minimum time should not expire in less than 10 ms.

❑ The Completion Timeout timer must expire if a Request is not completed in 50 ms.

A Completion Timeout is a reported error associated with the Requestor device/function (see Section 6.2).

Note: A Memory Read Request for which there are multiple Completions must be considered completed only when all Completions have been received by the Requester. If some, but not all, requested data is returned before the Completion Timeout timer expires, the Requestor is permitted to keep or to discard the data that was returned prior to timer expiration.

Configuration Requests have special requirements (see Sections 2.3.1 and 6.6). Because of these special requirements, the support and timer values for a Completion Timeout for Configuration Requests are implementation specific. Furthermore, PCI Express to PCI/PCI-X Bridges, by default, are not enabled to return Configuration Request Retry Status (CRS) for Configuration Requests to a PCI/PCI-X device behind the Bridge. This may result in lengthy completion delays that must be comprehended by the Completion Timeout value in the Root Complex. System software may enable PCI Express to PCI/PCI-X Bridges to return Configuration Request Retry Status by setting the Bridge Configuration Retry Enable bit in the Device Control register, subject to the restrictions noted in the *PCI Express to PCI/PCI-X Bridge Specification, Rev. 1.0.*

2.9.   Link Status Dependencies

2.9.1.   Transaction Layer Behavior in DL_Down Status

DL_Down status indicates that there is no connection with another component on the Link, or that the connection with the other component has been lost and is not recoverable by the Physical or Data Link Layers. This section specifies the Transaction Layer's behavior when the Data Link Layer reports DL_Down status to the Transaction Layer, indicating that the Link is non-operational.

For a Root Complex, or any Port on a Switch other than the one closest to the Root Complex, DL_Down status is handled by:

❑ initializing back to their default state any buffers or internal states associated with outstanding requests transmitted downstream

- Note: Port configuration registers must not be affected, except as required to update status associated with the transition to DL_Down

❑ forming completions for any Requests submitted by the device core for Transmission, returning Unsupported Request Completion Status, then discarding the Requests

- This is a reported error associated with the device/function for the (virtual) Bridge associated with the Port (see Section 6.2)

- Requests already being processed by the Transaction Layer, for which it may not be practical to return Completions, are discarded

    Note: This is equivalent to the case where the Request had been Transmitted but not yet Completed before the Link status became DL_Down

    ♦ These cases are handled by the Requester using the Completion Timeout mechanism

    Note: The point at which a Request becomes "uncompletable" is implementation specific

❑ discarding all Completions submitted by the device core for Transmission

For a Port on an Endpoint, and the Port on a Switch or Bridge that is closest to the Root Complex, DL_Down status is handled as a reset by:

❑ returning all PCI Express-specific registers, state machines and externally observable state to the specified default or initial conditions (except for registers defined as sticky – see Section 7.4)

❑ discarding all TLPs being processed

❑ (for Switch and Bridge) propagating hot reset to all other Ports

2.9.2.   Transaction Layer Behavior in DL_Up Status

DL_Up status indicates that a connection has been established with another component on the associated Link. This section specifies the Transaction Layer's behavior when the Data Link Layer reports entry to the DL_Up status to the Transaction Layer, indicating that the Link is operational. These behaviors relate to Slot Power Limit support.

For a Downstream Port on a Root Complex or a Switch:

❑ When transitioning from a non-DL_Up Status to a DL_Up Status, the Port must initiate the transmission of a Set_Slot_Power_Limit Message to the other component on the Link to convey the value programmed in the Slot Power Limit Scale and Value fields of the Slot Capabilities register.

3

3. Data Link Layer Specification

The Data Link Layer acts as an intermediate stage between the Transaction Layer and the Physical Layer. Its primary responsibility is to provide a reliable mechanism for exchanging Transaction Layer Packets (TLPs) between the two components on a Link.

3.1. Data Link Layer Overview
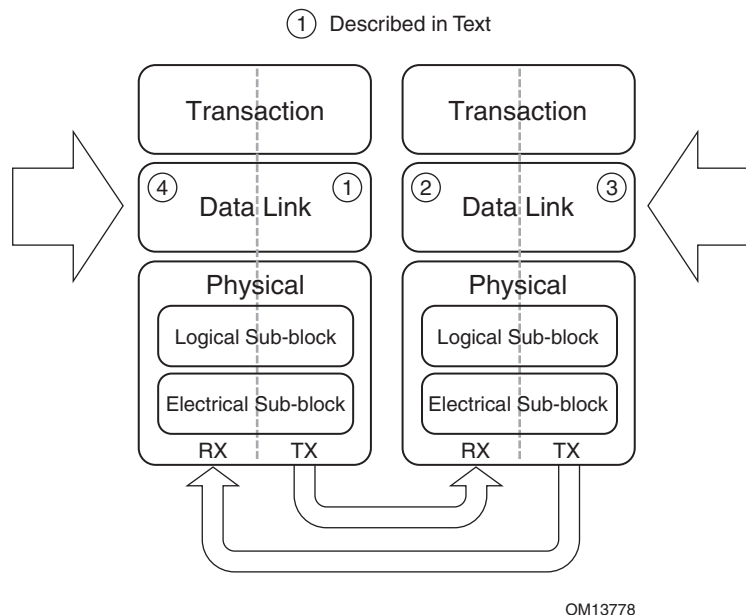
Figure 3-1: Layering Diagram Highlighting the Data Link Layer

The Data Link Layer is responsible for reliably conveying Transaction Layer Packets (TLPs) supplied by the Transaction Layer across a PCI Express Link to the other component's Transaction Layer. Services provided by the Data Link Layer include:

Data Exchange:

❑ Accept TLPs for transmission from the Transmit Transaction Layer and convey them to the Transmit Physical Layer

❑ Accept TLPs received over the Link from the Physical Layer and convey them to the Receive Transaction Layer

119

Error Detection and Retry:

❑ TLP Sequence Number and LCRC generation

❑ Transmitted TLP storage for Data Link Layer Retry

❑ Data integrity checking for TLPs and Data Link Layer Packets (DLLPs)

5 ❑ Positive and negative acknowledgement DLLPs

❑ Error indications for error reporting and logging mechanisms

❑ Link Acknowledgement Timeout replay mechanism

Initialization and power management services:

❑ Track Link state and convey active/reset/disconnected state to Transaction Layer

10 Data Link Layer Packets (DLLPs) are:

❑ used for Link Management functions including TLP acknowledgement, power management, and exchange of Flow Control information.

❑ transferred between Data Link Layers of the two directly connected components on a Link

DLLPs are sent point-to-point, between the two components on one Link. TLPs are routed from
15 one component to another, potentially through one or more intermediate components.

Data Integrity checking for DLLPs and TLPs is done using a CRC included with each packet sent across the Link. DLLPs use a 16 bit CRC and TLPs (which can be much longer) use a 32-bit LCRC. TLPs additionally include a sequence number, which is used to detect cases where one or more entire TLPs have been lost.

20 Received DLLPs which fail the CRC check are discarded. The mechanisms which use DLLPs may suffer a performance penalty from this loss of information, but are self-repairing such that a successive DLLP will supercede any information lost.

TLPs which fail the data integrity checks (LCRC and sequence number), or which are lost in transmission from one component to another, are re-sent by the Transmitter. The Transmitter
25 stores a copy of all TLPs sent, re-sending these copies when required, and purges the copies only when it receives a positive acknowledgement of error-free receipt from the other component. If a positive acknowledgement has not been received within a specified time period, the Transmitter will automatically start re-transmission. The Receiver can request an immediate re-transmission using a negative acknowledgement.

30 The Data Link Layer appears as an information conduit with varying latency to the Transaction Layer. On any given individual Link all TLPs fed into the Transmit Data Link Layer (1 and 3) will appear at the output of the Receive Data Link Layer (2 and 4) in the same order at a later time, as illustrated in Figure 3-1. The latency will depend on a number of factors, including pipeline latencies, width and operational frequency of the Link, transmission of electrical signals across the
35 Link, and delays caused by Data Link Layer Retry. Because of these delays, the Transmit Data Link Layer (1 and 3) can apply backpressure to the Transmit Transaction Layer, and the Receive Data Link Layer (2 and 4) communicates the presence or absence of valid information to the Receive Transaction Layer.

3.2. Data Link Control and Management State Machine

The Data Link Layer tracks the state of the Link. It communicates Link status with the Transaction and Physical Layers, and performs Link Management through the Physical Layer. The Data Link Layer contains the Data Link Control and Management State Machine (DLCMSM) to perform these tasks. The states for this machine are described below, and are shown in Figure 3-2.

States:

❑ DL_Inactive – Physical Layer reporting Link is non-operational or nothing is connected to the Port

❑ DL_Init – Physical Layer reporting Link is operational, initialize Flow Control for the default Virtual Channel

❑ DL_Active – Normal operation mode

Status Outputs:

❑ DL_Down – The Data Link Layer is not communicating with the component on the other side of the Link.

❑ DL_Up – The Data Link Layer is communicating with the component on the other side of the Link.
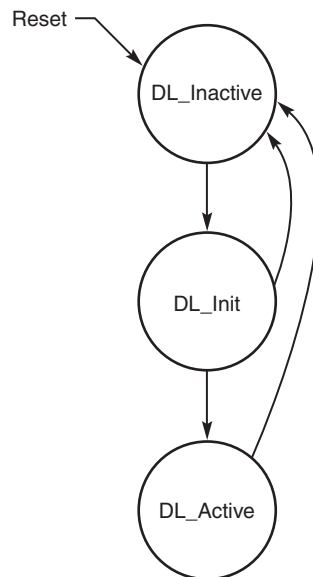
Figure 3-2: Data Link Control and Management State Machine

3.2.1. Data Link Control and Management State Machine Rules

Rules per state:

❑ DL_Inactive

5
- Initial state following PCI Express hot, warm, or cold reset (see Section 6.6)

- Upon entry to DL_Inactive

  ♦ Reset all Data Link Layer state information to default values

  ♦ Discard the contents of the Data Link Layer Retry Buffer (refer Section 3.5)

- While in DL_Inactive:

10
  ♦ Report DL_Down status to the Transaction Layer as well as to the rest of the Data Link Layer

  ♦ Note: This will cause the Transaction Layer to discard any outstanding transactions and to terminate internally any attempts to transmit a TLP. For a Port on a Root Complex or at the "bottom" of a Switch, this is equivalent to a "Hot-Remove." For Port on an
15
  Endpoint or at the "top" of a Switch, having the Link go down is equivalent to a hot reset (see Section 2.9).

  ♦ Discard TLP information from the Transaction and Physical Layers

  ♦ Do not generate or accept DLLPs

- Exit to DL_Init if:

20
  ♦ Indication from the Transaction Layer that the Link is not disabled by software and the Physical Layer reports Physical LinkUp = 1

❑ DL_Init

- While in DL_Init:

  ♦ Initialize Flow Control for the default Virtual Channel, VC0, following the Flow Control
25
  initialization protocol described in Section 3.3

  ♦ Report DL_Down status while in state FC_INIT1; DL_Up status in state FC_INIT2

- Exit to DL_Active if:

  ♦ Flow Control initialization completes successfully, and the Physical Layer continues to report Physical LinkUp = 1

30
- Terminate attempt to initialize Flow Control for VC0 and Exit to DL_Inactive if:

  ♦ Physical Layer reports Physical LinkUp = 0

❑ DL_Active

  • DL_Active is referred to as the normal operating state

  • While in DL_Active:

    ♦ Accept and transfer TLP information with the Transaction and Physical Layers as specified in this chapter

    ♦ Generate and accept DLLPs as specified in this chapter

    ♦ Report DL_Up status to the Transaction and Data Link Layers

  • Exit to DL_Inactive if:

    ♦ Physical Layer reports Physical LinkUp = 0

IMPLEMENTATION NOTE

Physical Layer Throttling

Note that there are conditions where the Physical Layer may be temporarily unable to accept TLPs and DLLPs from the Data Link Layer. The Data Link Layer must comprehend this by providing mechanisms for the Physical Layer to communicate this condition, and for TLPs and DLLPs to be temporarily blocked by the condition.

3.3. Flow Control Initialization Protocol

Before starting normal operation following power-up or interconnect Reset, it is necessary to initialize Flow Control for the default Virtual Channel, VC0 (see Section 6.6). In addition, when additional Virtual Channels (VCs) are enabled, the Flow Control initialization process must be completed for each newly enabled VC before it can be used (see Section 2.4.2). This section describes the initialization process that is used for all VCs. Note that since VC0 is enabled before all other VCs, no TLP traffic of any kind will be active prior to initialization of VC0. However, when additional VCs are being initialized there will typically be TLP traffic flowing on other, already enabled, VCs. Such traffic has no direct effect on the initialization process for the additional VC(s).

There are two states in the VC initialization process. These states are:

❑ FC_INIT1

❑ FC_INIT2

The rules for this process are given in the following section. Figure 3-3 shows a flowchart of the process.
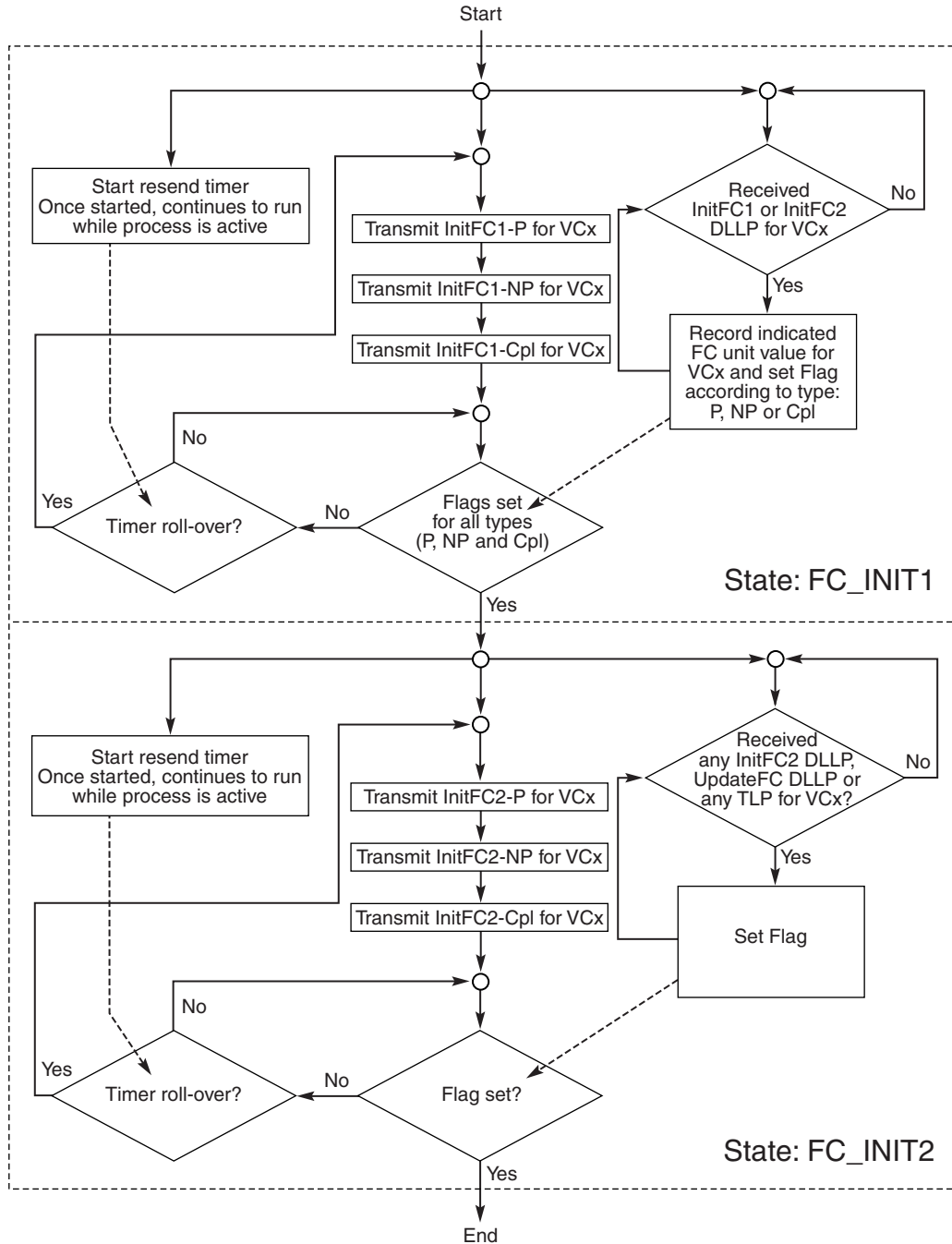
Figure 3-3: Flowchart Diagram of Flow Control Initialization Protocol

3.3.1. Flow Control Initialization State Machine Rules

❏ If at any time during initialization for VCx (where x does not equal 0) VCx is disabled, the flow control initialization process for VCx is terminated

❏ Rules for state FC_INIT1:

- Entered when initialization of a VC (VCx) is required

  ♦ Entrance to DL_Init state

  ♦ When a VC is enabled by software (see Section 7.11)

- While in FC_INIT1:

  ♦ Transaction Layer must block transmission of TLPs using VCx

  ♦ Transmit the following uninterrupted sequence of three successive InitFC1 DLLPs for VCx in the following pattern:

    ■ InitFC1 – P (first)

    ■ InitFC1 – NP (second)

    ■ InitFC1 – Cpl (third)

  ♦ Repeat this InitFC1 DLLP transmission sequence as follows:

    ■ For VC0, transmit continuously at the maximum rate possible on the Link (resend timer value is 0)

    ■ For VCs other than VC0, repeat the sequence when no other TLPs or DLLPs are available for Transmission, but no less frequently than at an interval of 17 µs (-0%/+100%), measured from the start of transmission of the preceding sequence

  ♦ Process received InitFC1 and InitFC2 DLLPs:

    ■ Record the indicated FC unit values

    ■ Set Flag FI1 once FC unit values have been recorded for each of P, NP, and Cpl for VCx

- Exit to FC_INIT2 if:

  ♦ Flag FI1 has been set indicating that FC unit values have been recorded for each of P, NP, and Cpl for VCx

❑ Rules for state FC_INIT2:

- While in FC_INIT2:

  ♦ Transmission of TLPs using VCx by the Transaction Layer is permitted

  ♦ Transmit the following uninterrupted sequence of three successive InitFC2 DLLPs for VCx in the following pattern:

    ■ InitFC2 – P (first)

    ■ InitFC2 – NP (second)

    ■ InitFC2 – Cpl (third)

  ♦ Repeat this InitFC2 DLLP transmission sequence as follows:

    ■ For VC0, transmit continuously at the maximum rate possible on the Link (resend timer value is 0)

    ■ For VCs other than VC0, repeat the sequence when no other TLPs or DLLPs are available for Transmission, but no less frequently than at an interval of 17 µs (-0%/+100%), measured from the start of transmission of the preceding sequence

  ♦ Process received InitFC1 and InitFC2 DLLPs:

    ■ Ignore the indicated FC unit values

    ■ Set flag FI2 on receipt of any InitFC2 DLLP for VCx

  ♦ Set flag FI2 on receipt of any TLP on VCx, or any UpdateFC DLLP for VCx

- Signal completion and exit if:

  ♦ Flag FI2 has been set

❑ Violations of Flow Control initialization protocol are Data Link Layer Protocol Errors (DLLPE). Checking for such errors in FC initialization protocol is optional. If checking is implemented, any detected error is a reported error associated with the Port (see Section 6.2)

🗒 IMPLEMENTATION NOTE

Example of Flow Control Initialization

Figure 3-4 illustrates an example of the Flow Control initialization protocol for VC0 between a Switch and an Endpoint. In this example, each component advertises the minimum permitted values for each type of Flow Control credit. For both components the largest Max_Payload_Size value supported is 1024 bytes, corresponding to a data payload credit advertisement of 040h. All DLLPs are shown as received without error.

Endpoint   Endpoint   Endpoint   Endpoint

In this example, the Endpoint enters the initialization sequence for VC0 first, followed shortly by the Switch.

Head of 2nd set of InitFC1's from Endpoint

Head of 1st set of InitFC1's from Endpoint

After the Switch receives a complete set of InitFC1 DLLPs, it transitions to FC_INIT2...

...and starts transmitting InitFC2 DLLPs, shortly followed by the Endpoint, which has now received a full set of InitFC1 DLLPs.

Head of 1st set of InitFC2's from Endpoint

Head of 1st set of InitFC2's from Switch

Both components have now received at least one InitFC2 DLLP, and the initialization sequence is complete.

Legend – DLLP Type field:
P, NP or Cpl
1 = InitFC1, 2 = InitFC2

All DLLP field values shown are in hexadecimal.
Credit values are the minimum permitted values.
Largest Max_Payload_Size supported is 1024 bytes.

Switch   Switch   Switch   Switch

OM14548

Figure 3-4: VC0 Flow Control Initialization Example

3.4. Data Link Layer Packets (DLLPs)

The following DLLPs are used to support Link operations:

❑ Ack DLLP: TLP Sequence number acknowledgement; used to indicate successful receipt of some number of TLPs

5 ❑ Nak DLLP: TLP Sequence number negative acknowledgement; used to initiate a Data Link Layer Retry

❑ InitFC1, InitFC2, and UpdateFC DLLPs: For Flow Control

❑ DLLPs used for Power Management

3.4.1. Data Link Layer Packet Rules

10 All DLLP fields marked Reserved (sometimes abbreviated as R) must be filled with all 0's when a DLLP is formed. Values in such fields must be ignored by Receivers. The handling of reserved values in encoded fields is specified for each case.

All DLLPs include the following fields:

❑ DLLP Type - Specifies the type of DLLP. The defined encodings are shown in Table 3-1.

15 ❑ 16-bit CRC

See Figure 3-5.
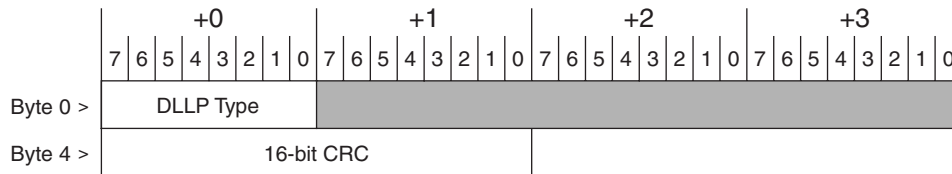
OM14303A

Figure 3-5: DLLP Type and CRC Fields

Table 3-1: DLLP Type Encodings

Encodings	DLLP Type
0000 0000	Ack
0001 0000	Nak
0010 0000	PM_Enter_L1
0010 0001	PM_Enter_L23
0010 0011	PM_Active_State_Request_L1
0010 0100	PM_Request_Ack
0011 0000	Vendor Specific – Not used in normal operation
0100 0$v_2v_1v_0$	InitFC1-P (v[2:0] specifies Virtual Channel)
0101 0$v_2v_1v_0$	InitFC1-NP

Encodings	DLLP Type
0110 $0v_2v_1v_0$	InitFC1-Cpl
1100 $0v_2v_1v_0$	InitFC2-P
1101 $0v_2v_1v_0$	InitFC2-NP
1110 $0v_2v_1v_0$	InitFC2-Cpl
1000 $0v_2v_1v_0$	UpdateFC-P
1001 $0v_2v_1v_0$	UpdateFC-NP
1010 $0v_2v_1v_0$	UpdateFC-Cpl
All other encodings	Reserved

❑ For Ack and Nak DLLPs (see Figure 3-6):

- The AckNak_Seq_Num field is used to indicate what TLPs are affected

- Transmission and Reception is handled by the Data Link Layer according to the rules provided in Section 3.5.

❑ For InitFC1, InitFC2, and UpdateFC DLLPs:

- The HdrFC field contains the credit value for headers of the indicated type (P, NP, or Cpl)

- The DataFC field contains the credit value for payload Data of the indicated type (P, NP, or Cpl)

- The packet formats are shown in Figure 3-7, Figure 3-8, and Figure 3-9

- Transmission is triggered by the Data Link Layer when initializing Flow Control for a Virtual Channel (see Section 3.3), and following Flow Control initialization by the Transaction Layer according to the rules in Section 2.6

- Checked for integrity on reception by the Data Link Layer and if correct, the information content of the DLLP is passed to the Transaction Layer. If the check fails, the information is discarded.

  Note: InitFC1 and InitFC2 DLLPs are used only for VC initialization

❑ Power Management (PM) DLLPs (see Figure 3-10):

- Transmission is triggered by the component's power management logic according to the rules in Chapter 5

- Checked for integrity on reception by the Data Link Layer, then passed to the component's power management logic

❑ Vendor Specific (see Figure 3-11)

	+0		+1		+2		+3	
	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Byte 0 >	0000 0000 - Ack 0001 0000 - Nak	Reserved		AckNak_Seq_Num
Byte 4 >	16-bit CRC			
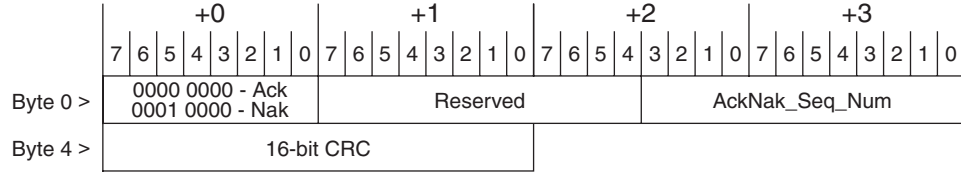
OM13781A

Figure 3-6: Data Link Layer Packet Format for Ack and Nak

	+0		+1		+2		+3	
	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Byte 0 >	0100 - P 0101 - NP 0110 - Cpl	0	v[2:0] -VC ID	R	HdrFC	R	DataFC
Byte 4 >	16-bit CRC			
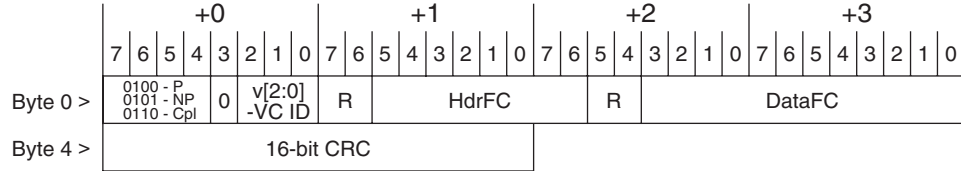
OM13782A

Figure 3-7: Data Link Layer Packet Format for InitFC1

	+0		+1		+2		+3	
	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Byte 0 >	1100 - P 1101 - NP 1110 - Cpl	0	v[2:0] -VC ID	R	HdrFC	R	DataFC
Byte 4 >	16-bit CRC			
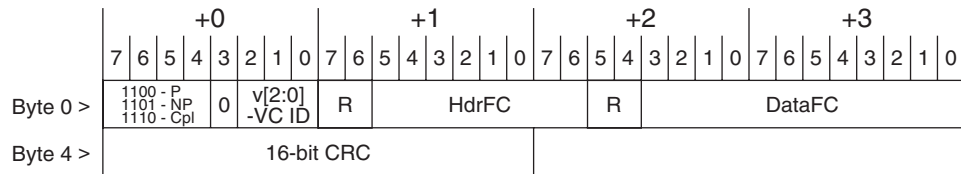
OM13783A

Figure 3-8: Data Link Layer Packet Format for InitFC2

	+0		+1		+2		+3	
	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Byte 0 >	1000 - P 1001 - NP 1010 - Cpl	0	v[2:0] -VC ID	R	HdrFC	R	DataFC
Byte 4 >	16-bit CRC			
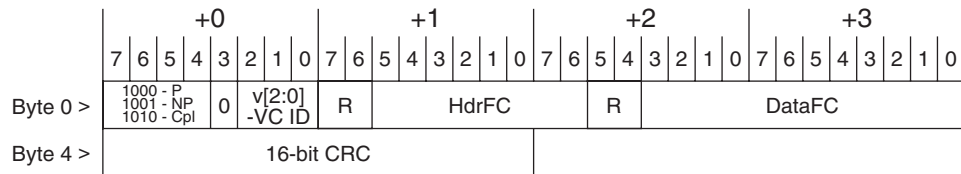
OM13784A

Figure 3-9: Data Link Layer Packet Format for UpdateFC

	+0		+1		+2		+3	
	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Byte 0 >	0 0 1 0 0 x x x	Reserved		
Byte 4 >	16-bit CRC			
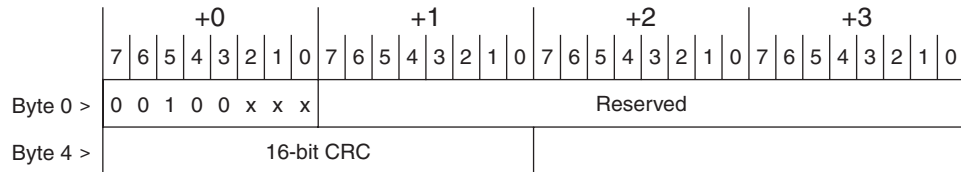
OM14304A

Figure 3-10: PM Data Link Layer Packet Format
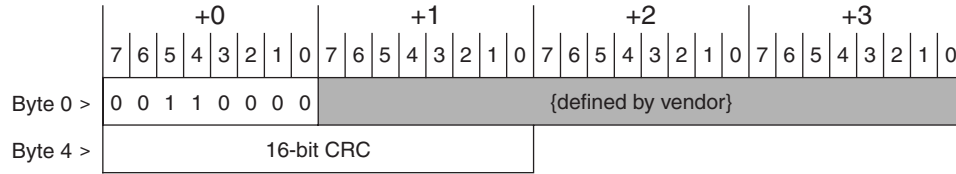
OM14305A

Figure 3-11: Vendor Specific Data Link Layer Packet Format

The following are the characteristics and rules associated with Data Link Layer Packets (DLLPs):

❑ DLLPs are differentiated from TLPs when they are presented to, or received from, the Physical Layer.

❑ DLLP data integrity is protected using a 16 bit CRC

❑ The CRC value is calculated using the following rules (see Figure 3-12):

- The polynomial used for CRC calculation has a coefficient expressed as 100Bh

- The seed value (initial value for CRC storage registers) is FFFFh

- CRC calculation starts with bit 0 of byte 0 and proceeds from bit 0 to bit 7 of each byte

- Note that CRC calculation uses all bits of the DLLP, regardless of field type, including reserved fields. The result of the calculation is complemented, then placed into the 16 bit CRC field of the DLLP as shown in Table 3-2.

Table 3-2: Mapping of Bits into CRC Field

CRC Result Bit	Corresponding Bit Position in the 16 bit CRC Field
0	7
1	6
2	5
3	4
4	3
5	2
6	1
7	0
8	15
9	14
10	13
11	12
12	11
13	10
14	9
15	8

Figure 3-12: Diagram of CRC Calculation for DLLPs

3.5.   Data Integrity

3.5.1.   Introduction

The Transaction Layer provides TLP boundary information to Data Link Layer. This allows the Data Link Layer to apply a Sequence Number and Link CRC (LCRC)error detection to the TLP. The Receive Data Link Layer validates received TLPs by checking the Sequence Number, LCRC code and any error indications from the Receive Physical Layer. In case of error in a TLP, Data Link Layer Retry is used for recovery.

The format of a TLP with the Sequence Number and LCRC code applied is shown in Figure 3-13.
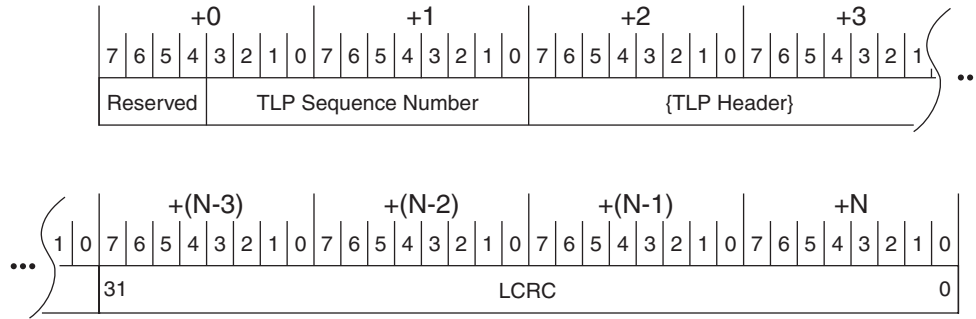
Figure 3-13: TLP with LCRC and Sequence Number Applied

3.5.2.   LCRC, Sequence Number, and Retry Management (TLP Transmitter)

The TLP transmission path through the Data Link Layer (paths labeled 1 and 3 in Figure 3-1) prepares each TLP for transmission by applying a sequence number, then calculating and appending a Link CRC (LCRC) which is used to ensure the integrity of TLPs during transmission across a Link from one component to another. TLPs are stored in a retry buffer, and are re-sent unless a positive acknowledgement of receipt is received from the other component. If repeated attempts to transmit a TLP are unsuccessful, the Transmitter will determine that the Link is not operating correctly, and instruct the Physical Layer to retrain the Link (via the LTSSM Recovery state, Section 4.2.6). If Link retraining fails, the Physical Layer will indicate that the Link is no longer up, causing the DLCMSM to move to the DL_Inactive state.

The mechanisms used to determine the TLP LCRC and the Sequence Number and to support Data Link Layer Retry are described in terms of conceptual "counters" and "flags." This description does not imply nor require a particular implementation and is used only to clarify the requirements.

3.5.2.1.   LCRC and Sequence Number Rules (TLP Transmitter)

The following counters and timer are used to explain the remaining rules in this section:

❑ The following 12-bit counters are used:

-   NEXT_TRANSMIT_SEQ – Stores the packet sequence number applied to TLPs

    ◆   Set to all 0's in DL_Inactive state

-   ACKD_SEQ – Stores the sequence number acknowledged in the most recently received Ack or Nak DLLP.

    ◆   Set to all 1's in DL_Inactive state

❑ The following 2-bit counter is used:

-   REPLAY_NUM – Counts the number of times the Retry Buffer has been re-transmitted

    ◆   Set to all 0's in DL_Inactive state

133

❑ The following timer is used:

- REPLAY_TIMER - Counts time since last Ack or Nak DLLP received

  ♦ Started at the last Symbol of any TLP transmission or retransmission, if not already running

  ♦ For each replay, reset and restart REPLAY_TIMER when sending the last Symbol of the first TLP to be retransmitted

  ♦ Restarts for each Ack DLLP received while there are unacknowledged TLPs outstanding, if, and only if, the received Ack DLLP acknowledges some TLP in the retry buffer

    ■ Note: This ensures that REPLAY_TIMER is reset only when forward progress is being made

  ♦ Reset and hold until restart conditions are met for each Nak received (except during a replay) or when the REPLAY_TIMER expires

  ♦ Not advanced during Link retraining (holds value when LTSSM in Recovery). See Section 4.2.5.4.

  ♦ Resets and holds when there are no outstanding unacknowledged TLPs

The following rules describe how a TLP is prepared for transmission before being passed to the Physical Layer:

❑ The Transaction Layer indicates the start and end of the TLP to the Data Link Layer while transferring the TLP

- The Data Link Layer treats the TLP as a "black box" and does not process or modify the contents of the TLP

❑ Each TLP is assigned a 12-bit sequence number when it is accepted from the Transmit side of Transaction Layer

- Upon acceptance of the TLP from the Transaction Layer, the packet sequence number is applied to the TLP by:

  ♦ prepending the 12-bit value in NEXT_TRANSMIT_SEQ to the TLP

  ♦ prepending four Reserved bits to the TLP, preceding the sequence number (see Figure 3-13)

- If the equation:

  (NEXT_TRANSMIT_SEQ – ACKD_SEQ) mod 4096 >= 2048

  is true, the Transmitter must cease accepting TLPs from the Transaction Layer until the equation is no longer true

- Following the application of NEXT_TRANSMIT_SEQ to a TLP accepted from the Transmit side of Transaction Layer, NEXT_TRANSMIT_SEQ is incremented:

  NEXT_TRANSMIT_SEQ := (NEXT_TRANSMIT_SEQ + 1) mod 4096
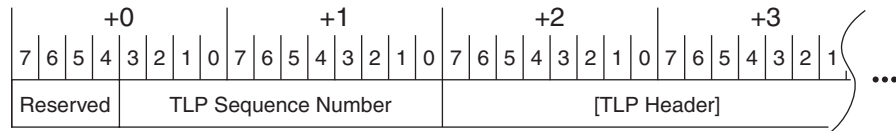
OM13787

Figure 3-14: TLP Following Application of Sequence Number and Reserved Bits

❑ TLP data integrity is protected during transfer between Data Link Layers using a 32-bit LCRC

❑ The LCRC value is calculated using the following mechanism (see Figure 3-15):

- The polynomial used has coefficients expressed as 04C1 1DB7h

- The seed value (initial value for LCRC storage registers) is FFFF FFFFh

- The LCRC is calculated using the TLP following sequence number application (see Figure 3-14)

- LCRC calculation starts with bit 0 of byte 0 (bit 8 of the TLP sequence number) and proceeds from bit 0 to bit 7 of each successive byte.

   ♦ Note that LCRC calculation uses all bits of the TLP, regardless of field type, including reserved fields

- The remainder of the LCRC calculation is complemented, and the complemented result bits are mapped into the 32-bit LCRC field as shown in Table 3-3.

Table 3-3: Mapping of Bits into LCRC Field

LCRC Result Bit	Corresponding Bit Position in the 32-bit LCRC Field
0	7
1	6
2	5
3	4
4	3
5	2
6	1
7	0
8	15
9	14
10	13
11	12
12	11
13	10
14	9
15	8

135

LCRC Result Bit	Corresponding Bit Position in the 32-bit LCRC Field
16	23
17	22
18	21
19	20
20	19
21	18
22	17
23	16
24	31
25	30
26	29
27	28
28	27
29	26
30	25
31	24

- The 32-bit LCRC field is appended to the TLP following the bytes received from the Transaction Layer (see Figure 3-13)

OM13788A
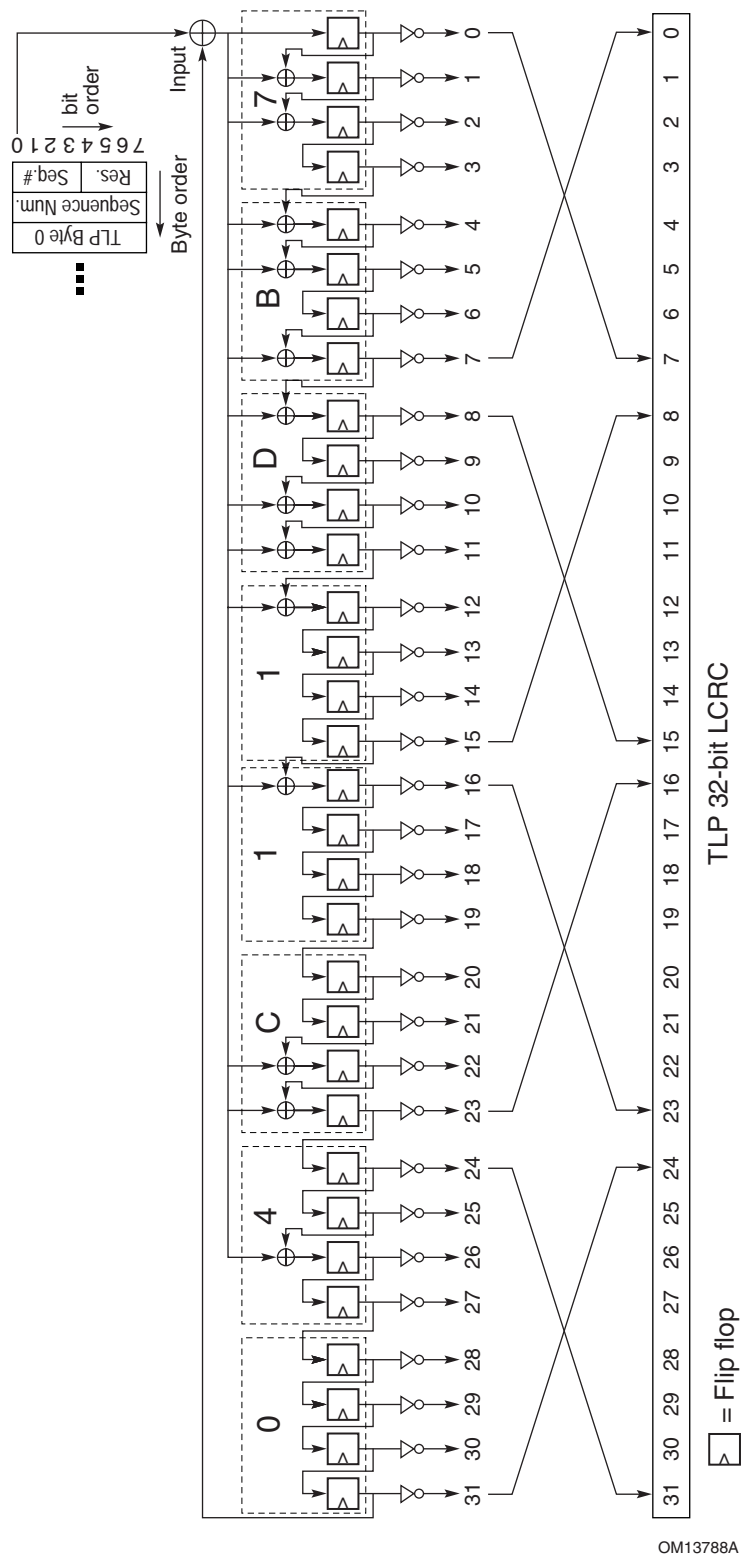
Figure 3-15: Calculation of LCRC

137

To support cut-through routing of TLPs, a Transmitter is permitted to modify a transmitted TLP to indicate that the Receiver must ignore that TLP ("nullify" the TLP).

❑ A Transmitter is permitted to nullify a TLP being transmitted; to do this in a way which will robustly prevent misinterpretation or corruption, the Transmitter must do both of the following:

- use the remainder of the calculated LCRC value without inversion (the logical inverse of the value normally used)

- indicate to the Transmit Physical Layer that the final framing Symbol must be EDB instead of END

❑ When this is done, the Transmitter does not increment NEXT_TRANSMIT_SEQ

The following rules describe the operation of the Data Link Layer Retry Buffer, from which TLPs are re-transmitted when necessary:

❑ Copies of Transmitted TLPs must be stored in the Data Link Layer Retry Buffer, except for nullified TLPs.

When a replay is initiated, either due to reception of a Nak or due to REPLAY_TIMER expiration, the following rules describe the sequence of operations that must be followed:

❑ If all TLPs transmitted have been acknowledged (the Retry Buffer is empty), terminate replay, otherwise continue.

❑ Increment REPLAY_NUM.

- If REPLAY_NUM rolls over from 11b to 00b, the Transmitter signals the Physical Layer to retrain the Link, and waits for the completion of retraining before proceeding. This is a reported error associated with the Port (see Section 6.2).

  Note that Data Link Layer state, including the contents of the Retry Buffer, are not reset by this action unless the Physical Layer reports Physical LinkUp = 0 (causing the Data Link Control and Management State Machine to transition to the DL_Inactive state).

- If REPLAY_NUM does not roll over from 11b to 00b, continue.

❑ Block acceptance of new TLPs from the Transmit Transaction Layer.

❑ Complete transmission of any TLP currently being transmitted.

❑ Retransmit unacknowledged TLPs, starting with the oldest unacknowledged TLP and continuing in original transmission order

- Reset and restart REPLAY_TIMER when sending the last Symbol of the first TLP to be retransmitted

- Once all unacknowledged TLPs have been re-transmitted, return to normal operation.

- If any Ack or Nak DLLPs are received during a replay, the Transmitter is permitted to complete the replay without regard to the Ack or Nak DLLP(s), or to skip retransmission of any newly acknowledged TLPs.

  ♦ Once the Transmitter has started to resend a TLP, it must complete transmission of that TLP in all cases.

- Ack and Nak DLLPs received during a replay must be processed, and may be collapsed

  ♦ Example: If multiple Acks are received, only the one specifying the latest Sequence Number value must be considered – Acks specifying earlier Sequence Number values are effectively "collapsed" into this one

  ♦ Example: During a replay, Nak is received, followed by an Ack specifying a later Sequence Number – the Ack supercedes the Nak, and the Nak is ignored

- Note: Since all entries in the Retry Buffer have already been allocated space in the Receiver by the Transmitter's Flow Control gating logic, no further flow control synchronization is necessary.

❑ Re-enable acceptance of new TLPs from the Transmit Transaction Layer.

A replay can be initiated by the expiration of REPLAY_TIMER, or by the receipt of a Nak. The following rule covers the expiration of REPLAY_TIMER:

❑ If the Transmit Retry Buffer contains TLPs for which no Ack or Nak DLLP has been received, and (as indicated by REPLAY_TIMER) no Ack or Nak DLLP has been received for a period exceeding the time indicated in Table 3-4, the Transmitter initiates a replay.

This is a reported error associated with the Port (see Section 6.2).

The following formula defines the timeout count values for the REPLAY_TIMER. The values are specified according to the largest TLP payload size and Link width.

The values are measured at the Port of the TLP Transmitter, from last Symbol of TLP to First Symbol of TLP retransmission. The values are calculated using the formula (note – this is simply three times the Ack Latency value – see Section 3.5.3.1):

$$\left(\frac{(Max\_Payload\_Size + TLPOverhead) * AckFactor}{LinkWidth} + InternalDelay\right) * 3 + Rx\_L0s\_Adjustment$$

where

Max_Payload_Size	is the value in the Max_Payload_Size field of the Device Control register
TLP Overhead	represents the additional TLP components which consume Link bandwidth (header, LCRC, framing Symbols) and is treated here as a constant value of 28 Symbols
AckFactor	represents the number of maximum size TLPs which can be received before an Ack is sent, and is used to balance Link bandwidth efficiency and retry buffer size – the value varies according to Max_Payload_Size and Link width, and is included in Table 3-5
LinkWidth	is the operating width of the Link
InternalDelay	represents the internal processing delays for received TLPs and transmitted DLLPs, and is treated here as a constant value of 19 Symbol Times

Rx_L0s_Adjustment   equals the time required by the component's receive circuits to exit from L0s to L0 (as to receive an Ack DLLP from the other component on the Link) expressed in Symbol Times (see Section 4.2.6.6.1)

The values in Table 3-4 do not include this adjustment offset.

Thus, the timeout value for REPLAY_TIMER is the value given in Table 3-4 plus the Receiver L0s adjustment offset, described above.

Table 3-4: Unadjusted[17] REPLAY_TIMER Limits by Link Width and Max_Payload_Size (Symbol Times) Tolerance: -0%/+100%

		Link Operating Width						
		x1	**x2**	**x4**	**x8**	**x12**	**x16**	**x32**
Max_Payload_Size (bytes)	**128**	711	384	219	201	174	144	99
	256	1248	651	354	321	270	216	135
	512	1677	867	462	258	327	258	156
	1024	3213	1635	846	450	582	450	252
	2048	6285	3171	1614	834	1095	834	444
	4096	12429	6243	3150	1602	2118	1602	828

---

[17] The values in this table are determined using the formula shown above minus the "Rx_L0s_Adjustment" term.

📝 IMPLEMENTATION NOTE

Recommended Priority of Scheduled Transmissions

When multiple DLLPs of the same type are scheduled for transmission but have not yet been transmitted, it is possible in many cases to "collapse" them into a single DLLP. For example, if a scheduled Ack DLLP transmission is stalled waiting for another transmission to complete, and during this time another Ack is scheduled for transmission, it is only necessary to transmit the second Ack, since the information it provides will supercede the information in the first Ack.

In addition to any TLP from the Transaction Layer (or the Retry Buffer, if a retry is in progress), Multiple DLLPs of different types may be scheduled for transmission at the same time, and must be prioritized for transmission. The following list shows the preferred priority order for selecting information for transmission. Note that the priority of the vendor specific DLLP is not listed, as this is completely implementation specific, and there is no recommended priority. Note that this priority order is a guideline, and that in all cases a fairness mechanism is highly recommended to ensure that no type of traffic is blocked for an extended or indefinite period of time by any other type of traffic. Note that the Ack Latency value and REPLAY_TIMER limit specify requirements measured at the Port of the component, and the internal arbitration policy of the component must ensure that these externally measured requirements are met.

1) Completion of any transmission (TLP or DLLP) currently in progress (highest priority)

2) Nak DLLP transmissions

3) Ack DLLP transmissions scheduled for transmission as soon as possible due to:
   receipt of a duplicate TLP –OR–
   expiration of the Ack latency timer (see Section 3.5.3.1)

4) FC DLLP transmissions required to satisfy Section 2.6

5) Retry Buffer re-transmissions

6) TLPs from the Transaction Layer

7) FC DLLP transmissions other than those required to satisfy Section 2.6

8) All other DLLP transmissions (lowest priority)

Since Ack/Nak and Flow Control DLLPs affect TLPs flowing in the opposite direction across the Link, the TLP transmission mechanisms in the Data Link Layer are also responsible for Ack/Nak and Flow Control DLLPs received from the other component on the Link. These DLLPs are processed according to the following rules (see Figure 3-16):

❑ If the Physical Layer indicates a Receiver Error, discard any DLLP currently being received and free any storage allocated for the DLLP. Note that reporting such errors to software is done by the Physical Layer (and, therefore, not reported by the Data Link Layer).

❑ For all received DLLPs, the CRC value is checked by:

- applying the same algorithm used for calculation of transmitted DLLPs to the received DLLP, not including the 16 bit CRC field of the received DLLP

- comparing the calculated result with the value in the CRC field of the received DLLP

    ♦ if not equal, the DLLP is corrupt

- A corrupt received DLLP is discarded, and is a reported error associated with the Port (see Section 6.2).

❑ A received DLLP which is not corrupt, but which uses unsupported DLLP Type encodings is discarded without further action. This is not considered an error.

❑ Non-zero values in Reserved fields are ignored.

❑ Receivers must process all DLLPs received at the rate they are received
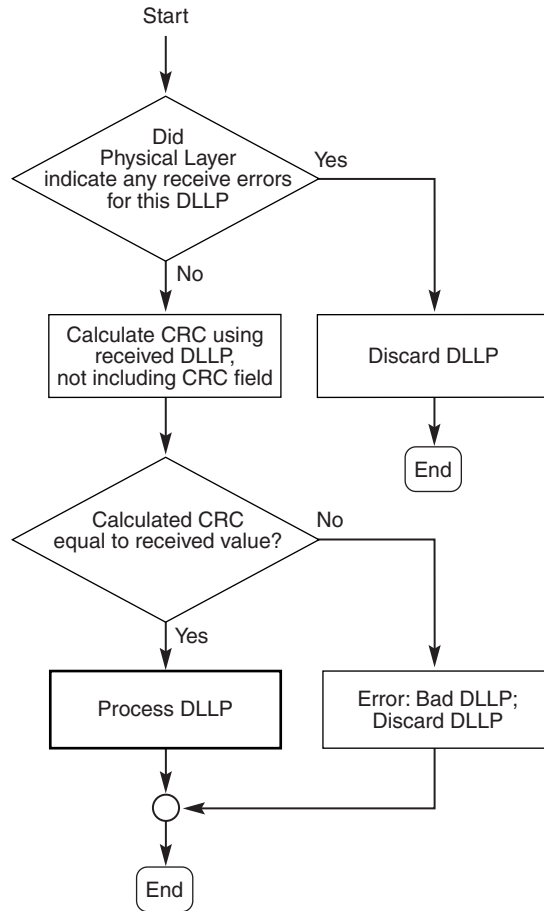
OM13789

Figure 3-16: Received DLLP Error Check Flowchart

❑ Received FC DLLPs are passed to the Transaction Layer

❑ Received PM DLLPs are passed to the component's power management control logic

❑ For Ack and Nak DLLPs, the following steps are followed (see Figure 3-17):

- If the Sequence Number specified by the AckNak_Seq_Num does not correspond to an
  unacknowledged TLP, or to the value in ACKD_SEQ, the DLLP is discarded

  ♦ This is a DL Layer Protocol Error which is a reported error associated with the Port (see
    Section 6.2).

  Note that it is not an error to receive an Ack DLLP when there are no outstanding
  unacknowledged TLPs, including the time between reset and the first TLP transmission,
  as long as the specified Sequence Number matches the value in ACKD_SEQ.

143

- If the AckNak_Seq_Num does not specify the Sequence Number of the most recently acknowledged TLP, then the DLLP acknowledges some TLPs in the retry buffer:

    ♦ Purge from the retry buffer all TLPs from the oldest to the one corresponding to the AckNak_Seq_Num

    ♦ Load ACKD_SEQ with the value in the AckNak_Seq_Num field

    ♦ Reset REPLAY_NUM and REPLAY_TIMER

- If the DLLP is a Nak, initiate a replay (see above)

    Note: Receipt of a Nak is not a reported error.
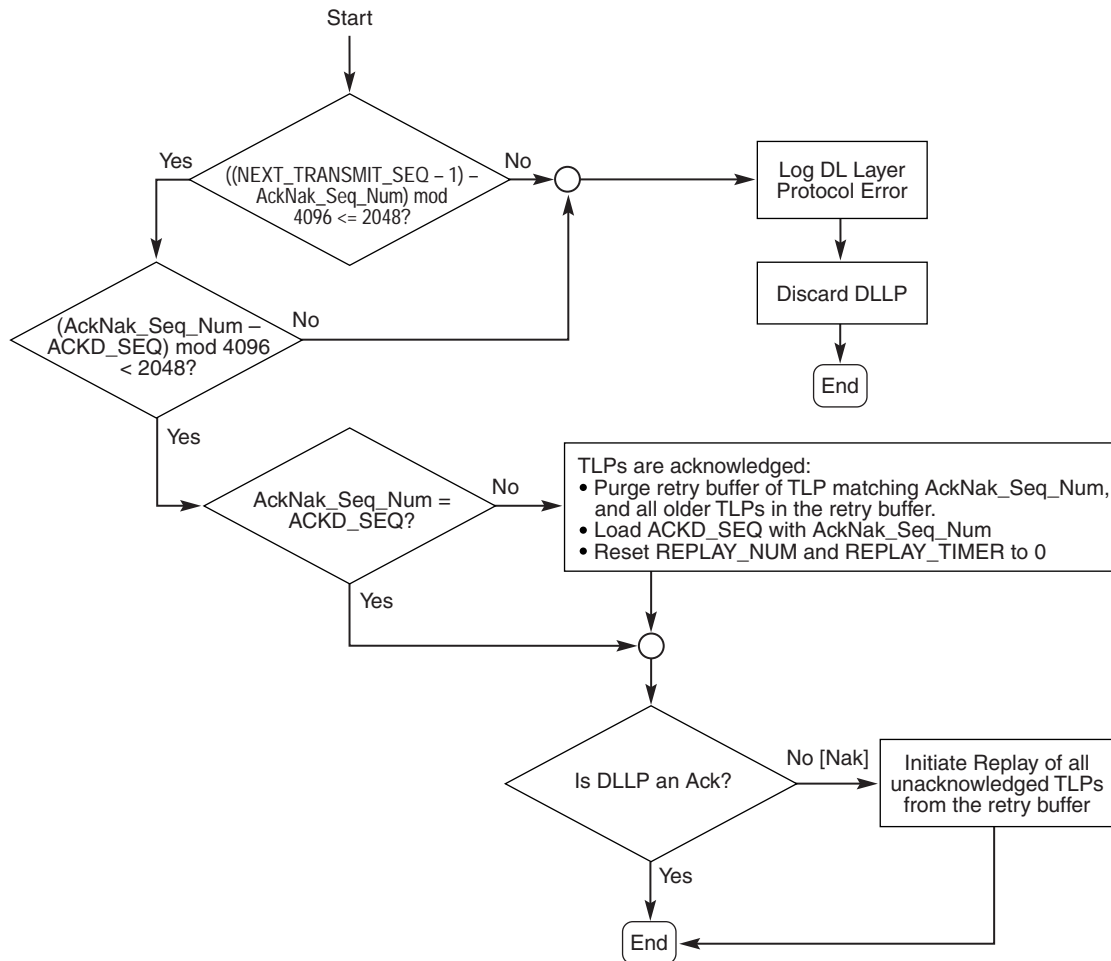
Figure 3-17: Ack/Nak DLLP Processing Flowchart

The following rules describe the operation of the Data Link Layer Retry Buffer, from which TLPs are re-transmitted when necessary:

❏ Copies of Transmitted TLPs must be stored in the Data Link Layer Retry Buffer

3.5.3.   LCRC and Sequence Number (TLP Receiver)

The TLP Receive path through the Data Link Layer (paths labeled 2 and 4 in Figure 3-1) processes TLPs received by the Physical Layer by checking the LCRC and sequence number, passing the TLP to the Receive Transaction Layer if OK and requesting a retry if corrupted.

The mechanisms used to check the TLP LCRC and the Sequence Number and to support Data Link Layer Retry are described in terms of conceptual "counters" and "flags." This description does not imply or require a particular implementation and is used only to clarify the requirements.

3.5.3.1.   LCRC and Sequence Number Rules (TLP Receiver)

The following counter, flag, and timer are used to explain the remaining rules in this section:

❑ The following 12-bit counter is used:

- NEXT_RCV_SEQ – Stores the expected Sequence Number for the next TLP

  ♦ Set to all 0's in DL_Inactive state

❑ The following flag is used:

- NAK_SCHEDULED

  ♦ Cleared when in DL_Inactive state

❑ The following timer is used:

- AckNak_LATENCY_TIMER – Counts time since an Ack or Nak DLLP was scheduled for transmission

  ♦ Set to 0 in DL_Inactive state

  ♦ Restart from 0 each time an Ack or Nak DLLP is scheduled for transmission; Reset to 0 when all TLPs received have been acknowledged with an Ack DLLP

  ♦ If there are initially no unacknowledged TLPs and a TLP is then received, the AckNak_LATENCY_TIMER starts counting only when the TLP has been forwarded to the Receive Transaction Layer

The following rules are applied in sequence to describe how received TLPs are processed, and what events trigger the transmission of Ack and Nak DLLPs (see Figure 3-18):

❑ If the Physical Layer indicates a Receiver Error, discard any TLP currently being received and free any storage allocated for the TLP. Note that reporting such errors to software is done by the Physical Layer (and so are not reported by the Data Link Layer).

- If a TLP was being received at the time the Receive Error was indicated and the NAK_SCHEDULED flag is clear,

  ♦ schedule a Nak DLLP for transmission

  ♦ set the NAK_SCHEDULED flag

❑ If the Physical Layer reports that the received TLP end framing Symbol was EDB, and the LCRC is the logical NOT of the calculated value, discard the TLP and free any storage allocated for the TLP. This is not considered an error.

❑ The LCRC value is checked by:

₅
- applying the same algorithm used for calculation (above) to the received TLP, not including the 32-bit LCRC field of the received TLP

- comparing the calculated result with the value in the LCRC field of the received TLP

    ♦ if not equal, the TLP is corrupt - discard the TLP and free any storage allocated for the TLP

₁₀
        ■ If the NAK_SCHEDULED flag is clear,

            ◘ schedule a Nak DLLP for transmission

            ◘ set the NAK_SCHEDULED flag

        This is a reported error associated with the Port (see Section 6.2).

❑ If the TLP Sequence Number is not equal to the expected value, stored in NEXT_RCV_SEQ:

₁₅
- discard the TLP and free any storage allocated for the TLP

- If the TLP Sequence Number satisfies the following equation:

    (NEXT_RCV_SEQ - TLP Sequence Number) mod 4096 <= 2048

    the TLP is a duplicate, and an Ack DLLP is scheduled for transmission (per transmission priority rules)

₂₀
- Otherwise, the TLP is out of sequence (indicating one or more lost TLPs):

    ♦ if the NAK_SCHEDULED flag is clear,

        ■ schedule a Nak DLLP for transmission

        ■ set the NAK_SCHEDULED flag

        ■ report TLP missing

₂₅
    This is a reported error associated with the Port (see Section 6.2).

❑ If the TLP Sequence Number is equal to the expected value stored in NEXT_RCV_SEQ:

- The Reserved bits, Sequence Number, and LCRC are removed and the remainder of the TLP is forwarded to the Receive Transaction Layer

    ♦ The Data Link Layer indicates the start and end of the TLP to the Transaction Layer
₃₀ while transferring the TLP

        ■ The Data Link Layer treats the TLP as a "black box" and does not process or modify the contents of the TLP

    ♦ Note that the Receiver Flow Control mechanisms do not account for any received TLPs until the TLP(s) are forwarded to the Receive Transaction Layer

₃₅
- NEXT_RCV_SEQ is incremented

- If set, the NAK_SCHEDULED flag is cleared
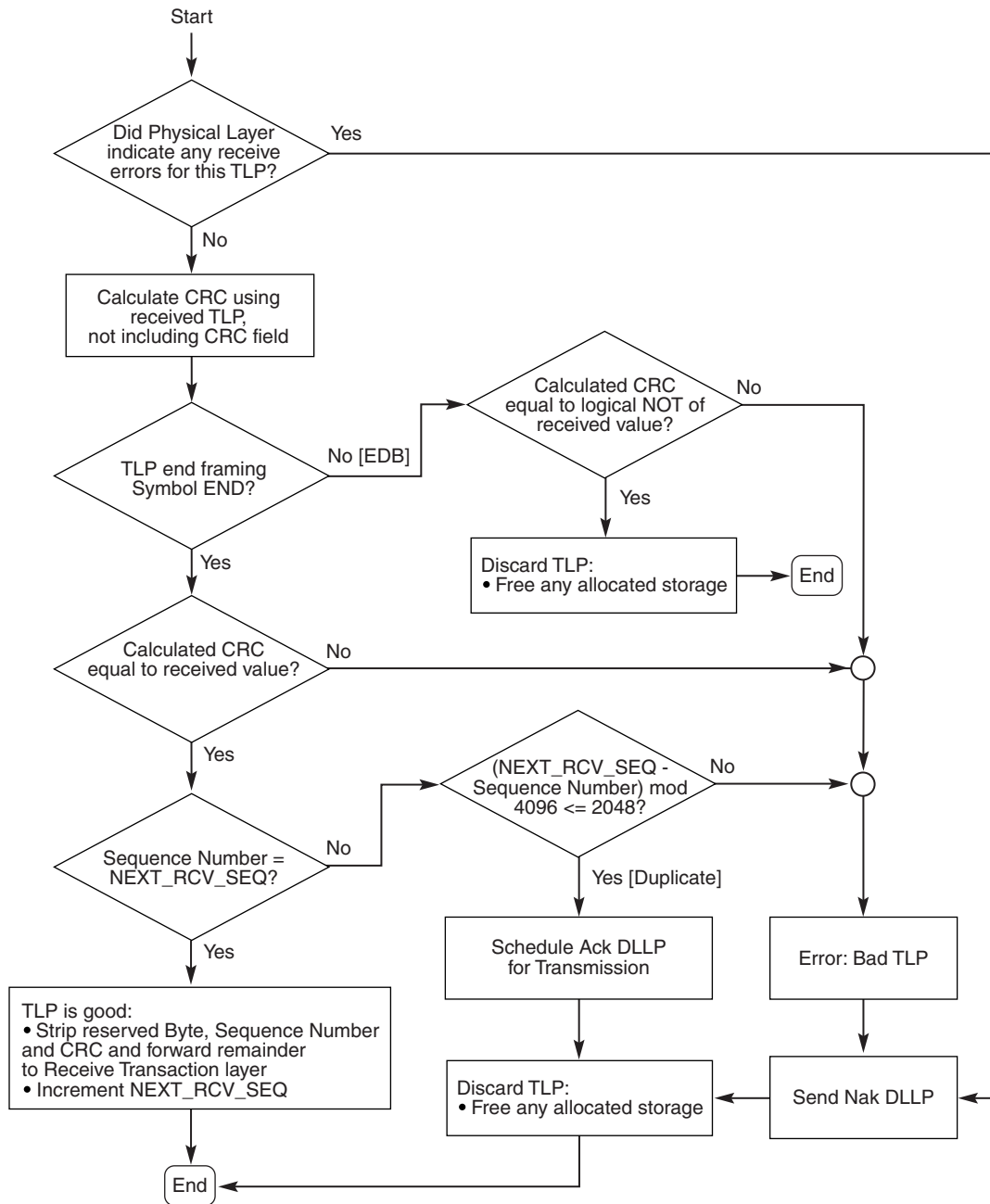
OM13791

Figure 3-18: Receive Data Link Layer Handling of TLPs

❑ In addition to the other requirements for sending Ack DLLPs, an Ack or Nak DLLP must be transmitted when all of the following conditions are true:

- The Data Link Control and Management State Machine is in the DL_Active state

- TLPs have been accepted, but not yet acknowledged by sending an Ack DLLP

- The AckNak_LATENCY_TIMER reaches or exceeds the value specified in Table 3-5

❑ Data Link Layer Ack DLLPs may be Transmitted more frequently than required

❑ Data Link Layer Ack and Nak DLLPs specify the value (NEXT_RCV_SEQ - 1) in the AckNak_Seq_Num field

Table 3-5 defines the threshold values for the AckNak_LATENCY_TIMER timer, which for any specific case is called the Ack Latency. The values are specified according to the largest TLP payload size and Link width. The values are measured at the Port of the TLP Receiver, starting with the time the last Symbol of a TLP is received to the first Symbol of the Ack/Nak DLLP being transmitted. The values are calculated using the formula:

$$\frac{(Max\_Payload\_Size + TLPOverhead) * AckFactor}{LinkWidth} + InternalDelay + Tx\_L0s\_Adjustment$$

where

Max_Payload_Size	is the value in the Max_Payload_Size field of the Device Control register.
TLP Overhead	represents the additional TLP components which consume Link bandwidth (header, LCRC, framing Symbols) and is treated here as a constant value of 28 Symbols.
AckFactor	represents the number of maximum size TLPs which can be received before an Ack is sent, and is used to balance Link bandwidth efficiency and retry buffer size – the value varies according to Max_Payload_Size and Link width, and is defined in Table 3-5.
LinkWidth	is the operating width of the Link.
InternalDelay	represents the internal processing delays for received TLPs and transmitted DLLPs, and is treated here as a constant value of 19 Symbol Times.
Tx_L0s_Adjustment	if L0s is enabled, the time required for the Transmitter to exit L0s (see Section 4.2.6.6.2), expressed in Symbol Times, or 0 if L0s is not enabled.
	Note that the setting of the Extended Synch bit of the Link Control register affects the exit time from L0s to L0, and must be taken into account in this adjustment.
	The values in Table 3-5 do not include this adjustment offset.

Thus, the Ack Latency is the value given in Table 3-5 plus the Transmitter L0s adjustment offset, described above.

Table 3-5: Unadjusted[18] Ack Transmission Latency Limit and AckFactor by Link Width and Max Payload (Symbol Times)

		Link Operating Width						
		x1	**x2**	**x4**	**x8**	**x12**	**x16**	**x32**
Max_Payload_Size (bytes)	**128**	237 AF = 1.4	128 AF = 1.4	73 AF = 1.4	67 AF = 2.5	58 AF = 3.0	48 AF = 3.0	33 AF = 3.0
	256	416 AF = 1.4	217 AF = 1.4	118 AF = 1.4	107 AF = 2.5	90 AF = 3.0	72 AF = 3.0	45 AF = 3.0
	512	559 AF = 1.0	289 AF = 1.0	154 AF = 1.0	86 AF = 1.0	109 AF = 2.0	86 AF = 2.0	52 AF = 2.0
	1024	1071 AF = 1.0	545 AF = 1.0	282 AF = 1.0	150 AF = 1.0	194 AF = 2.0	150 AF = 2.0	84 AF = 2.0
	2048	2095 AF = 1.0	1057 AF = 1.0	538 AF = 1.0	278 AF = 1.0	365 AF = 2.0	278 AF = 2.0	148 AF = 2.0
	4096	4143 AF = 1.0	2081 AF = 1.0	1050 AF = 1.0	534 AF = 1.0	706 AF = 2.0	534 AF = 2.0	276 AF = 2.0

---

[18] The values in this table are determined using the formula shown above minus the "Tx_L0s_Adjustment" term.

✎ IMPLEMENTATION NOTE

Retry Buffer Sizing

The Retry Buffer should be large enough to ensure that under normal operating conditions, transmission is never throttled because the retry buffer is full. In determining the optimal buffer size, one must consider the Ack Latency value, any differences between the actual implementation and the internal processing delay used to generate these values, and the delays caused by the physical Link interconnect.

The Receiver L0s exit latency (see Section 4.6.2.2.1) should also be accounted for, as is demonstrated with the following example using components A and B:

❑ A exits L0s on its Transmit path to B and starts transmitting a long burst of write Requests to B

❑ B initiates L0s exit on its Transmit path to A, but the L0s exit time required by A's Receiver is large

❑ Meanwhile, B is unable to send Ack DLLPs to A, and A stalls due to lack of Retry Buffer space

❑ The Transmit path from B to A returns to L0, B transmits an Ack DLLP to A, and the stall is resolved

This stall can be avoided by matching the size of a component's Retry Buffer to the L0s exit latency of the component's Receiver, or, conversely, matching the Receiver L0s exit latency to the desired size of the Retry Buffer.

AckFactor values were chosen to allow implementations to achieve good performance without requiring an uneconomically large retry buffer. To enable consistent performance across a general purpose interconnect with differing implementations and applications, it is necessary to set the same requirements for all components without regard to the application space of any specific component. If a component does not require the full transmission bandwidth of the Link, it may reduce the size of its retry buffer below the minimum size required to maintain available retry buffer space with the Ack Latency values specified.

Note that the Ack Latency values specified ensure that the range of permitted outstanding Sequence Numbers will never be the limiting factor causing transmission stalls.

4

4.   Physical Layer Specification

4.1.   Introduction

The Physical Layer isolates the Transaction and Data Link Layers from the signaling technology
used for Link data interchange. The Physical Layer is divided into the logical and electrical
functional sub-blocks (see Figure 4-1).
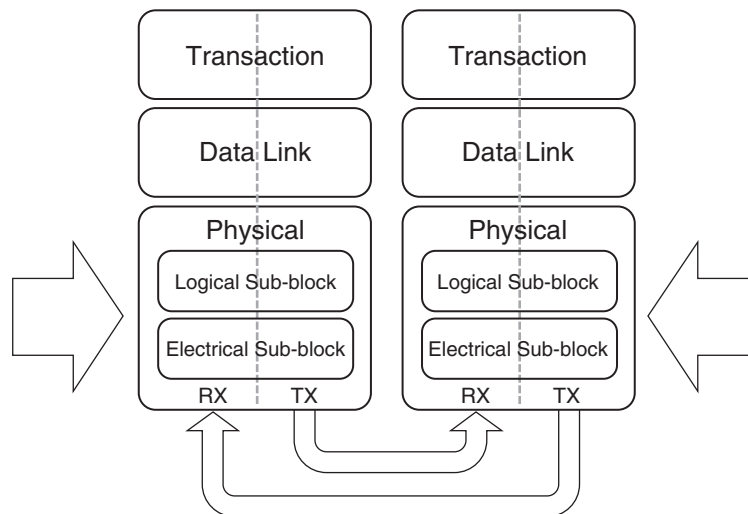
OM13792

Figure 4-1: Layering Diagram Highlighting Physical Layer

4.2.   Logical Sub-block

The logical sub-block has two main sections: a Transmit section that prepares outgoing information
passed from the Data Link Layer for transmission by the electrical sub-block, and a Receiver section
that identifies and prepares received information before passing it to the Data Link Layer.

The logical sub-block and electrical sub-block coordinate the state of each transceiver through a
status and control register interface or functional equivalent. The logical sub-block directs control
and management functions of the Physical Layer.

4.2.1. Symbol Encoding

PCI Express uses an 8b/10b transmission code. The definition of this transmission code is identical to that specified in ANSI X3.230-1994, clause 11 (and also IEEE 802.3z, 36.2.4). Using this scheme, 8-bit data characters are treated as three bits and five bits mapped onto a 4-bit code group and a 6-bit code group, respectively. The control bit in conjunction with the data character is used to identify when to encode one of the 12 Special Symbols included in the 8b/10b transmission code. These code groups are concatenated to form a 10-bit Symbol. As shown in Figure 4-2, ABCDE maps to abcdei and FGH maps to fghj.
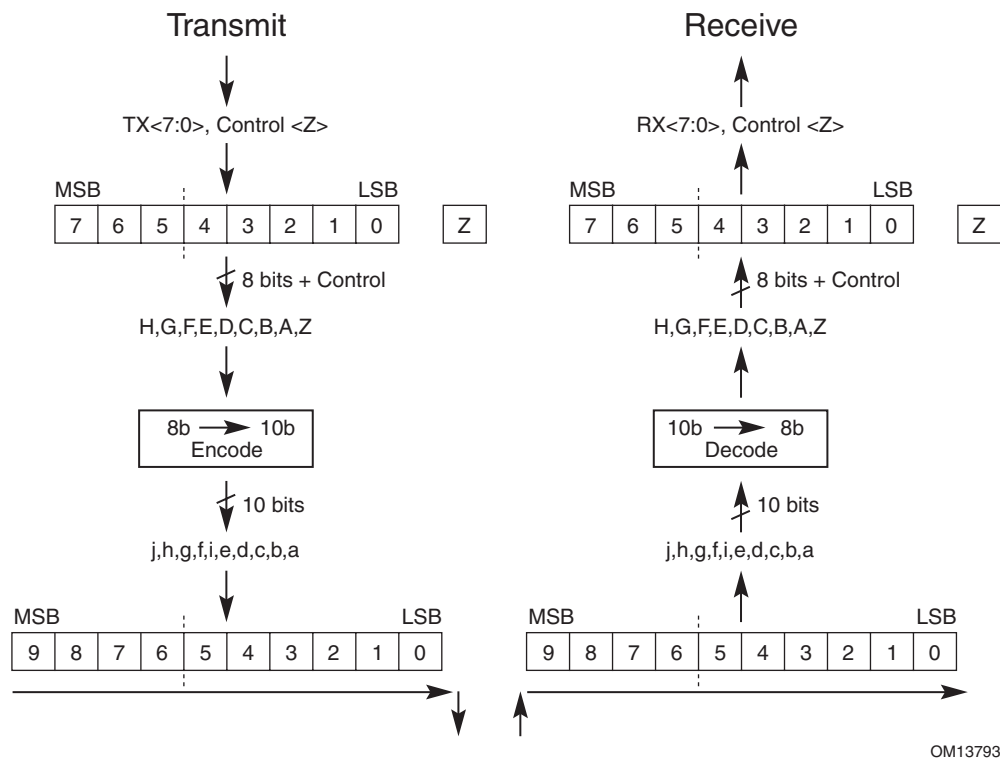
Figure 4-2: Character to Symbol Mapping

4.2.1.1. Serialization and De-serialization of Data

The bits of a Symbol are placed on a Lane starting with bit "a" and ending with bit "j." Examples are shown in Figure 4-3 and Figure 4-4.
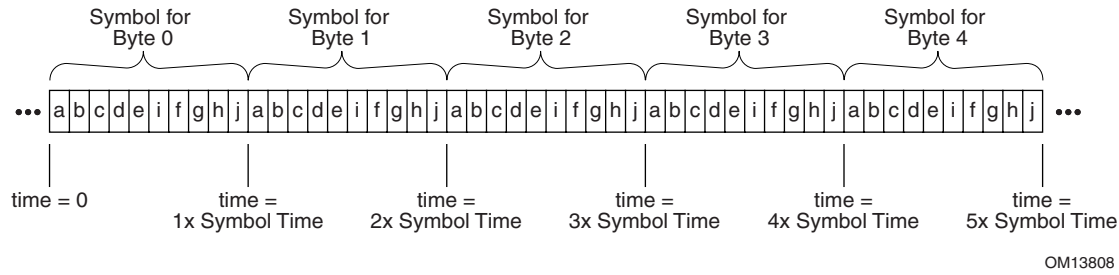
Figure 4-3: Bit Transmission Order on Physical Lanes - x1 Example
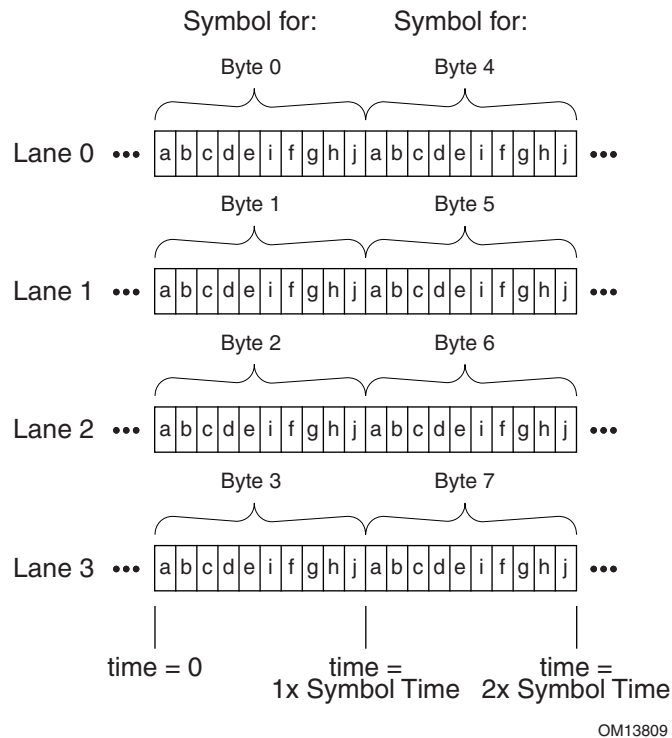
Figure 4-4: Bit Transmission Order on Physical Lanes - x4 Example

4.2.1.2. *Special Symbols for Framing and Link Management (K Codes)*

The 8b/10b encoding scheme used by PCI Express provides Special Symbols that are distinct from the Data Symbols used to represent Characters. These Special Symbols are used for various Link Management mechanisms described later in this chapter. Special Symbols are also used to frame DLLPs and TLPs, using distinct Special Symbols to allow these two types of Packets to be quickly and easily distinguished.

Table 4-1 shows the Special Symbols used for PCI Express and provides a brief description for the use of each. The use of these Symbols will be discussed in greater detail in following sections.

Table 4-1: Special Symbols

Encoding	Symbol	Name	Description
K28.5	COM	Comma	Used for Lane and Link initialization and management
K27.7	STP	Start TLP	Marks the start of a Transaction Layer Packet
K28.2	SDP	Start DLLP	Marks the start of a Data Link Layer Packet
K29.7	END	End	Marks the end of a Transaction Layer Packet or a Data Link Layer Packet
K30.7	EDB	EnD Bad	Marks the end of a nullified TLP
K23.7	PAD	Pad	Used in Framing and Link Width and Lane ordering negotiations
K28.0	SKP	Skip	Used for compensating for different bit rates for two communicating Ports
K28.1	FTS	Fast Training Sequence	Used within an ordered set to exit from L0s to L0
K28.3	IDL	Idle	Symbol used in the Electrical Idle ordered set
K28.4			Reserved
K28.6			Reserved
K28.7			Reserved

4.2.1.3. 8b/10b Decode Rules

The Symbol tables for the valid 8b/10b codes are given in Appendix B. These tables have one column for the positive disparity and one column for the negative disparity.

A Transmitter may pick any disparity when first transmitting differential data after being in an
Electrical Idle state. The Transmitter must then follow proper 8b/10b encoding rules until the next Electrical Idle state is entered.

The initial disparity for a Receiver that detects an exit from Electrical Idle is set to the disparity of the first character used to obtain Symbol lock. Disparity may also be-reinitialized if Symbol lock is lost and regained during the transmission of differential information due to an implementation
specific number of errors. All following received Symbols after the initial disparity is set must be in the found in the proper column corresponding to the current running disparity.

If a received Symbol is found in the column corresponding to the incorrect running disparity or if the Symbol does not correspond to either column, the Physical Layer must notify the Data Link Layer that the received Symbol is invalid. This is a Receiver Error, and is a reported error associated
with the Port (see Section 6.2).

4.2.2.   Framing and Application of Symbols to Lanes

There are two classes of framing and application of Symbols to Lanes. The first class is the ordered sets and the second is TLP and DLLP packet. Ordered sets are always transmitted serially on each Lane, such that a full ordered set appears simultaneously on all Lanes of a multi-Lane Link.

5   The Framing mechanism uses Special Symbol K28.2 "SDP" to start a DLLP and Special Symbol K27.7 "STP" to start a TLP. The Special Symbol K29.7 "END" is used to mark the end of either a TLP or a DLLP.

The conceptual stream of Symbols must be mapped from its internal representation, which is implementation dependent, onto the external Lanes. The Symbols are mapped onto the Lanes such

10   that the first Symbol (representing Character 0) is placed onto Lane 0; the second is placed onto Lane 1; etc. The x1 Link represents a degenerate case and the mapping is trivial, with all Symbols placed onto the single Lane in order.

When no packet information or special ordered sets are being transmitted, the Transmitter is in the Logical Idle state. During this time idle data must be transmitted. The idle data must consist of the

15   data byte 0 (00 Hexadecimal), scrambled according to the rules of Section 4.2.3 and 8b/10b encoded according to the rules of Section 4.2.1, in the same way that TLP and DLLP data characters are scrambled and encoded. Likewise, when the Receiver is not receiving any packet information or special ordered sets, the Receiver is in Logical Idle and shall receive idle data as described above. During transmission of the idle data, the SKP ordered set must continue to be transmitted as

20   specified in Section 4.2.7.

4.2.2.1. Framing and Application of Symbols to Lanes – Rules

In this section, "placed" is defined to mean a requirement on the Transmitter to put the Symbol into the proper Lane of a Link.

❑ TLPs must be framed by placing an STP Symbol at the start of the TLP and an END Symbol or
25   EDB Symbol at the end of the TLP (see Figure 4-5).

❑ DLLPs must be framed by placing an SDP Symbol at the start of the DLLP and an END Symbol at the end of the DLLP (see Figure 4-6).

❑ Logical Idle is defined to be a period of one or more Symbol Times when no information: TLPs, DLLPs or any type of Special Symbol is being Transmitted/Received. Unlike Electrical Idle,
30   during Logical Idle the Idle character (00h) is being transmitted and received.

• When the Transmitter is in Logical Idle, the Idle data character (00h) shall be transmitted on all Lanes. This is scrambled according to the rules in Section 4.2.3.

• Receivers must ignore incoming Logical Idle data, and must not have any dependency other than scramble sequencing on any specific data patterns.

35 ❑ For Links wider than x1, the STP Symbol (representing the start of a TLP) must be placed in Lane 0 when starting Transmission of a TLP from a Logical Idle Link condition.

❑ For Links wider than x1, the SDP Symbol (representing the start of a DLLP) must be placed in Lane 0 when starting Transmission of a DLLP from a Logical Idle Link condition.

❑ The STP Symbol must not be placed on the Link more frequently than once per Symbol Time.

❑ The SDP Symbol must not be placed on the Link more frequently than once per Symbol Time.

❑ As long as the above rules are satisfied, TLP and DLLP Transmissions are permitted to follow each other successively.

5 ❑ One STP Symbol and one SDP Symbol may be placed on the Link in the same Symbol Time.

- Note: Links wider than x4 can have STP and SDP Symbols placed in Lane 4*N, where N is a positive integer. For example, for x8, STP and SDP Symbols can be placed in Lanes 0 and 4; and for x16, STP and SDP Symbols can be placed in Lanes 0, 4, 8, or 12.

❑ For xN Links where N is 8 or more, if an END or EDB Symbol is placed in a Lane K, where K
10 does not equal N-1, and is not followed by a STP or SDP Symbol in Lane K+1 (i.e., there is no TLP or DLLP immediately following), then PAD Symbols must be placed in Lanes K+1 to Lane N-1.

- Note: For example, on a x8 Link, if END or EDB is placed in Lane 3, PAD must be placed in Lanes 4 to 7, when not followed by STP or SDP.

15 ❑ The EDB Symbol is used to mark the end of a nullified TLP. Refer to Section 3.5.2.1 for information on the usage of EDB.

❑ Receivers may optionally check for violations of the rules of this section. Any such violation is a Receiver Error, and is a reported error associated with the Port (see Section 6.2).
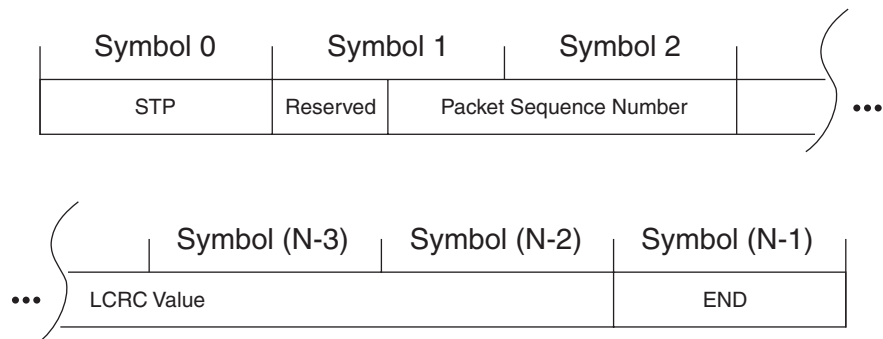
OM13794

Figure 4-5: TLP with Framing Symbols Applied
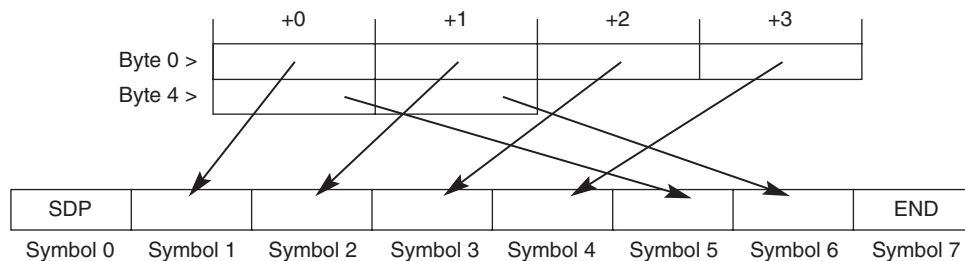
OM13795

Figure 4-6: DLLP with Framing Symbols Applied

STP Framing Symbol
added by Physical Layer

Reserved bits and
Sequence Number
added by
Data Link Layer

TLP generated by
Transaction Layer

LCRC added by
Data Link Layer

END Framing Symbol
added by Physical Layer
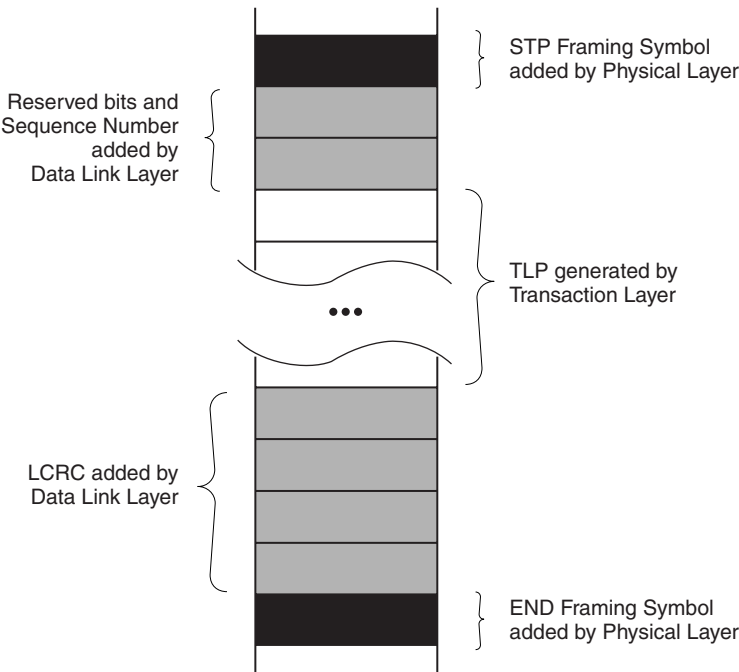
OM13796

Figure 4-7: Framed TLP on a x1 Link

Lane 0   Lane 1

STP/END Framing Symbols-
Physical Layer

Sequence Number/LCRC-
Data Link Layer

TLP - Transaction Layer
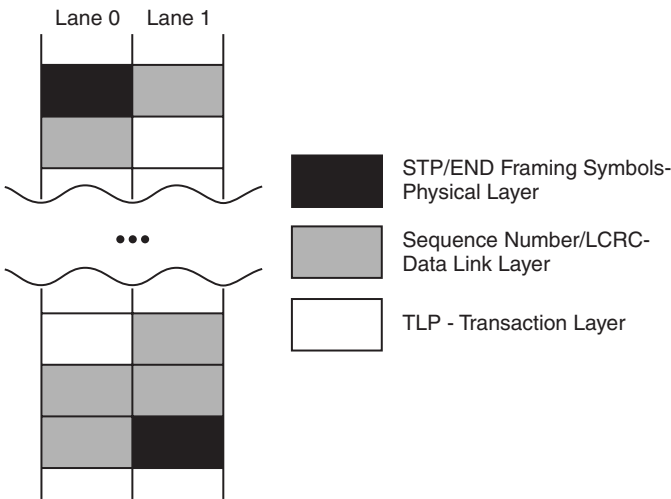
OM13797

Figure 4-8: Framed TLP on a x2 Link
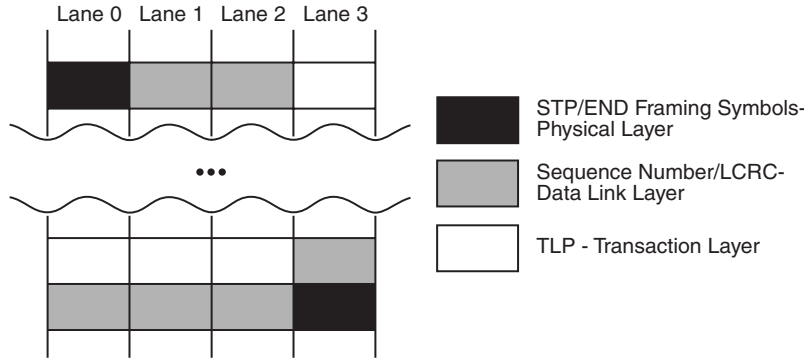
OM13798

Figure 4-9: Framed TLP on a x4 Link

4.2.3.   Data Scrambling

The scrambling function can be implemented with one or many Linear Feedback Shift Registers (LFSRs) on a multi-Lane Link. When there is more than one Transmit LFSR per Link, these must operate in concert, maintaining the same simultaneous (see Table 4-5, Lane-to-Lane Output Skew) value in each LFSR. When there is more than one Receive LFSR per Link, these must operate in concert, maintaining the same simultaneous (see Table 4-6, Total Skew) value in each LFSR. Regardless of how they are implemented, LFSRs must interact with data on a Lane-by-Lane basis as if there was a separate LFSR as described here for each Lane within that Link. On the Transmit side, scrambling is applied to characters prior to the 8b/10b encoding. On the Receive side, de-scrambling is applied to characters after 8b/10b decoding.

The LFSR is graphically represented in Figure 4-10. Scrambling or unscrambling is performed by serially XORing the 8-bit (D0-D7) character with the 16-bit (D0-D15) output of the LFSR. An output of the LFSR, D15, is XORed with D0 of the data to be processed. The LFSR and data register are then serially advanced and the output processing is repeated for D1 through D7. The LFSR is advanced after the data is XORed. The LFSR implements the polynomial:

$G(X)=X^{16}+X^5+X^4+X^3+1$

The mechanism(s) and/or interface(s) utilized by the Data Link Layer to notify the Physical Layer to disable scrambling is implementation specific and beyond the scope of this specification.

The data scrambling rules are the following:

❑ The COM character initializes the LFSR.

❑ The LFSR value is advanced eight serial shifts for each character except the SKP.

❑ All data characters (D codes) except those within a Training Sequence ordered sets (TS1, TS2) and the Compliance Pattern (see Section 4.2.8) are scrambled.

❑ All special characters (K codes) are not scrambled.

❑ The initialized value of an LFSR seed (D0-D15) is FFFFh. Immediately after a COM exits the Transmit LFSR, the LFSR on the Transmit side is initialized. Every time a COM enters the Receive LFSR on any Lane of that Link, the LFSR on the Receive side is initialized.

❑ Scrambling can only be disabled at the end of Configuration (see Section 4.2.6.3.5).

❑ Scrambling does not apply to a Loopback Slave.

❑ Scrambling is always enabled in Detect by default.

📌 IMPLEMENTATION NOTE

Disabling Scrambling

Disabling scrambling is intended to help simplify test and debug equipment. Control of the exact data patterns is useful in a test and debug environment. Since scrambling is reset at the Physical Layer there is no reasonable way to reliably control the state of the data transitions through software. Thus, the Disable Scrambling bit is provided for these purposes.

The mechanism(s) and/or interface(s) utilized by the Data Link Layer to notify the Physical Layer to disable scrambling is component implementation specific and beyond the scope of this specification.

For more information on scrambling, see Appendix C.

OM13799A
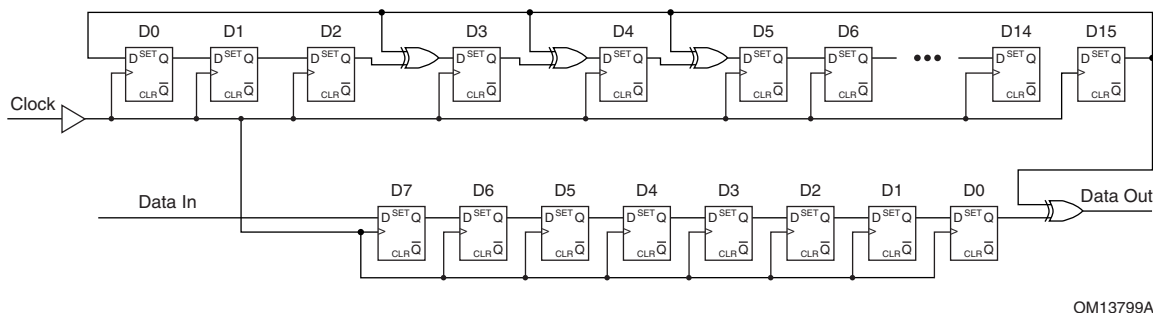
Figure 4-10: LFSR with Scrambling Polynomial

4.2.4. Link Initialization and Training

This section defines the Physical Layer control process that configures and initializes each Link for normal operation. This section covers the following functions:

❑ Configuring and initializing the Link.

❑ Supporting normal packet transfers.

❑ Supported state transitions when recovering from Link errors.

❑ Restarting a Port from low power states.

The following are discovered and determined during the training process:

❑ Link width.

❑ Link data rate.[19]

❑ Lane reversal.

❑ Polarity inversion.

Training does:

5 ❑ Link data rate[20] negotiation.

❑ Bit lock per Lane.

❑ Lane polarity.

❑ Symbol lock per Lane.

❑ Lane ordering within a Link.

10 ❑ Link width negotiation.

❑ Lane-to-Lane de-skew within a multi-Lane Link.

Receivers may optionally check for violations of the Link Initialization and Training Protocols. If such checking is implemented, any violation is a Training Error. A Training Error is a reported error associated with the Port (see Section 6.2). A Training Error is considered fatal to the Link.

15 ## 4.2.4.1. Training Sequence Ordered Sets

Training sequences are composed of ordered sets used for initializing bit alignment, Symbol alignment and to exchange Physical Layer parameters. Training sequence ordered sets are never scrambled but are always 8b/10b encoded.

Training sequences (TS1 or TS2) are transmitted consecutively and can only be interrupted by SKP
20 ordered sets (see Section 4.2.7).

The Training control bits for Hot Reset, Disable Link, and Enable Loopback are mutually exclusive, only one of these bits may be set at a time as well as transmitted on all Lanes in a configured (all Lanes in L0) or possible (all Lanes in Configuration) Link. If more than one of the Hot Reset, Disable Link, or Enable Loopback bits are set at the same time, the Link behavior is undefined.

---

[19] This specification only defines one data rate. Future revisions will define additional rates.

[20] This specification defines the mechanism for negotiating the Link operational bit rate to the highest supported operational data rate.

Table 4-2: TS1 Ordered Set

Symbol Number	Allowed Values	Encoded Values	Description
0		K28.5	COMMA code group for Symbol alignment
1	0 – 255	D0.0 - D31.7, K23.7	Link Number within component
2	0 – 31	D0.0 - D31.0, K23.7	Lane Number within Port
3	0 – 255	D0.0 - D31.7	N_FTS. This is the number of fast training ordered sets required by the Receiver to obtain reliable bit and Symbol lock.
4	2	D2.0	Data Rate Identifier Bit 0 – Reserved, set to 0 Bit 1 = 1, generation 1 (2.5 Gb/s) data rate supported Bit 2:7 – Reserved, set to 0
5	Bit 0 = 0, 1 Bit 1 = 0, 1 Bit 2 = 0, 1 Bit 3 = 0, 1 Bit 4:7 = 0	D0.0, D1.0, D2.0, D4.0, D8.0	Training Control Bit 0 – Hot Reset Bit 0 = 0, De-assert Bit 0 = 1, Assert Bit 1 – Disable Link Bit 1 = 0, De-assert Bit 1 = 1, Assert Bit 2 – Loopback Bit 2 = 0, De-assert Bit 2 = 1, Assert Bit 3 – Disable Scrambling Bit 3 = 0, De-assert Bit 3 = 1, Assert Bit 4:7 – Reserved Set to 0
6 – 15		D10.2	TS1 Identifier

Table 4-3: TS2 Ordered Set

Symbol Number	Allowed Values	Encoded Values	Description
0		K28.5	COMMA code group for Symbol alignment
1	0 – 255	D0.0 - D31.7, K23.7	Link Number within component
2	0 – 31	D0.0 - D31.0, K23.7	Lane Number within Port
3	0 – 255	D0.0 - D31.7	N_FTS. This is the number of fast training ordered sets required by the Receiver to obtain reliable bit and Symbol lock.
4	2	D2.0	Data Rate Identifier Bit 0 – Reserved, set to 0 Bit 1 = 1, generation 1 (2.5 Gb/s) data rate supported Bit 2:7 – Reserved, set to 0
5	Bit 0 = 0, 1 Bit 1 = 0, 1 Bit 2 = 0, 1 Bit 3 = 0, 1 Bit 4:7 = 0	D0.0, D1.0, D2.0, D4.0, D8.0	Training Control <u>Bit 0 – Hot Reset</u> Bit 0 = 0, De-assert Bit 0 = 1, Assert <u>Bit 1 – Disable Link</u> Bit 1 = 0, De-assert Bit 1 = 1, Assert <u>Bit 2 – Loopback</u> Bit 2 = 0, De-assert Bit 2 = 1, Assert <u>Bit 3 – Disable Scrambling</u> Bit 3 = 0, De-assert Bit 3 = 1, Assert <u>Bit 4:7 – Reserved</u> Set to 0
6 – 15		D5.2	TS2 Identifier

4.2.4.2.   Lane Polarity Inversion

During the training sequence, the Receiver looks at Symbols 6-15 of TS1 and TS2 as the indicator of Lane polarity inversion (D+ and D- are swapped). If Lane polarity inversion occurs, the TS1 Symbols 6-15 received will be D21.5 as opposed to the expected D10.2. Similarly, if Lane polarity inversion occurs, Symbols 6-15 of the TS2 ordered set will be D26.5 as opposed to the expected D5.2. This provides the clear indication of Lane polarity inversion.

If polarity inversion is detected the Receiver must invert the received data. The Transmitter must never invert the transmitted data. Support for Lane Polarity Inversion is required on all PCI Express Receivers across all Lanes independently.

4.2.4.3.   Fast Training Sequence (FTS)

FTS is the mechanism that is used for bit and Symbol lock when transitioning from L0s to L0. The FTS is used by the Receiver to detect the exit from Electrical Idle and align the Receiver's bit/Symbol receive circuitry to the incoming data. See Section 4.2.5 for a description of L0 and L0s.

A single FTS training sequence is an ordered set composed of one K28.5 (COM) Symbol and three K28.1 Symbols. The maximum number of FTS ordered sets (N_FTS) that a component can request is 255, providing a bit time lock of $4 * 255 * 10 * UI$. 4096 FTS ordered sets must be sent when the Extended Synch bit is set in order to provide external Link monitoring tools with enough time to achieve bit and framing synchronization. SKP ordered sets must be scheduled and transmitted between FTS ordered sets as necessary to meet the definitions in Section 4.2.7 with the exception that no SKP ordered sets can be transmitted during the first N_FTS FTS ordered sets. A single SKP ordered set is always sent after the last FTS is transmitted. Note that it is possible that two SKP ordered sets can be transmitted back to back (one SKP ordered set to signify the completion of the 4096 FTSs and one scheduled and transmitted to meet the definitions described in Section 4.2.7).

N_FTS defines the number of FTS ordered sets that must be transmitted when transitioning from L0s to L0. At the generation 1 data rate, the value that can be requested by a component corresponds to a Symbol lock time of 16 ns (N_FTS set to 0 and one SKP ordered set) to ~4 µs (N_FTS set to 255), except when the Extended Synch bit is set, which requires the transmission of 4096 FTS ordered sets resulting in a bit lock time of 64 µs. Note that the N_FTS value reported by a component may change; for example, due to software modifying the value in the Common Clock Configuration bit (Section 7.8.7).

If the N_FTS period of time expires before the Receiver obtains bit, Symbol, and Lane-to-Lane de-skew on all Lanes of the configured Link, the Receiver must transition to the Recovery. This sequence is detailed in the LTSSM in Section 4.2.5.

4.2.4.4. Link Error Recovery

❏ Link Errors are defined as 8b/10b decode errors, loss of Symbol lock, or loss of Lane-to-Lane de-skew.

- Note: 8b/10b decode errors trigger a Receiver Error (see Table 4-4), which is a reported error associated with the Port (see Section 6.2).

❏ On a configured Link, which is in L0, error recovery will at a minimum be managed in a Layer above the Physical Layer (as described in Section 3.5) by directing the Link to transition to Recovery.

- Note: Link Errors may also result in the Physical Layer initiating a LTSSM state transition from L0 to Recovery.

❏ All LTSSM states other than L0 make progress[21] when Link Errors occur.

- Note: Link errors that occur, while in LTSSM states other than L0, must not result in the Physical Layer initiating a LTSSM state transition.

❏ If a Lane detects an implementation specific number of 8b/10b errors, Symbol lock must be verified or re-established as soon as possible.[22]

4.2.4.5. Reset

Reset is described from a system point of view in Section 6.6.

4.2.4.5.1. Fundamental Reset

❏ Fundamental Reset applies only when Main power is present.

❏ Fundamental Reset does not apply when no power or Aux power is present.

When Fundamental Reset is asserted:

❏ The Receiver terminations are required to only meet $Z_{RX-HIGH-IMP-DC}$ (see Table 4-6).

❏ The Transmitter terminations are required to only meet Z_{TX-DC} (see Table 4-6).

❏ The Transmitter holds a constant DC common mode voltage.[23]

When Fundamental Reset is de-asserted:

❏ The Port LTSSM (see Section 4.2.5) is initialized (i.e., Detect is immediately entered).

---

[21] In this context, progress is defined as the LTSSM not remaining indefinitely in one state with the possible exception of Detect.

[22] The method to verify and re-establish Symbol lock is implementation specific.

[23] The common mode being driven does not need to meet the Absolute Delta Between DC Common Mode during L0 and Electrical Idle ($V_{TX-CM-DC-ACTIVE-IDLE-DELTA}$) specification (see Table 4-5).

4.2.4.5.2. Hot Reset

Hot Reset is a protocol reset defined in Section 4.2.5.11.

4.2.4.6. Link Data Rate Negotiation

All devices are required to start Link initialization using a generation 1 data rate on each Lane. A field in the training sequence ordered set (see Section 4.2.4) is used to advertise all supported data rates, and any higher speed supported by both sides of the Link will be initiated during Polling.Speed.

4.2.4.7. Link Width and Lane Sequence Negotiation

PCI Express Links must consist of 1, 2, 4, 8, 12, 16, or 32 Lanes in parallel, referred to as x1, x2, x4, x8, x12, x16, and x32 Links, respectively. All Lanes within a Link shall simultaneously (as defined by $L_{TX-SKEW}$ in Table 4-5) transmit data based on the same frequency. The negotiation process is described as a sequence of steps.

The negotiation establishes values for Link number and Lane number for each Lane that is part of a valid Link; each Lane that is not part of a valid Link exits the negotiation to become a separate Link or remain in Electrical Idle.

During Link width and Lane number negotiation, the two communicating Ports must accommodate the maximum allowed Lane-to-Lane skew as specified by $L_{RX-SKEW}$ in Table 4-6.

Optional Link negotiation behaviors include Lane reversal, variable width Links, splitting of Ports into multiple Links and the configuration of a crosslink.

Annex specifications to this specification may impose other rules and restrictions that must be comprehended by components compliant to those annex specifications; it is the intent of this specification to comprehend interoperability for a broad range of component capabilities.

4.2.4.7.1. Required and Optional Port Behavior

1. The ability for a xN Port to form a xN Link as well as a x1 Link (where N can be 32, 16, 12, 8, 4, 2, and 1) is required.

    *   Note: Designers must connect Ports between two different components in a way that allows those components to meet the above requirement. If the Ports between components are connected in ways that are not consistent with intended usage as defined by the component's Port descriptions/data sheets, behavior is undefined.

2. The ability for a xN Port to form any Link width between N and 1 is optional.

    *   Note: An example of this behavior includes a x16 Port which can only configure into only one Link, but the width of the Link can be configured to be x12, x8, x4, x2 as well as the required widths of x16 and x1.

3. The ability to split a Port into two or more Links is optional.

- Note: An example of this behavior would be a x16 Port that may be able to configure two x8 Links, four x4 Links, or even 16 x1 Links.

4. Support for Lane reversal is optional.

- Note: Lane reversal must be done for both the Transmitter and Receiver of a given Port for a multi-Lane Link.

- Note: An example of Lane reversal consists of Lane 0 of an Upstream Port attached to Lane N-1 of a Downstream Port where either the Downstream or Upstream device may reverse the Lane order to configure a xN Link.

Support for formation of a crosslink is optional. In this context, a Downstream Port connected to a Downstream Port or an Upstream Port connected to an Upstream Port is a crosslink.

Current and future electromechanical and/or form factor specifications may require the implementation of some optional features listed above. Component designers must read the specifications for the systems that the component(s) they are designing will used in to ensure compliance to those specifications.

4.2.4.8. Lane-to-Lane De-skew

The Receiver must compensate for the allowable skew between Lanes within a multi-Lane Link (see Table 4-5 and Table 4-6) before delivering the data and control to the Data Link Layer.

Lane-to-Lane de-skew shall be done across all Lanes within multi-Lane Links. An unambiguous de-skew mechanism is the COM Symbol transmitted during training sequence or SKP ordered sets across all Lanes within a configured Link. Other de-skew mechanisms may also be employed. Lane-to-Lane de-skew must be performed during Configuration, Recovery, and L0s in the LTSSM.

4.2.4.9. Lane vs. Link Training

The Link initialization process builds unassociated Lanes of a Port into associated Lanes that form a Link. For Lanes to configure properly into a desired Link, the TS1 and TS2 ordered sets must have the appropriate fields (Symbol 3, 4, and 5) set to the same value on all Lanes.

Links are formed at the conclusion of Configuration.

❑ Note: If the optional behavior of a Port being able to configure multiple Links is employed, the following observations can be made:

- A separate LTSSM is needed for the maximum number of Links that are desired to be configured by any given Port.

- The LTSSM Rules are written for configuring one Link. The decision to configure Links in a serial fashion or parallel is implementation specific.

4.2.5. Link Training and Status State Machine (LTSSM) Descriptions

The LTSSM states are illustrated in Figure 4-11. These states are described in following sections.

All timeout values specified in the Link Training and Status state machine (LTSSM) timeout values are minus 0 seconds and plus 50% unless explicitly stated otherwise. All timeout values must be set to the specified values after power-on/reset. All counter values must be set to the specified values after power-on/reset.

4.2.5.1. Detect

The purpose of this state is to detect when a far end termination is present. This state can be entered at any time if directed.

4.2.5.2. Polling

The Port transmits training ordered sets and responds to the received training ordered sets. In this state, bit lock and Symbol lock are established, Lane polarity is configured, and Lane data rate is established.

The polling state includes Polling.Compliance (see Section 4.2.6.2.2). This state is intended for use with test equipment used to assess if the Transmitter and the interconnect present in the device under test setup is compliant with the voltage and timing specifications in Table 4-5 and Table 4-6.

🗒 IMPLEMENTATION NOTE

Use of Polling.Compliance

Polling.Compliance is intended for a compliance test environment and not entered during normal operation and cannot be disabled for any reason. Polling.Compliance is entered based on the physical system environment as described in Section 4.2.6.2.1. Any other mechanism that causes a Transmitter to output the compliance pattern is implementation specific and is beyond the scope of this specification.

4.2.5.3. Configuration

In Configuration, both the Transmitter and Receiver are sending and receiving data at the negotiated data rate. The Lanes of a Port configure into a Link through a width and Lane negotiation sequence. Also, Lane-to-Lane de-skew must occur, scrambling can be disabled, the N_FTS is set, and the Disable or Loopback states can be entered.

4.2.5.4. Recovery

In Recovery, both the Transmitter and Receiver are sending and receiving data using the configured Link and Lane number as well as the previously negotiated data rate. Recovery allows a configured Link to re-establish bit lock, Symbol lock, and Lane-to-Lane de-skew. Recovery is also used to set a new N_FTS and enter the Loopback, Disable, Hot Reset, and Configuration states.

4.2.5.5. L0

L0 is the normal operational state where data and control packets can be transmitted and received. All power management states are entered from this state.

4.2.5.6. L0s

L0s is intended as a power savings state.

L0s allows a Link to quickly enter and recover from a power conservation state without going through Recovery.

The entry to L0s occurs after receiving an Electrical Idle ordered set.

The exit from L0s to L0 must re-establish bit lock, Symbol lock, and Lane-to-Lane de-skew.

A Transmitter and Receiver Lane pair on a Port are not required to both be in L0s simultaneously.

4.2.5.7. L1

L1 is intended as a power savings state.

The L1 state allows an additional power savings over L0s at the cost of additional resume latency.

The entry to L1 occurs after being directed by the Data Link Layer and receiving an Electrical Idle ordered set.

4.2.5.8. L2

Power can be aggressively conserved in L2. Most of the Transmitter and Receiver may be shut off.[24] Main power and clocks are not guaranteed, but aux[25] power is available.

When Beacon support is required by the associated system or form factor specification, an Upstream Port that supports the wakeup capability must be able to send; and a Downstream Port must be able to receive; a wakeup signal referred to as a Beacon.[26]

---

[24] The exception is the Receiver termination, which must remain in a low impedance state.

[25] In this context, "aux" power means a power source which can be used to drive the Beacon circuitry.

[26] Certain form factor specifications require the use of a Beacon for a device to request main power reactivation, for example to wake a system that is in $D3_{cold}$. See Section 4.3.2.4 for information on the electrical requirements of the Beacon. Refer to Chapter 5 for more information on how a device may use the Beacon as the wakeup mechanism.

The entry to L2 occurs after being directed by the Data Link Layer and receiving an Electrical Idle ordered set.

4.2.5.9. Disabled

The intent of the Disabled state is to allow a configured Link to be disabled until directed or Electrical Idle is exited (i.e., due to a hot removal and insertion) after entering Disabled.

Disabled uses bit 1 (Disable Link) in the Training Control field (see Table 4-2 and Table 4-3) which is sent within the TS1 and TS2 training ordered set.

A Link can enter Disabled if directed by a higher Layer. A Link can also reach the Disable state by receiving two consecutive TS1 ordered sets with the Disable bit asserted (see Section 4.2.6.9).

4.2.5.10. Loopback

Loopback is intended for test and fault isolation use. Only the entry and exit behavior is specified, all other details are implementation specific. Loopback can operate on either a per Lane or configured Link basis.

A Loopback Master is the component requesting Loopback.

A Loopback Slave is the component looping back the data.

Loopback uses bit 2 (Loopback) in the Training Control field (see Table 4-2 and Table 4-3) which is sent within the TS1 and TS2 training ordered set.

The entry mechanism for Loopback Master is device specific.

The Loopback Slave device enters Loopback whenever two consecutive TS1 ordered sets are received with the Loopback bit set.

IMPLEMENTATION NOTE

Use of Loopback

Once in the Loopback state, the master can send any pattern of Symbols as long as the rules of 8b/10b encoding (including disparity) are followed. Once in Loopback, the concept of data scrambling is no longer relevant; what is sent out is looped back. The mechanism(s) and/or interface(s) utilized by the Data Link Layer to notify the Physical Layer to enter the Loopback state is component implementation specific and beyond the scope of this specification.

4.2.5.11. Hot Reset

Hot Reset uses bit 0 (Hot Reset) in the Training Control field (see Table 4-2 and Table 4-3) which is sent within the TS1 and TS2 training ordered set.

A Link can enter Hot Reset if directed by a higher Layer. A Link can also reach the Hot Reset state by receiving two consecutive TS1 ordered sets with the Hot Reset bit asserted (see Section 4.2.6.10).

4.2.6.   Link Training and Status State Rules

Various Link status bits are monitored through software with the exception of LinkUp which is monitored by the Data Link Layer. Table 4-4 describes how the Link status bits must be handled throughout the LTSSM (for more information, see Section 3.1 for LinkUp; Section 7.8.8 for Link Speed, LinkWidth, and Link Training; Section 6.2 for Receiver Error; and Section 6.7 for In-Band Presence).

Table 4-4: Table of Link Status Mapped to the LTSSM

LTSSM State	Link Width	Link Speed	LinkUp	Link Training	Receiver Error	In-Band Presence [27]
Detect	Undefined	Undefined	0	0	No action	0
Polling	Undefined	Set	0	0	No action	1
Configuration	Set	Not action	0/1[28]	1	Set on 8b/10b Error	1
Recovery	No action	No action	1	1	No action	1
L0	No action	No action	1	0	Set on 8b/10b Error or optionally on Framing Violation	1
L0s	No action	No action	1	0	No action	1
L1	No action	No action	1	0	No action	1
L2	No action	No action	1	0	No action	1
Disabled	Undefined	Undefined	0	0	Set on 8b/10b Error	1
Loopback	No action	No action	0	0	No action	1
Hot Reset	No action	No action	0	0	Set on 8b/10b Error	1

The state machine rules for configuring and operating a PCI Express Link are defined in the following sections.

[27] In-band refers to the fact that no sideband signals are used to calculate the presence of a powered up device on the other end of a Link.

[28] LinkUp will always be 0 if coming into Configuration via Detect -> Polling -> Configuration and LinkUp will always be 1 if coming into Configuration from any other state.
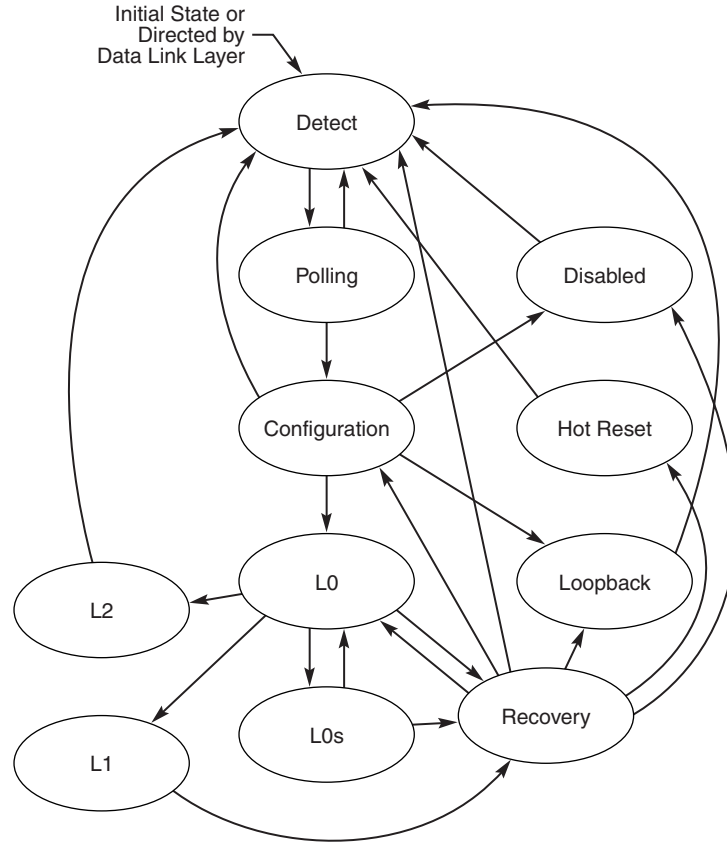
OM13800A

Figure 4-11: Main State Diagram for Link Training and Status State Machine

4.2.6.1. Detect

The Detect sub-state machine is shown in Figure 4-12.

4.2.6.1.1. Detect.Quiet

❑ Transmitter is in an Electrical Idle state.

5 • Note: The DC common mode voltage does not have to be within specification.[29]

❑ Generation 1 data rate is selected.

• Note: This does not affect the advertised data rate in TS1 and TS2.

❑ LinkUp = 0 (status is cleared).

❑ The next state is Detect.Active after a 12 ms timeout or if Electrical Idle is broken on any Lane.

---

[29] The common mode being driven does not need to meet the Absolute Delta Between DC Common Mode During L0 and Electrical Idle ($V_{TX-CM-DC-ACTIVE-IDLE-DELTA}$) specification (see Table 4-5).

4.2.6.1.2. Detect.Active

❑ The Transmitter performs a Receiver Detection sequence on all un-configured Lanes that can form one or more Links (see Section 4.3.1.8 for more information).

❑ Next state is Polling if a Receiver is detected on all unconfigured Lanes

❑ Next state is Detect.Quiet if a Receiver is not detected on any Lanes.

❑ If at least one but not all un-configured Lanes detect a Receiver, then:

1. Wait for 12 ms.

2. The Transmitter performs a Receiver Detection sequence on all un-configured Lanes that can form one or more Links (see Section 4.3.1.8 for more information),

   i) The next state is Polling if exactly the same Lanes detect a Receiver as the first Receiver Detection sequence.

      • Note: Lanes that did not detect a Receiver must:

        i) Be associated with a new LTSSM if this optional feature is supported.

        or

        ii) All Lanes that cannot be associated with an optional new LTSSM must transition to Electrical Idle.[30]

           • Note: These Lanes must be re-associated with the LTSSM immediately after the LTSSM in progress transitions back to Detect.

           • Note: An Electrical Idle ordered set does not need to be sent before transitioning to Electrical Idle.

   ii) Otherwise, the next state is Detect.Quiet.
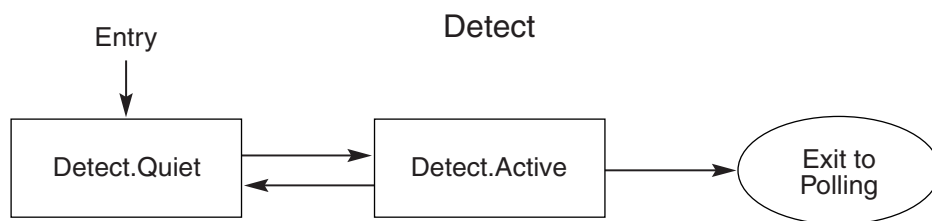
Figure 4-12: Detect Sub-State Machine

---

[30] The common mode being driven does not need to meet the Absolute Delta Between DC Common Mode During L0 and Electrical Idle ($V_{TX-CM-DC-ACTIVE-IDLE-DELTA}$) specification (see Table 4-5).

4.2.6.2. *Polling*

The Polling sub-state machine is shown in Figure 4-13.

4.2.6.2.1. Polling.Active

❑ Transmitter sends TS1 ordered sets with Lane and Link numbers set to PAD (K23.7) on all Lanes that detected a Receiver during Detect.

❑ Next state is Polling.Configuration after eight consecutive TS1 or TS2 ordered sets or their complement is received with the Lane and Link numbers set to PAD (K23.7) on all Lanes that detected a Receiver during Detect and at least 1024 TS1 ordered sets were transmitted.

❑ Otherwise, after a 24 ms timeout the next state is:

1. Polling.Configuration if,

    a) Any Lane, which detected a Receiver during Detect, received eight consecutive TS1 or TS2 ordered sets (or their complement) with the Lane and Link numbers set to PAD (K23.7), and a minimum of 1024 TS1s are transmitted after receiving one TS1.

        And

    b) All Lanes that detected a Receiver during Detect have detected an exit from Electrical Idle at least once since entering Polling.Active.

        • Note: This prevents one or more bad Receivers or Transmitters from holding up a valid Link from being configured, and allows for additional training in Polling.Configuration.

2. Polling.Compliance if at least one Lane's Receiver, which detected a Receiver during Detect, has never detected an exit from Electrical Idle since entering Polling.Active.

    • Note: This indicates the presence of a passive test load on at least one Lane, which will force all Lanes to enter Polling.Compliance.

3. Detect if no TS1 or TS2 ordered set is received with Link and Lane number set to Pad on any Lane. The highest advertised speed in TS1 and TS2 is lowered (unless generation 1 is the highest advertised speed).

4.2.6.2.2. Polling.Compliance

❑ Transmitter sends out the compliance pattern on all Lanes that detected a Receiver during Detect at the data rate which was employed upon entry to Polling.Compliance (see Section 4.2.8).

❑ Next state is Polling.Active if Electrical Idle exit has been detected at the Receiver of all Lanes that detected a Receiver during Detect.

4.2.6.2.3. Polling.Configuration

❑ Receiver must invert polarity if necessary (see Section 4.2.4.2).

❑ Transmitter sends TS2 ordered sets with Link and Lane numbers set to PAD (K23.7) on all Lanes that detected a Receiver during Detect .

❑ The next state is Configuration after eight consecutive TS2 ordered sets, with Link and Lane numbers set to PAD (K23.7), are received on any Lanes that detected a Receiver during Detect, 16 TS2 ordered sets are transmitted after receiving one TS2 ordered set, and none of those same Lanes is transmitting and receiving a higher Data Rate Identifier.[31]

❑ The next state is Polling.Speed after eight consecutive TS2 ordered sets, with Link and Lane numbers set to PAD (K23.7), are received on any Lanes that detected a Receiver during Detect, 16 TS2 ordered sets are transmitted after receiving one TS2 ordered set, and at least one of those same Lanes is transmitting and receiving a higher Data Rate Identifier.[32]

❑ Otherwise, next state is Detect after a 48 ms timeout.

4.2.6.2.4. Polling.Speed

❑ The Transmitter enters Electrical Idle for a minimum of $T_{TX-IDLE-MIN}$ (see Table 4-5) and no longer than 2 ms.

  • Note: Electrical Idle ordered set is sent prior to entering Electrical Idle.

  • Note: The DC common mode voltage does not have to be within specification.[33]

❑ Data rate is changed on all Lanes to the highest common data rate supported on both sides of the Link indicated by the training sequence (see Section 4.2.4.1).

❑ Next state is Polling.Active.

---

[31] Higher data rate than what is currently being executed. Data rate support on a Link is determined by the highest common speed being transmitted and received in TS1 and TS2 ordered sets.

[32] Higher data rate than what is currently being executed. Data rate support on a Link is determined by the highest common speed being transmitted and received in TS1 and TS2 ordered sets.

[33] The common mode being driven does not need to meet the Absolute Delta Between DC Common Mode During L0 and Electrical Idle ($V_{TX-CM-DC-ACTIVE-IDLE-DELTA}$) specification (see Table 4-5).

Polling

Entry

```
          ┌──────────────────┐          ┌──────────────────────┐
          │                  │─────────▶│                      │
          │  Polling.Active  │          │  Polling.Compliance  │
          │                  │◀─────────│                      │
          └──────────────────┘          └──────────────────────┘

          ┌──────────────────────┐      ┌──────────────────┐
          │                      │─────▶│                  │
          │ Polling.Configuration│      │  Polling.Speed   │
          │                      │      │                  │
          └──────────────────────┘      └──────────────────┘

        ╭─────────╮          ╭───────────────╮
        │ Exit to │          │    Exit to    │
        │ Detect  │          │ Configuration │
        ╰─────────╯          ╰───────────────╯
```
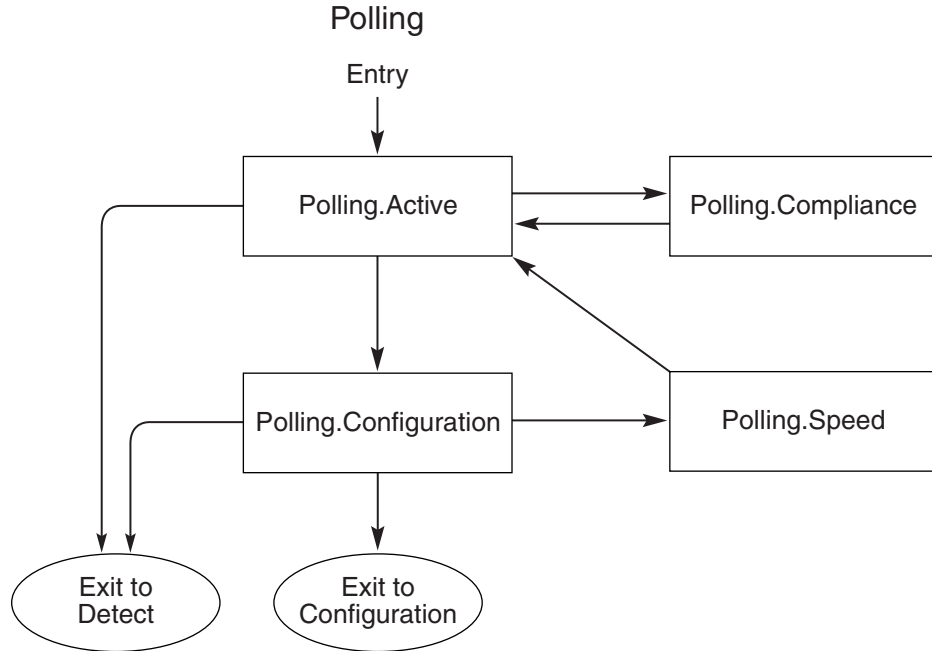
OM13801A

Figure 4-13: Polling Sub-State Machine

4.2.6.3.   Configuration

The Configuration sub-state machine is shown in Figure 4-14.

4.2.6.3.1. Configuration.Linkwidth.Start

4.2.6.3.1.1. Downstream Lanes

5 ❑ Next state is Disable if directed.

- Note: "if directed" applies to a Downstream Port that is instructed by a higher Layer to assert the Disable Link bit (TS1 and TS2) on all Lanes that detected a Receiver during Detect.

❑ Next state is Loopback if directed to this state, and the Transmitter is capable of being a
10   Loopback Master, which is determined by implementation specific means.

- Note: "if directed" applies to a Port that is instructed by a higher Layer to assert the Loopback bit (TS1 and TS2) on all Lanes that detected a Receiver during Detect.

❑ In the optional case where a crosslink is supported, the next state is Disable after all Lanes that detected a Receiver during Detect, that are also receiving TS1 ordered sets, receive the Disable
15   bit asserted in two consecutive TS1 ordered sets.

❑ Next state is Loopback after all Lanes that detected a Receiver during Detect, that are also receiving ordered sets, receive the Loopback bit asserted in two consecutive TS1 ordered sets.

- Note that the device receiving the ordered set with the Loopback bit set becomes the Loopback Slave.

5 ❑ The Transmitter sends TS1 ordered sets with selected Link numbers and sets Lane numbers to PAD (K23.7) on Downstream Lanes that are in Configuration.

- Note: Link numbers can only be different for groups of Lanes capable of being a unique Link.

- Note: An example of Link number assignments includes a set of eight Lanes on an
10 Upstream component (Downstream Lanes) capable of negotiating to become one x8 Port when connected to one Downstream component (Upstream Lanes) or two x4 Ports when connected to two different Downstream components. The Upstream component (Downstream Lanes) sends out TS1 ordered sets with the Link number set to N on four Lanes and Link number set to N+1 on the other four Lanes. The Lane numbers are all set
15 to PAD (K23.7).

❑ If any Lanes first received at least one or more TS1 ordered sets with a Link and Lane number set to PAD (K23.7), the next state is Configuration.Linkwidth.Accept immediately after any of those same Downstream Lanes receive two consecutive TS1 ordered sets with a non-PAD Link number that matches any of the transmitted Link numbers, and with a Lane number set to PAD
20 (K23.7).

- Note: If the crosslink configuration is not supported, the condition of first receiving a Link and Lane number set to PAD is always true.

❑ Optionally, if crosslinks are supported and all Downstream Lanes initially receive two consecutive TS1 ordered sets with a Link number different than PAD (K23.7) and a Lane
25 Number set to PAD, the Downstream Lanes are now designated as Upstream Lanes and a new random cross Link timeout is chosen (see $T_{crosslink}$ in Table 4-5). The next state is Configuration.Linkwidth.Start as Upstream Lanes.

- Note: This supports the optional crosslink where both sides may try to act as a Downstream Port. This is resolved by making both Ports become Upstream and assigning a random
30 timeout until one side of the Link becomes a Downstream Port and the other side remains an Upstream Port. This timeout must be random even when hooking up two of the same devices so as to eventually break any possible deadlock.

❑ The next state is Detect after a 24 ms timeout.

4.2.6.3.1.2.   Upstream Lanes

35 ❑ In the optional case where crosslinks are supported the next state is Disable if directed.

- Note: "if directed" only applies to an optional crosslink Port that is instructed by a higher Layer to assert the Disable Link bit (TS1 and TS2) on all Lanes that detected a Receiver during Detect.

❑ Next state is Loopback if directed to this state, and the Transmitter is capable of being a Loopback Master, which is determined by implementation specific means.

- Note: "if directed" applies to a Port that is instructed by a higher Layer to assert the Loopback bit (TS1 and TS2) on all Lanes that detected a Receiver during Detect.

❑ Next state is Disable after any Lanes that detected a Receiver during Detect and are receiving TS1 ordered sets with the Disable Link bit asserted in two consecutive TS1 ordered sets.

- Note: In the optional case where a crosslink is supported, the next state is Disable only after all Lanes that detected a Receiver during Detect, that are also receiving TS1 ordered sets, receive the Disable bit asserted in two consecutive TS1 ordered sets.

❑ Next state is Loopback after all Lanes that detected a Receiver during Detect, that are also receiving TS1 ordered sets, receive the Loopback bit asserted in two consecutive TS1 ordered sets.

- Note: The device receiving the ordered set with the Loopback bit set becomes the Loopback Slave.

❑ The Transmitter sends out TS1 ordered sets with Link numbers and Lane numbers set to PAD (K23.7) on Upstream Lanes that detected a Receiver during Detect.

❑ If any Lanes receive two consecutive TS1 ordered sets with Link numbers that are different than PAD (K23.7) and Lane number set to PAD(K23.7), a single Link number is selected and transmitted on all Lanes that both detected a Receiver and also received two consecutive TS1 ordered sets with Link numbers that are different than PAD (K23.7) and Lane number set to PAD (K23.7). Any left over Lanes that detected a Receiver during Detect must transmit TS1 ordered sets with the Link and Lane number set to PAD (K23.7). The next state is Configuration.Linkwidth.Accept.

- Note: It is recommended that any possible multi-Lane Link that received an error in a TS1 ordered set on a subset of the received Lanes; delay the evaluation listed above by an additional two TS1 ordered sets so as not to prematurely configure a smaller Link than possible.

❑ Optionally, if crosslinks are supported and any Upstream Lanes first receive two consecutive TS1 ordered sets with Link and Lane numbers set to PAD (K23.7), then:

- The Transmitter continues to send out TS1 ordered sets with Link numbers and Lane numbers set to PAD (K23.7).

- If any Lanes receive two consecutive TS1 ordered sets with Link numbers that are different than PAD (K23.7) and Lane number set to PAD (K23.7), a single Link number is selected and transmitted on all Lanes that both detected a Receiver and also received two consecutive TS1 ordered sets with Link numbers that are different than PAD (K23.7) and Lane number set to PAD (K23.7). Any left over Lanes that detected a Receiver during Detect must transmit TS1 ordered sets with the Link and Lane number set to PAD (K23.7). The next state is Configuration.Linkwidth.Accept.

♦ Note: It is recommended that any possible multi-Lane Link that received an error in a TS1 ordered set on a subset of the received Lanes; delay the evaluation listed above by an additional two TS1 ordered sets so as not to prematurely configure a smaller Link than possible.

- Otherwise, after a $T_{crosslink}$ timeout the Upstream Lanes become Downstream Lanes and the next state is Configuration.Linkwidth.Start as Downstream Lanes.

  ♦ Note: This optional behavior is required for crosslink behavior where two Ports may start off with Upstream Ports, and one will eventually take the lead as a Downstream Port.

❑ The next state is Detect after a 24 ms timeout.

4.2.6.3.2. Configuration.Linkwidth.Accept

4.2.6.3.2.1. Downstream Lanes

❑ If a configured Link can be formed with at least one group of Lanes that received two consecutive TS1 ordered sets with the same received Link number (non-PAD and matching one that was transmitted by the Downstream Lanes), TS1 ordered sets are transmitted with the same Link number and unique non-PAD Lane numbers are assigned to all theses same Lanes. The next state is Configuration.Lanenum.Wait.

- Note: The assigned non-PAD Lane numbers must range from 0 to n-1, be assigned sequentially to the same grouping of Lanes that are receiving the same Link number Lane numbers, and Downstream Lanes which are not receiving TS1 ordered sets must not disrupt the initial sequential numbering of the widest possible Link. Any left over Lanes must transmit TS1 ordered sets with the Link and Lane number set to PAD (K23.7).

- Note: It is recommended that any possible multi-Lane Link that received an error in a TS1 ordered set on a subset of the received Lanes delay the evaluation listed above by an additional two TS1 ordered sets so as not to prematurely configure a smaller Link than possible.

- Note: A couple of interesting cases to consider here are the following:

  1. A x8 Downstream Port, which can be divided into two x4 Links, sends two different Link numbers on to two x4 Upstream Ports. The Upstream Ports respond simultaneously by picking the two Link numbers. The Downstream Port will have to choose one of these sets of Link numbers to configure as a Link, and leave the other for a secondary LTSSM to configure (which will ultimately happen in Configuration.Complete).

  2. A x16 Downstream Port, which can be divided into two x8 Links, is hooked up to a x12 Upstream Port that can be configured as a x12 Link or a x8 and a x4 Link. During Configuration.Linkwidth.Start the Upstream Port returned the same Link number on all 12 Lanes. The Downstream Port would then return the same received Link number and assign Lane numbers on the eight Lanes that can form a x8 Link with the remaining four Lanes transmitting a Lane number and a Link number set to PAD (K23.7).

178

3. A x8 Downstream Port where only seven Lanes are receiving TS1s with the same received Link number (non-PAD and matching one that was transmitted by the Downstream Lanes) and an eighth Lane, which is in the middle or adjacent to those same Lanes, is not receiving a TS1 ordered-set. In this case, the eighth Lane is treated the same as the other seven Lanes and Lane numbering for a x8 Lane should occur as described above.

❑ The next state is Detect after a 2 ms timeout or if no Link can be configured or if all Lanes receive two consecutive TS1 ordered sets with Link and Lane numbers set to Pad (K23.7).

4.2.6.3.2.2. Upstream Lanes

❑ If a configured Link can be formed using Lanes that transmitted a non-PAD Link number which are receiving two consecutive TS1 ordered sets with the same non-PAD Link number and any non-PAD Lane number, TS1 Lane numbers are transmitted that, if possible, match the received Lane numbers or are different, if necessary, (i.e., Lane reversed). The next state is Configuration.Lanenum.Wait.

- Note: The newly assigned Lane numbers must range from 0 to m-1, be assigned sequentially only to some continuous grouping of Lanes that are receiving non-PAD Lane numbers (i.e., Lanes which are not receiving any TS1 ordered sets always distrupt a continuous grouping and must not be included in this grouping), must include either Lane 0 or n-1 (largest received Lane number), and m-1 must be equal to or smaller than the largest received Lane number (n-1). Remaining Lanes must transmit TS1 with Link and Lane numbers set to PAD (K23.7).

- Note: It is recommended that any possible multi-Lane Link that received an error in a TS1 ordered sets on a subset of the received Lanes delay the evaluation listed above by an additional two TS1 ordered sets so as not to prematurely configure a smaller Link than possible.

- Note: A few interesting cases to consider here are the following:

  1. A x8 Upstream Port is presented with Lane numbers that are backward from the preferred numbering. If the optional behavior of Lane reversal is supported by the Upstream Port, the Upstream Port transmits the same Lane numbers back to the Downstream Port. Otherwise the opposite Lane numbers are transmitted back to the Downstream Port, and it will be up to the Downstream Port to optionally fix the Lane ordering or exit Configuration.

     ■ Note: Optional Lane reversal behavior is required to configure a Link where the Lane numbers are reversed and the Downstream Port does not support Lane reversal. Specifically, the Upstream Port Lane reversal will accommodate the scenario where the default Upstream sequential Lane numbering (0 to n-1) is receiving a reversed Downstream sequential Lane number (n-1 to 0).

2. A x8 Upstream Port is not receiving TS1 ordered-sets on the Upstream Port Lane 0:

    a. In the case where the Upstream Port can only support a x8 or x1 Link and the Upstream Port can support Lane reversal. The Upstream Port will assign a Lane 0 to only the received Lane 7 (received Lane number n-1) and the remaining seven Lanes must transmit TS1 with Link and Lane numbers set to PAD (K23.7)

    b. In the case where the Upstream Port can only support a x8 or x1 Link and the Upstream Port cannot support Lane reversal. No Link can be formed and the Upstream Port will eventually timeout after 2 ms and exit to Detect.

3. An optional x8 Upstream crosslink Port, which can be divided into two x4 Links, is attached to two x4 Downstream Ports that present the same Link number, and each x4 Downstream Port presents Lane numbers simultaneously that were each numbered 0 to 3. The Upstream Port will have to choose one of these sets of Lane numbers to configure as a Link, and leave the other for a second pass through Configuration.

❑ The next state is Detect after a 2 ms timeout or if no Link can be configured or if all Lanes receive two consecutive TS1 ordered sets with Link and Lane numbers set to Pad (K23.7).

4.2.6.3.3.   Configuration.Lanenum.Accept

4.2.6.3.3.1   Downstream Lanes

❑ If two consecutive TS1 ordered sets are received with non-PAD Link and non-PAD Lane numbers that match all the non-PAD Link and non-PAD Lane numbers (or reversed Lane numbers if Lane reversal is optionally supported) that are being transmitted in Downstream Lane TS1 ordered sets, the next state is Configuration.Complete.

- Note: Reversed Lane numbers is defined strictly as the downstream Lane 0 receiving a TS1 ordered set with a Lane number equal to n-1 and the downstream Lane n-1 receiving a TS1 ordered set with a Lane number equal to 0.

- Note: It is recommended that any possible multi-Lane Link that received an error in a TS1 ordered sets on a subset of the received Lanes delay the evaluation listed above by an additional two TS1 ordered sets so as not to prematurely configure a smaller Link than possible.

❑ If a configured Link can be formed with any subset of the Lanes that receive two consecutive TS1 ordered sets with the same transmitted non-PAD Link numbers and any non-Pad Lane numbers, TS1 ordered sets are transmitted with new Lane numbers assigned and the next state is Configuration.Lanenum.Wait.

- Note: The newly assigned transmitted Lane numbers must range from 0 to m-1, be assigned sequentially only to some continuous grouping of the Lanes that are receiving non-PAD Lane numbers (i.e., Lanes which are not receiving any TS1 ordered sets always disrupt a continuous grouping and must not be included in this grouping), must include either Lane 0 or n-1 (largest received Lane number), and m-1 must be smaller than the largest received Lane number (n-1). Any left over Lanes must transmit TS1 ordered sets with the Link and Lane number set to PAD (K23.7).

- Note: It is recommended that any possible multi-Lane Link that received an error in a TS1 ordered set on a subset of the received Lanes delay the evaluation listed above by an additional two TS1 ordered sets so as not to prematurely configure a smaller Link than possible.

5 ❑ The next state is Detect if no Link can be configured or if all Lanes receive two consecutive TS1 ordered sets with Link and Lane numbers set to Pad (K23.7).

4.2.6.3.3.2 Upstream Lanes

❑ If two consecutive TS2 ordered sets are received with non-PAD Link and non-PAD Lane numbers that match all non-PAD Link and non-PAD Lane numbers that are being transmitted
10 in Upstream Lane TS1 ordered sets, the next state is Configuration.Complete.

❑ If a configured Link can be formed with any subset of the Lanes that receive two consecutive TS1 ordered sets with the same transmitted non-PAD Link numbers and any non-PAD Lane numbers, TS1 ordered sets are transmitted with new Lane numbers assigned and the next state is Configuration.Lanenum.Wait.

15 - Note: The newly assigned transmitted Lane numbers must range from 0 to m-1, be assigned sequentially only to some continuous grouping of Lanes that are receiving non-PAD Lane numbers (i.e., Lanes which are not receiving any TS1 ordered sets always disrupt a continuous grouping and must not be included in this grouping), must include either Lane 0 or n-1 (largest received Lane number), and m-1 must be smaller than the largest received
20 Lane number (n-1). Any left over Lanes must transmit TS1 ordered sets with the Link and Lane number set to PAD (K23.7).

- Note: It is recommended that any possible multi-Lane Link that received an error in a TS1 ordered sets on a subset of the received Lanes delay the evaluation listed above by an additional two TS1 ordered sets so as not to pre-maturely configure a smaller Link than
25 possible.

❑ The next state is Detect if no Link can be configured or if all Lanes receive two consecutive TS1 ordered sets with Link and Lane numbers set to Pad (K23.7).

4.2.6.3.4. Configuration.Lanenum.Wait

4.2.6.3.4.1. Downstream Lanes

30 ❑ The next state is Configuration.Lanenum.Accept if any of the Lanes receive two consecutive TS1 which have a Lane number different from when the Lane first entered Configuration.Lanenum.Wait, and not all the Lanes Link numbers are set to Pad (K23.7).

❑ The next state is Detect after a 2 ms timeout or if all Lanes receive two consecutive TS1 ordered sets with Link and Lane numbers set to Pad (K23.7).

4.2.6.3.4.2.   Upstream Lanes

❑ The next state is Configuration.Lanenum.Accept

1. If any of the Lanes receive two consecutive TS1s that have a Lane number different from when the Lane first entered Configuration.Lanenum.Wait, and not all the Lanes' Link numbers are set to Pad (K23.7)

or

2. If any Lane receives two consecutive TS2 ordered sets

❑ The next state is Detect after a 2 ms timeout or if all Lanes receive two consecutive TS1 ordered sets with Link and Lane numbers set to Pad (K23.7).

4.2.6.3.5.   Configuration.Complete

4.2.6.3.5.1.   Downstream Lanes

❑ TS2 ordered sets are transmitted using Link and Lane numbers that match the received TS1 Link and Lane numbers.

❑ N_FTS must be noted for use in L0s when leaving this state.

❑ Lane-to-Lane de-skew must be completed when leaving this state.

❑ Scrambling is disabled if all configured Lanes have the Disable Scrambling bit asserted in two consecutively received TS2 ordered sets.

- Note: It is required that the Port that is sending the Disable Scrambling bit on all of the configured Lanes will also disable scrambling.

❑ The next state is Configuration.Idle immediately after all Lanes that are transmitting TS2 ordered sets receive eight consecutive TS2 ordered sets with matching Lane and Link numbers (non-Pad) and 16 TS2 ordered sets are sent after receiving one TS2 ordered set.

- Note: All remaining Lanes that are not part of the configured Link are no longer associated with the LTSSM in progress and must:

  i. Be associated with a new LTSSM if this optional feature is supported.

  or

ii. All Lanes that cannot be associated with an optional new LTSSM must transition to Electrical Idle.[34]

- ■ Note: In the case of an optional crosslink, the Receiver terminations are required to meet $Z_{RX\text{-}HIGH\text{-}IMP\text{-}DC}$ (see Table Table 4-6).

- ■ Note: These Lanes must be re-associated with the LTSSM immediately after the LTSSM in progress transitions back to Detect.

- ■ Note: An Electrical Idle ordered set does not need to be sent before transitioning to Electrical Idle.

❑ The next state is Detect after a 2 ms timeout.

4.2.6.3.5.2.   Upstream Lanes

❑ TS2 ordered sets are transmitted using Link and Lane numbers that match the received TS2 Link and Lane numbers.

❑ N_FTS must be noted for use in L0s when leaving this state.

❑ Lane-to-Lane de-skew must be completed when leaving this state.

❑ Scrambling is disabled if all configured Lanes have the Disable Scrambling bit asserted in two consecutively received TS2 ordered sets.

- • Note: It is required that the Port that is sending the Disable Scrambling bit on all of the configured Lanes will also disable scrambling.

❑ The next state is Configuration.Idle immediately after all Lanes that are transmitting TS2 ordered sets receive eight consecutive TS2 ordered sets with matching Lane and Link numbers (non-Pad) and 16 consecutive TS2 ordered sets are sent after receiving one TS2 ordered set.

- • Note: All remaining Lanes that are not part of the configured Link are no longer associated with the LTSSM in progress and must:

  i. Optionally be associated with a new crosslink LTSSM if this feature is supported.

  or

  ii. All remaining Lanes that are not associated with a new crosslink LTSSM must transition to Electrical Idle[35], and Receiver terminations are required to meet $Z_{RX\text{-}HIGH\text{-}IMP\text{-}DC}$ (see Table 4-6).

  - ■ Note: These Lanes must be re-associated with the LTSSM immediately after the LTSSM in progress transitions back to Detect.

  - ■ Note: An Electrical Idle ordered set does not need to be sent before transitioning to Electrical Idle.

---

[34] The common mode being driven does not need to meet the Absolute Delta Between DC Common Mode During L0 and Electrical Idle ($V_{TX\text{-}CM\text{-}DC\text{-}ACTIVE\text{-}IDLE\text{-}DELTA}$) specification (see Table 4-5).

[35] The common mode being driven does not need to meet the Absolute Delta Between DC Common Mode During L0 and Electrical Idle ($V_{TX\text{-}CM\text{-}DC\text{-}ACTIVE\text{-}IDLE\text{-}DELTA}$) specification (see Table 4-5).

❑ The next state is Detect after a 2 ms timeout.

4.2.6.3.6. Configuration.Idle

❑ Transmitter sends Idle data Symbols on all configured Lanes.

❑ Receiver waits for Idle data.

5 ❑ LinkUp = 1 (status is set true).

❑ Next state is L0 if eight consecutive Symbol Times of Idle data received on all configured Lanes and 16 Idle data Symbols are sent after receiving one Idle data Symbol.

❑ Otherwise, the next state is Detect after a minimum 2 ms timeout.

Configuration

Entry

Exit to
Disable

Configuration.Linkwidth.Start

Exit to
Loopback

Configuration.Linkwidth.Accept

Exit to
Detect

Configuration.Lanenum.Wait

Configuration.Lanenum.Accept

Configuration.Complete

Configuration.Idle

Exit to
L0

OM13802A

Figure 4-14: Configuration Sub-State Machine

4.2.6.4. Recovery

The Recovery sub-state machine is shown in Figure 4-15.

4.2.6.4.1. Recovery.RcvrLock

❑ Transmitter sends TS1 ordered sets on all configured Lanes using the same Link and Lane
numbers that were set after leaving Configuration.

❑ Next state is Recovery.RcvrCfg if eight consecutive TS1 or TS2 ordered sets are received on all
configured Lanes with the same Link and Lane numbers that match what is being transmitted on
those same Lanes.

- • Note: If the Extended Synch bit is set, the Transmitter must send a minimum of 1024
consecutive TS1 ordered sets before transitioning to Recovery.RcvrCfg.

    - ♦ Note: Extended Synch allows external Link monitoring tools (e.g., logic analyzers)
enough time to achieve bit and Symbol lock.

❑ Otherwise, after a 24 ms timeout:

1. The next state is Configuration if any of the configured Lanes that are receiving a TS1 or
TS2 ordered set have received at least one TS1 or TS2 with Link and Lane numbers that
match what is being transmitted on those same Lanes.

2. Otherwise, the next state is Detect.

4.2.6.4.2. Recovery.RcvrCfg

❑ Transmitter sends TS2 ordered sets on all configured Lanes using the same Link and Lane
numbers that were set after leaving Configuration.

❑ Next state is Recovery.Idle if eight consecutive TS2 ordered sets are received on all configured
Lanes with the same Link and Lane number that match what is being transmitted on those same
Lanes and 16 TS2 ordered sets are sent after receiving one TS2 ordered set.

- • Note: If the N_FTS value was changed, the new value must be used for future L0s states.

- • Note: Lane-to-Lane de-skew must be completed before leaving Recovery.RcvrCfg.

❑ Next state is Configuration if eight consecutive TS1 ordered sets are received on any configured
Lanes with Link or Lane numbers that do not match what is being transmitted on those same
Lanes and 16 TS2 ordered sets are sent after receiving one TS1 ordered set.

- • Note: If the N_FTS value was changed, the new value must be used for future L0s states.

❑ Otherwise, after a 48 ms timeout the next state is Detect.

4.2.6.4.3. Recovery.Idle

❑ Next state is Disabled if directed.

- • Note: "if directed" applies to a Downstream or optional crosslink Port that is instructed by a higher Layer to assert the Disable Link bit (TS1 and TS2) on the Link.

5 ❑ Next state is Hot Reset if directed.

- • Note: "if directed" applies to a Downstream or optional crosslink Port that is instructed by a higher Layer to assert the Hot Reset bit (TS1 and TS2) on the Link.

❑ Next state is Configuration if directed.

- • Note: "if directed" applies to a Port that is instructed by a higher Layer to optionally re-
10 configure the Link (i.e., different width Link).

❑ Next state is Loopback if directed to this state, and the Transmitter is capable of being a Loopback Master, which is determined by implementation specific means.

- • Note: "if directed" applies to a Port that is instructed by a higher Layer to assert the Loopback bit (TS1 and TS2) on the Link.

15 ❑ Next state is Disabled immediately after any configured Lane has the Disable Link bit asserted in two consecutively received TS1 ordered sets.

- • Note: This is behavior only applicable to Upstream and optional crosslink Ports.

❑ Next state is Hot Reset immediately after any configured Lane has the Hot Reset bit asserted in two consecutive TS1 ordered sets.

20 • Note: This is behavior only applicable to Upstream and optional crosslink Ports.

❑ Next state is Configuration if two consecutive TS1 ordered sets are received on any configured Lane with a Lane number set to PAD.

- • Note: A Port that optionally transitions to Configuration to change the Link configuration is guaranteed to send Lane numbers set to PAD on all Lanes.

25 ❑ Next state is Loopback if any configured Lane has the Loopback bit asserted in two consecutive TS1 ordered sets.

- • Note: The device receiving the ordered set with the Loopback bit set becomes the Loopback Slave.

❑ Transmitter sends Idle data on all configured Lanes.

30 • Note: If directed to other states, Idle Characters do not have to be sent before transitioning to the other states (i.e., Disable, Hot Reset, Configuration, or Loopback)

❑ Next state is L0 if eight consecutive Symbol Times of Idle data received on all configured Lanes and 16 Idle data Symbols are sent after receiving one Idle data Symbol.

❑ Otherwise, after a 2 ms timeout the next state is Detect.
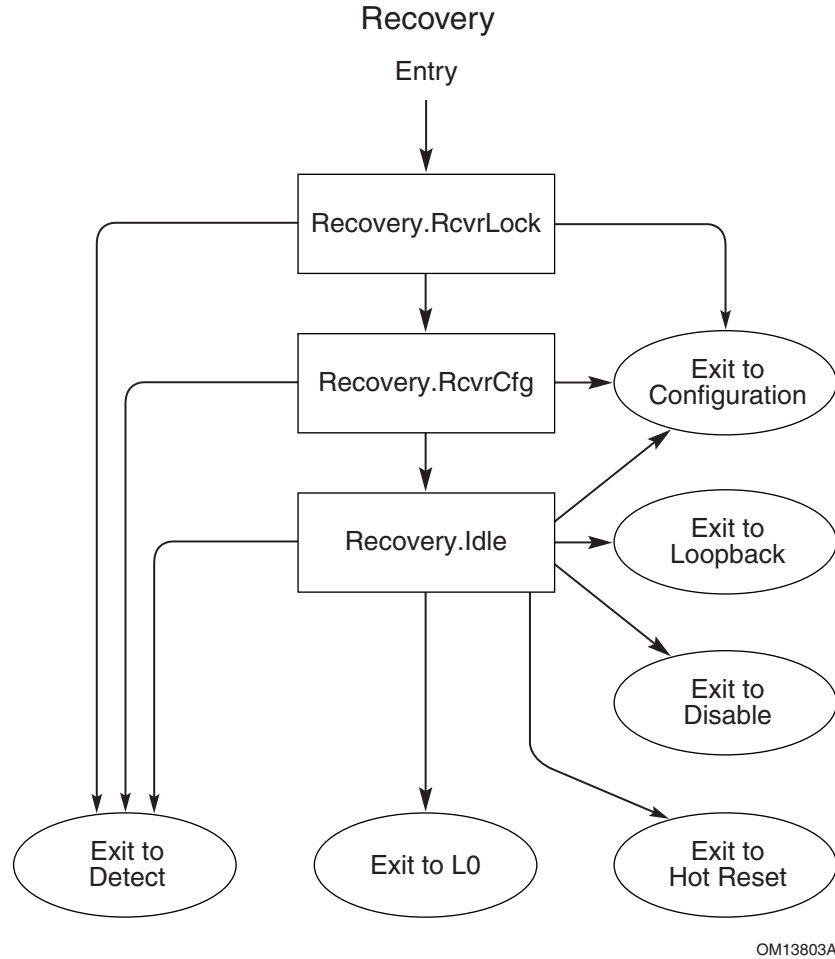
Figure 4-15: Recovery Sub-State Machine

4.2.6.5. L0

This is the normal operational state.

❑ LinkUp = 1 (status is set true).

❑ Next state is Recovery if a TS1 or TS2 ordered set is received on any configured Lane.

5 ❑ Next state is Recovery if directed to this state or if Electrical Idle is detected on all Lanes without receiving an Electrical Idle ordered set on any Lane.

  • Note: "if directed" applies to a Port that is instructed by a higher Layer to transition to Recovery.

  • Note: The Transmitter may complete any TLP or DLLP in progress.

❑ Next state is L0s for only the Transmitter if directed to this state.

- • Note: "if directed" applies to a Port that is instructed by a higher Layer to initiate L0s (see Section 5.4.1.1.1).

- • Note: This is a point where the TX and RX may diverge into different LTSSM states.

❑ Next state is L0s for only the Receiver if an Electrical Idle ordered set is received on any Lanes and the Port is not directed to L1 or L2 states by any higher layers.

- • Note: This is a point where the TX and RX may diverge into different LTSSM states.

❑ Next state is L1:

i. If directed

and

ii. an Electrical Idle ordered set is received on any Lane

and

iii. an Electrical Idle ordered set is transmitted on all Lanes.

- • Note: "if directed" is defined as both ends of the Link having agreed to enter L1 immediately after the condition of both the receipt and transmission of an Electrical Idle ordered set is met (see Section 4.3.2.1).

- • Note: When directed by a higher Layer one side of the Link always initiates and exits to L1 by transmitting an Electrical Idle ordered set on all Lanes, followed by a transition to Electrical Idle.[36] The same Port then waits for the receipt of an Electrical Idle ordered set on any Lane, and then immediately transitions to L1. Conversely, the side of the Link that first receives an Electrical Idle ordered set on any Lane must send an Electrical Idle ordered set on all Lanes and immediately transition to L1.

❑ Next state is L2:

i. If directed

and

ii. an Electrical Idle ordered set is received on any Lane

and

iii. an Electrical Idle ordered set is transmitted on all Lanes.

- • Note: "if directed" is defined as both ends of the Link having agreed to enter L2 immediately after the condition of both the receipt and transmission of an Electrical Idle ordered set is met (see Section 4.3.2.3 for more details).

---

[36] The common mode being driven must meet the Absolute Delta Between DC Common Mode During L0 and Electrical Idle ($V_{TX-CM-DC-ACTIVE-IDLE-DELTA}$) specification (see Table 4-5).

- Note: When directed by a higher Layer, one side of the Link always initiates and exits to L2 by transmitting an Electrical Idle ordered set on all Lanes followed by a transition to Electrical Idle.[37] The same Port then waits for the receipt of an Electrical Idle ordered set on any Lane, and then immediately transitions to L2. Conversely, the side of the Link that first receives an Electrical Idle ordered set on any Lane must send an Electrical Idle ordered set on all Lanes and immediately transition to L2.

4.2.6.6.   L0s

The L0s sub-state machine is shown in Figure 4-16.

4.2.6.6.1.   Receiver L0s

4.2.6.6.1.1. Rx_L0s.Entry

❑ Next state is Rx_L0s.Idle after a T_{TX-MIN} (Table 4-5) timeout

- Note: This is the minimum time the Transmitter must be in an Electrical Idle condition.

4.2.6.6.1.2. Rx_L0s.Idle

❑ Next state is Rx_L0s.FTS if the Receiver detects an exit from Electrical Idle on any Lane of the configured Link.

4.2.6.6.1.3. Rx_L0s.FTS

❑ The next state is L0 if a SKP ordered set is received on all configured Lanes of the Link.

- Note: The Receiver must be able to accept valid data immediately after the SKP ordered set.

- Note: Lane-to-Lane de-skew must be completed before leaving Rx_L0s.FTS.

❑ Otherwise, next state is Recovery after the N_FTS timeout.

- The N_FTS timeout shall be no shorter than 40*[N_FTS+3] * UI (The 3 * 40 UI is derived from six Symbols to cover a maximum SKP ordered set + four Symbols for a possible extra FTS ordered set + 2 Symbols of design margin), and no longer than twice this amount. When the extended synch bit is set the Receiver N_FTS timeout must be adjusted to no shorter than 40* [2048] * UI (2048 FTS ordered sets) and no longer than 40* [4096] * UI (4096 FTS ordered sets).

- Note: The Transmitter must also transition to Recovery, but may complete any TLP or DLLP in progress.

---

[37] The common mode being driven does not need to meet the Absolute Delta Between DC Common Mode During L0 and Electrical Idle ($V_{TX-CM-DC-ACTIVE-IDLE-DELTA}$) specification (see Table 4-5).

- Note: It is recommended that the N_FTS field be increased when transitioning to Recovery to prevent future transitions to Recovery from Rx_L0s.FTS.

4.2.6.6.2. Transmitter L0s

4.2.6.6.2.1. Tx_L0s.Entry

5 ❑ Transmitter sends the Electrical Idle ordered set and enters Electrical Idle.

- Note: The DC common mode voltage must be within specification by $T_{TX\text{-}IDLE\text{-}SET\text{-}TO\text{-}IDLE}$.[38]

❑ Next state is Tx_L0s.Idle after a $T_{TX\text{-}IDLE\text{-}MIN}$ (Table 4-5) timeout.

4.2.6.6.2.2. Tx_L0s.Idle

10 ❑ Next state is Tx_L0s.FTS if directed.

4.2.6.6.2.3. Tx_L0s.FTS

❑ Transmitter sends N_FTS Fast Training Sequences on all configured Lanes.

- Note: Up to one full FTS ordered set may be sent before the N_FTS FTS ordered sets are sent.

15 - Note: No SKP ordered sets can be inserted before all FTS ordered sets as defined by the agreed upon N_FTS parameter are transmitted.

- Note: If the Extended Synch bit is set, the Transmitter sends 4096 Fast Training Sequences.

❑ Transmitter sends a single SKP ordered set on all configured Lanes.

❑ Next state is L0.

---

[38] The common mode being driven must meet the Absolute Delta Between DC Common Mode During L0 and Electrical Idle ($V_{TX\text{-}CM\text{-}DC\text{-}ACTIVE\text{-}IDLE\text{-}DELTA}$) specification (see Table 4-5).

L0s: Receiver

```
Entry
  │
  ▼
┌──────────────┐      ┌──────────────┐
│ Rx_L0s.Entry │─────▶│ Rx_L0s.Idle  │
└──────────────┘      └──────────────┘
                             │
                             ▼
                      ┌──────────────┐
                      │  Rx_L0s.FTS  │
                      └──────────────┘
                         ╱        ╲
                        ▼          ▼
                   ⟨ Exit to ⟩  ⟨ Recovery ⟩
                   ⟨   L0    ⟩
```

L0s: Transmitter

```
Entry
  │
  ▼
┌──────────────┐   ┌──────────────┐   ┌──────────────┐
│ Tx_L0s.Entry │──▶│ Tx_L0s.Idle  │──▶│  Tx_L0s.FTS  │
└──────────────┘   └──────────────┘   └──────────────┘
                                             │
                                             ▼
                                        ⟨ Exit to ⟩
                                        ⟨   L0    ⟩
```
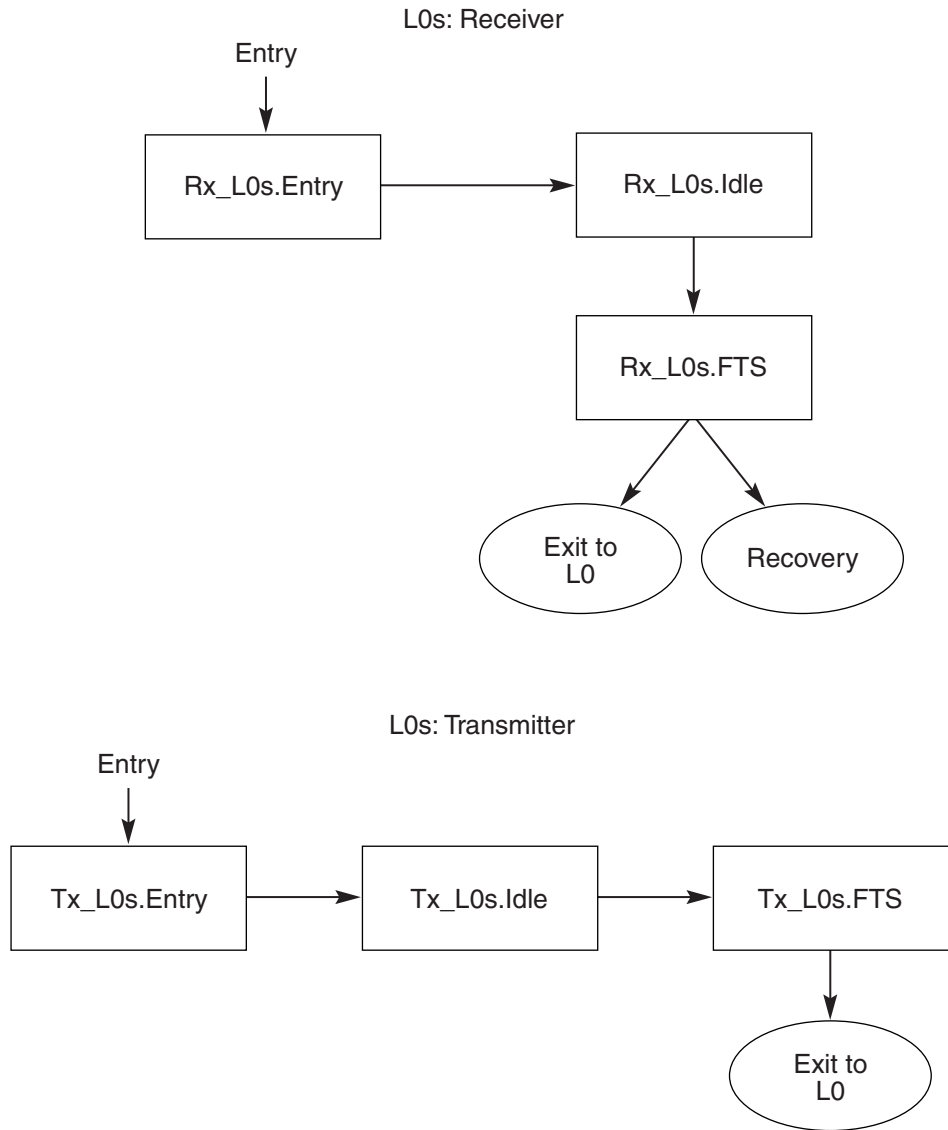
OM13804A

Figure 4-16: L0s Sub-State Machine

4.2.6.7.   L1

The L1 sub-state machine is shown in Figure 4-17.

4.2.6.7.1.   L1.Entry

❑ All configured Transmitters are in Electrical Idle.

5   • Note: The DC common mode voltage must be within specification by
         $T_{TX-IDLE-SET-TO-IDLE}$.[39]

❑ The next state is L1.Idle after a $T_{TX-IDLE-MIN}$ (Table 4-5) timeout.

   • Note: This guarantees that the Transmitter has established the Electrical Idle condition.

4.2.6.7.2.   L1.Idle

10 ❑ Transmitter remains in Electrical Idle.

   • Note: The DC common mode voltage must be within specification.[40]

❑ Next state is Recovery if any Receiver detects exit from Electrical Idle or directed.

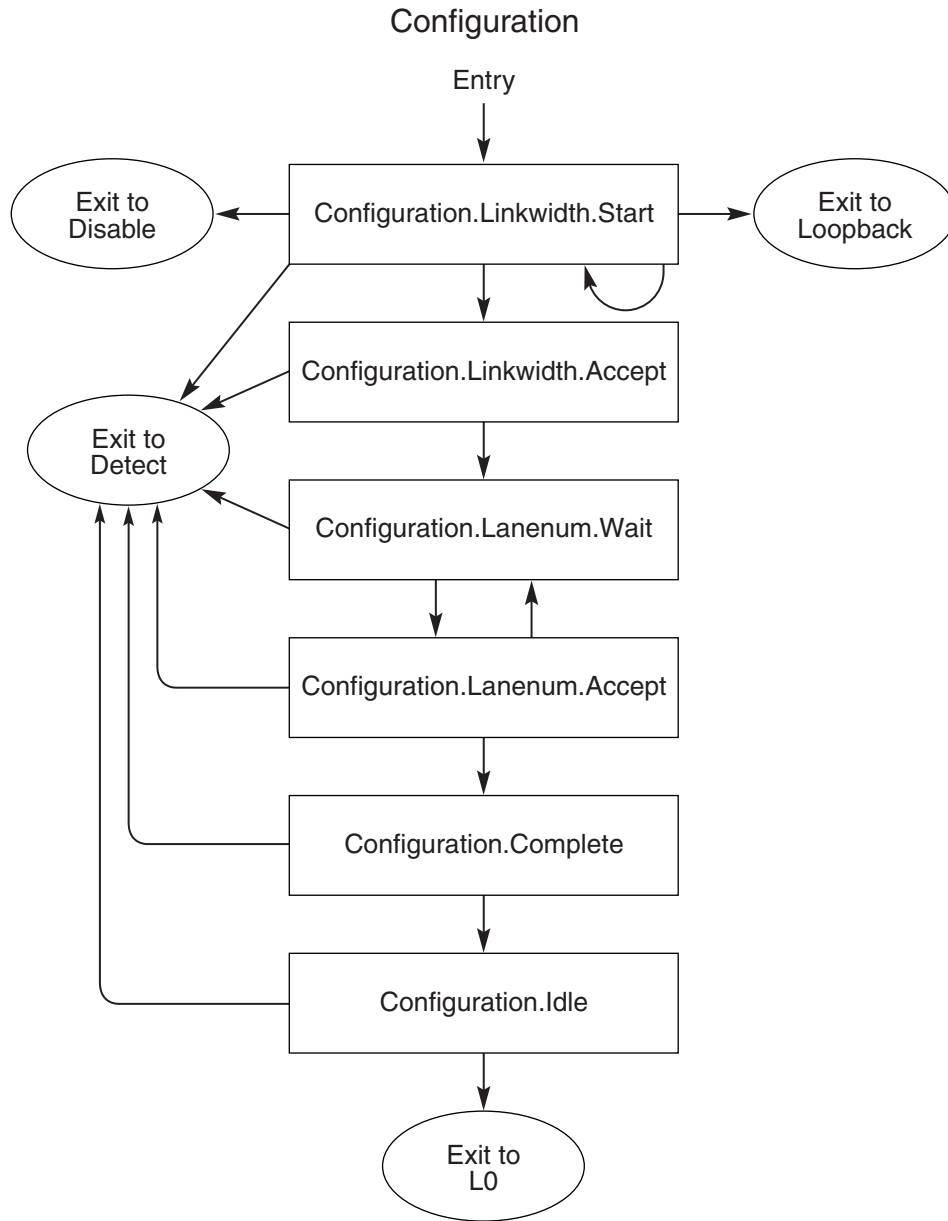[39] The common mode being driven must meet the Absolute Delta Between DC Common Mode During L0 and Electrical Idle ($V_{TX-CM-DC-ACTIVE-IDLE-DELTA}$) specification (see Table 4-5).

[40] The common mode being driven must meet the Absolute Delta Between DC Common Mode During L0 and Electrical Idle ($V_{TX-CM-DC-ACTIVE-IDLE-DELTA}$) specification (see Table 4-5).

OM13805A
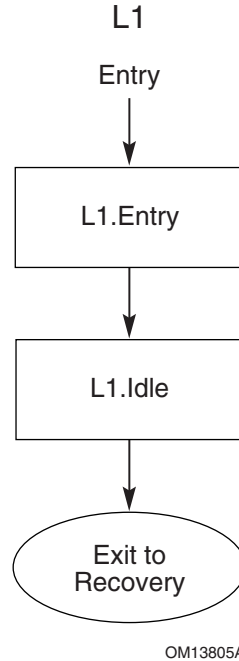
Figure 4-17: L1 Sub-State Machine

4.2.6.8. L2

The L2 sub-state machine is shown in Figure 4-18.

4.2.6.8.1. L2.Idle

❑ All RX Termination must remain enabled in low impedance.

5 ❑ All configured Transmitters must remain in Electrical Idle for a minimum time of $T_{TX\text{-}IDLE\text{-}MIN}$ (Table 4-5).

- • Note: The DC common mode voltage does not have to be within specification.[41]

- • Note: The Receiver needs to wait a minimum of $T_{TX\text{-}IDLE\text{-}MIN}$ to start looking for Electrical Idle Exit.

10 ❑ For Downstream Lanes:

- • For a Root Port, the next state is Detect if a Beacon is received on at least Lane 0 or if directed.

   - ♦ Note: Main power must be restored before entering Detect.

   - ♦ Note: "if directed" is defined as a higher layer decides to exit to Detect.

---

[41] The common mode being driven does not need to meet the Absolute Delta Between DC Common Mode During L0 and Electrical Idle ($V_{TX\text{-}CM\text{-}DC\text{-}ACTIVE\text{-}IDLE\text{-}DELTA}$) specification (see Table 4-5).

- For a Switch, if a Beacon is received on at least Lane 0, the Upstream Port must transition to L2.TransmitWake.

❑ For Upstream Lanes:

- The next state is Detect if Electrical Idle Exit is detected on any Lane.

  ♦ Note: A Switch must transition any Downstream Lanes to Detect.

- Next state is L2.TransmitWake for an Upstream Port if directed to transmit a Beacon.

  ♦ Note: Beacons may only be transmitted on Upstream Ports in the direction of the Root Complex.

4.2.6.8.2. L2.TransmitWake

Note: This state only applies to Upstream Ports.

❑ Transmit the Beacon on at least Lane 0 (Refer to Section 4.3.2.4).

❑ Next state is Detect if Electrical Idle exit is detected on any Upstream Port's Receiver that is in the direction of the Root Complex.

- Note: Power is guaranteed to be restored when Upstream Receivers see Electrical Idle exited, but it may also be restored prior to Electrical Idle being exited.
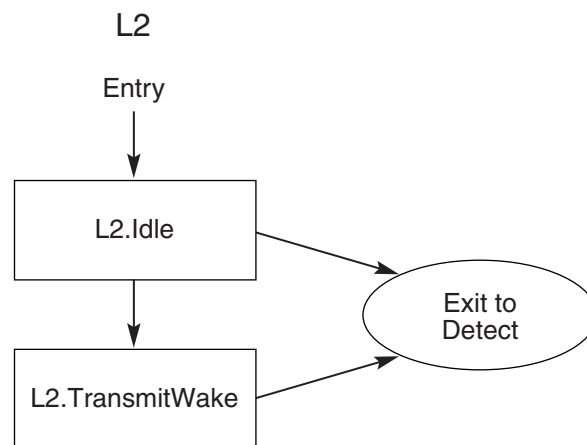
OM13806A

Figure 4-18: L2 Sub-State Machine

4.2.6.9.   Disabled

❑ All Lanes transmit 16 TS1 ordered sets with the Disable Link bit (bit 1) asserted and then transition to Electrical Idle.

- • Note: The Electrical Idle ordered set must be sent prior to entering Electrical Idle.

5   • Note: The DC common mode voltage does not have to be within specification.[42]

❑ If an Electrical Idle ordered set was transmitted and received (even while transmitting TS1 with the Disable Link bit asserted), then:

- • LinkUp = 0 (False)

    - ♦ Note: At this point, the Lanes are considered Disabled.

10   • The next state is Detect when directed or if Electrical Idle is exited.

❑ Otherwise, if no Electrical Idle ordered set is received after a 2 ms timeout, the next state is Detect.

Loopback The Loopback sub-state machine is shown in Figure 4-19.

4.2.6.9.1.   Loopback.Entry

15   ❑ LinkUp = 0 (False)

❑ The Loopback Master device transmits TS1 ordered sets with the Loopback bit (bit 2) asserted until the Loopback Master receives identical TS1 ordered sets with the Loopback bit asserted on an implementation specific number of Lanes. The next state is Loopback.Active.

- • Note: This indicates to the Loopback Master that the Loopback Slave has successfully
20   entered Loopback.

- • Note: The Loopback Master timeout is implementation specific but must be less than 100 ms. The exit is to Loopback.Exit.

- • Note: A boundary condition exists when the Loopback Slave transitions to Loopback.Active that can cause the Loopback Slave to discard a scheduled SKP ordered set. If this occurs,
25   the Loopback Master may not see a SKP ordered set for twice the normal SKP ordered-set scheduling interval.

❑ The next state for the Loopback Slave is Loopback.Active.

- • Note: The Lookback Slave must transition on a Transmit Symbol boundary and may truncated any ordered set in progress.

---

[42] The common mode being driven does need to meet the Absolute Delta Between DC Common Mode During L0 and Electrical Idle ($V_{TX-CM-DC-ACTIVE-IDLE-DELTA}$) specification (see Table 4-5).

4.2.6.9.2. Loopback.Active

❑ The Loopback Master must send valid 8b/10b data. The next state of the Loopback Master is Loopback.Exit if directed

❑ A Loopback Slave is required to retransmit the received 10-bit information exactly as received, while continuing to perform clock tolerance compensation:

- SKPs must be added or deleted on a per Lane basis as outlined in Section 4.2.4.7 with the exception that SKPs do not have to be simultaneously added or removed across Lanes of a configured Link.

  ♦ If a SKP ordered set retransmission requires adding a SKP Symbol to accommodate timing tolerance correction, the SKP Symbol is inserted in the retransmitted Symbol stream anywhere adjacent to a SKP Symbol in the SKP ordered set following the COM Symbol. The inserted SKP Symbol must be of the same disparity as the received SKPs Symbol(s) in the SKP ordered set.

  ♦ If a SKP ordered set retransmission requires dropping a SKP Symbol to accommodate timing tolerance correction, the SKP Symbol is simply not retransmitted.

- Note: No modifications of the received 10-bit data is allowed by the Loopback Slave even if it is determined to be an invalid 10-bit code (i.e., no legal translation to a control or data value possible).

❑ Next state of the Loopback Slave is Loopback.Exit when an Electrical Idle ordered set is received or Electrical Idle is detected on any Lane.

- Note: A Loopback Slave must be able to detect an Electrical Idle condition on any Lane with 1 ms of the Electrical Idle ordered set being received by the Loopback Slave.

- Note: During the time after an Electrical Idle ordered set is received and before Electrical Idle is actually detected by the Loopback Slave, the Loopback Slave may receive and transmit undefined 10-bit data.

- The $T_{TX-IDLE-SET-TO-IDLE}$ parameter does not apply in this case since the Loopback Slave may not even detect Electrical Idle until as much as 1 ms after the Electrical Idle ordered set.

❑ The next state of the Loopback Master is Loopback.Exit if directed.

4.2.6.9.3. Loopback.Exit

❑ The Loopback Master sends an Electrical Idle ordered set and enters Electrical Idle on all Lanes for 2 ms.

- The Loopback Master must transition to a valid Electrical Idle condition[43] on all Lanes within $T_{TX-IDLE-SET-TO-IDLE}$ after sending the Electrical Idle ordered set.

- Note: The Electrical Idle ordered set can be useful in signifying the logical end of transmit and compare operations that occurred by the Loopback Master. Any data received by the Loopback Master after the Electrical Idle ordered set should be ignored since it is undefined.

❑ The Loopback Slave must enter Electrical Idle on all Lanes for 2 ms.

- Note: Before entering Electrical Idle the Looback Slave must Loopback all Symbols that were received prior to detecting Electrical Idle. This ensures that the Loopack Master may see the Electrical Idle ordered set to signify the logical end of any Loopback send and compare operations.

❑ The next state of the Loopback Master and Loopback Slave is Detect.
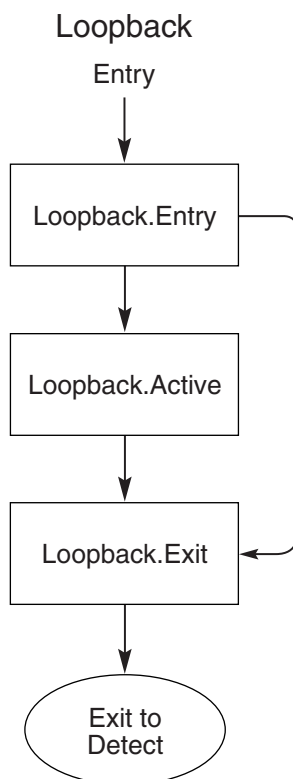
OM13807A

Figure 4-19: Loopback Sub-State Machine

---

[43] The common mode being driven does not need to meet the Absolute Delta Between DC Common Mode During L0 and Electrical Idle ($V_{TX-CM-DC-ACTIVE-IDLE-DELTA}$) specification (see Table 4-5).

4.2.6.10. Hot Reset

❑ Lanes that were directed by a higher Layer to initiate Hot Reset:

- All Lanes in the configured Link transmit TS1 ordered sets with the Hot Reset bit (bit 0) asserted and the configured Link and Lane numbers.

⁵
- If two consecutive TS1 ordered sets are received on any Lane with the Hot Reset bit (bit 0) asserted and configured Link and Lane numbers, then:

  ♦ LinkUp = 0 (False)

  ♦ If no higher Layer is directing the Physical Layer to remain in Hot Reset, the next state is Detect

¹⁰
  ♦ Otherwise, all Lanes in the configured Link continue to transmit TS1 ordered sets with the Hot Reset bit (bit 0) asserted and the configured Link and Lane numbers.

- Otherwise, after a 2 ms timeout next state is Detect.

❑ Lanes that were not directed by a higher Layer to initiate Hot Reset (i.e., received two consecutive TS1 ordered sets with the Hot Reset bit (bit 0) asserted on any configured Lanes):

¹⁵
- LinkUp = 0 (False)

- If any Lane of an Upstream Port of a Switch receives a training sequence with the Hot Reset bit asserted, all configured Downstream Ports must transition to Hot Reset as soon as possible.

  ♦ Note: Any optional crosslinks on the Switch are an exception to this rule and the
²⁰
  behavior is system specific.

- All Lanes in the configured Link transmit TS1 ordered sets with the Hot Reset bit (bit 0) asserted and the configured Link and Lane numbers.

- If two consecutive TS1 ordered sets were received with the Hot Reset bit (bit 0) asserted and the configured Link and Lane numbers, the next state is Hot Reset.

²⁵
- Otherwise, the next state is Detect after a 2 ms timeout.

4.2.7. Clock Tolerance Compensation

SKP ordered sets (defined below) are used to compensate for differences in frequencies between bit rates at two ends of a Link. The Receiver Physical Layer logical sub-block must include elastic buffering which performs this compensation. The interval between SKP ordered set transmissions is derived from the absolute value of the Transmit and Receive clock frequency difference specified in Table 4-5. Having worse case clock frequencies at the limits of the tolerance specified will result in a 600 ppm difference between the Transmit and Receive clocks of a Link. As a result, the Transmit and Receive clocks can shift one clock every 1666 clocks.

4.2.7.1. Rules for Transmitters

❑ All Lanes shall transmit Symbols at the same frequency (the difference between bit rates is 0 ppm within all multi-Lane Links).

❑ When transmitted, the SKP ordered set shall be transmitted simultaneously on all Lanes of a multi-Lane Link (See Section 4.2.4.8 and Table 4-5 for the definition of simultaneous in this context).

❑ The transmitted SKP ordered set is: one COM Symbol followed by three consecutive SKP Symbols

❑ The SKP ordered set shall be scheduled for insertion at an interval between 1180 and 1538 Symbol Times.

❑ Scheduled SKP ordered sets shall be transmitted if a packet or ordered set is not already in progress, otherwise they are accumulated and then inserted consecutively at the next packet or ordered set boundary.

❑ SKP ordered sets do not count as an interruption when monitoring for consecutive characters or ordered set (i.e., eight consecutive TS1 ordered sets in Polling.Active).

❑ SKP ordered-sets must not be transmitted while the Compliance Pattern (see Section 4.2.8) is in progress during Polling.Compliance.

❑ Any and all time spent in any lower power Link state (L0s, L1, L2), does not count in the 1180 to 1538 Symbol interval used to schedule the transmission of SKP ordered sets.

❑ During all lower power Link states any counter(s) or other mechanisms used to schedule SKP ordered sets must be reset.

4.2.7.2. Rules for Receivers

❑ Receivers shall recognize received SKP ordered set consisting of one COM Symbol followed consecutively by one to five SKP Symbols.

₅
- Note: The number of received SKP Symbols in an ordered set shall not vary from Lane-to-Lane in a multi-Lane Link.

❑ Receivers shall be tolerant to receive and process SKP ordered sets at an average interval between 1180 to 1538 Symbol Times.

❑ Receivers shall be tolerant to receive and process consecutive SKP ordered sets.

₁₀
- ♦ Note: Receivers shall be tolerant to receive and process SKP ordered sets that have a maximum separation dependent on the Max_Payload_Size a component supports. The formula for the maximum number of Symbols (N) between SKP ordered-sets is: N = 1538 + (Max_payload_size_byte+26).For example, if Max_Payload_Size is 4096 bytes, N = 1538 + 4096 + 26 = 5660

4.2.8. Compliance Pattern

₁₅ During Polling, the Polling.Compliance sub-state must be entered based on the presence of passive test equipment being attached to one Lane of a possible Link and being detected during the Detect State. The compliance pattern consists of the sequence of 8b/10b Symbols K28.5, D21.5, K28.5, and D10.2 repeating. The compliance sequence is as follows:

Symbol	K28.5	D21.5	K28.5	D10.2
Current Disparity	0	1	1	0
Pattern	0011111010	1010101010	1100000101	0101010101

For any given device that has multiple Lanes, every eighth Lane is delayed by a total of four
₂₀ Symbols. A two Symbol delay occurs at both the beginning and end of the four Symbol sequence.

This delay sequence on every eight Lane is illustrated below:

Symbol:	D	D	K28.5	D21.5	K28.5	D10.2	D	D

Where D is a K28.5 Symbol.

After the eight Symbols are sent, the delay Symbols are advanced to the next Lane and the process is repeated. An illustration of this process is shown below:

Lane 0	D	D	K28.5-	D21.5	K28.5+	D10.2	D	D	K28.5-	D21.5	K28.5+	D10.2
Lane 1	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2	D	D	K28.5-	D21.5
Lane 2	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2
Lane 3	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2
Lane 4	K28.5-	D21.5	K28.5-	D10.2	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2
Lane 5	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2
Lane 6	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2
Lane 7	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2
Lane 8	D	D	K28.5-	D21.5	K28.5+	D10.2	D	D	K28.5-	D21.5	K28.5+	D10.2
Lane 9	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2	D	D	K28.5-	D21.5

Key:

K28.5- Comma when disparity is negative, specifically: "0011111010"

K28.5+ Comma when disparity is positive, specifically: "1100000101"

D21.5   Out of phase data character, specifically: "1010101010"

D10.2   Out of phase data character, specifically: "0101010101"

D   Delay Character – K28.5

This sequence of delays ensures interference between adjacent Lanes, enabling measurement of the compliance pattern under close to worst-case Inter-Symbol Interference and cross-talk conditions.

5   The compliance pattern can be exited only when an Electrical Idle Exit is detected at all the Lanes that detected a Receiver during Detect.

4.3.   Electrical Sub-block

The electrical sub-block contains a Transmitter and a Receiver. The Transmitter is supplied by the logical sub-block with Symbols which it serializes and transmits onto a Lane. The Receiver is
10   supplied with serialized Symbols from the Lane. It transforms the electrical signals into a bit stream which is de-serialized and supplied to the logical sub-block along with a Link clock recovered from the incoming serial stream.

4.3.1.   Electrical Sub-Block Requirements

4.3.1.1.   Clocking Dependencies

15   The Ports on the two ends of a Link must transmit data at a rate that is within 600 parts per million (ppm) of each other at all times. This is specified to allow bit rate clock sources with a +/- 300 ppm tolerance.

4.3.1.1.1. Spread Spectrum Clock (SSC) Sources

The data rate can be modulated from +0% to -0.5% of the nominal data rate frequency, at a modulation rate in the range not exceeding 30 kHz – 33 kHz. The +/- 300 ppm requirement still holds, which requires the two communicating Ports be modulated such that they never exceed a total of 600 ppm difference. For most implementations this places the requirement that both Ports require the same bit rate clock source when the data is modulated with an SSC.

4.3.1.2. AC Coupling

Each Lane of a Link must be AC coupled. The minimum and maximum value for the capacitance is given in Table 4-5. The requirement for the inclusion of AC coupling capacitors on the interconnect media is specified at the Transmitter.

4.3.1.3. Interconnect

In the context of this specification, the interconnect comprises everything between the pins at of a Transmitter package and the pins of a Receiver package. Often, this will consist of traces on a printed circuit board or other suitable medium, AC coupling capacitors and perhaps connectors. The interconnect total capacitance to ground seen by the Receiver Detection circuit (see Section 4.3.1.8) must not exceed 3 nF, including capacitance added by attached test instrumentation. Note that this capacitance is separate and distinct from the AC Coupling capacitance value (see Section 4.3.1.2).

4.3.1.4. Termination

❑ The Transmitter is required to meet RL_{TX-DM}, RL_{TX-CM}, $Z_{TX-DIFF-DC}$, and Z_{TX-DC} (see Table 4-5) any time functional differential signals are being transmitted.

❑ The Transmitter is required to only meet Z_{TX-DC} (see Table 4-5) anytime functional differential signals are not being transmitted.

• Note: The differential impedance during this same time is not defined.

❑ The Receiver is required to meet RL_{RX-DM}, RL_{RX-CM}, $Z_{RX-DIFF-DC}$, and Z_{RX-DC} (see Table 4-6) during all LTSSM states excluding only times during when the device is powered down, Fundamental Reset is asserted, or when explicitly specified.

❑ The Receiver is required to meet $Z_{RX-HIGH-IMP-DC}$ (see Table 4-6) anytime adequate power is not provided to the Receiver, Fundamental Reset is asserted (see Section 4.2.4.5.1), or when explicitly specified.

• Note: The Receiver differential impedance is not defined.

4.3.1.5. DC Common Mode Voltage

The Receiver DC common mode voltage is always 0 V during all states.

The Transmitter DC common mode voltage is held at the same value during all states unless otherwise specified. The range of allowed Transmitter DC common mode values is specified in Table 4-5 ($V_{TX-DC-CM}$).

4.3.1.6. ESD

All signal and power pins must withstand 2000 V of ESD using the human body model and 500 V using the charged device model without damage. Class 2 per JEDEC JESE22-A114-A.

This ESD protection mechanism also helps protect the powered down Receiver from potential common mode transients during certain possible reset or surprise insertion situations.

4.3.1.7. Short Circuit Requirements

All Transmitters and Receivers must support surprise hot insertion/removal without damage to the component. The Transmitter and Receiver must be capable of withstanding sustained short circuit to ground of D+ and D-. The Transmitter short circuit current limit $I_{TX-SHORT}$ is provided in Table 4-5.

4.3.1.8. Receiver Detection

The Receiver Detection circuit is performed by a Transmitter and must correctly detect whether a load impedance of Z_{RX-DC} or lower is present.

The recommended behavior of the Receiver Detection sequence is described below.

Step 1.   A Transmitter must start at a stable voltage prior to the detect common mode shift. This voltage can be VDD, GND, or some other stable common-mode voltage between VDD and GND.

Step 2.   A Transmitter changes the common mode voltage on D+ and D- to a different value.

    a. If the common-mode voltage is equal to VDD, the change must be towards GND.

    b. If the common-mode voltage is equal to GND,the change must be towards VDD.

    c. If the common-mode is set to a stable value between VDD and GND, the direction that the voltage moved to get to this stable initial Electrical Idle voltage is important and the detect common mode shift must be in the opposite direction.

Step 3.   A Receiver is detected based on the rate that the lines change to the new voltage.

a. The Receiver is not present if the voltage at the Transmitter charges at a rate dictated only by the Transmitter impedance and the capacitance of the interconnect and series capacitor.

b. The Receiver is present if the voltage at the Transmitter charges at a rate dictated by the Transmitter impedance, the series capacitor, the interconnect capacitance, and the Receiver termination.

Anytime Electrical Idle is exited the detect sequence does not have to execute or can be aborted on that Lane.

4.3.1.9. Electrical Idle

Electrical Idle is a steady state condition where the Transmitter D+ and D- voltages are held constant at the same value. Electrical Idle is primarily used in power saving and inactive states (i.e., Disable).

Before a Transmitter enters Electrical Idle, it must always send the Electrical Idle ordered set, a K28.5 (COM) followed by three K28.3 (IDL) (see Table 4-5), unless otherwise specified. After sending the last Symbol of the Electrical Idle ordered set, the Transmitter must be in a valid Electrical Idle state as specified by $T_{TX-IDLE-SET-TO-IDLE}$ (see Table 4-5).

The successful reception of an Electrical Idle ordered set occurs upon the receipt of two out of the three K28.3 (IDL) characters in the transmitted Electrical Idle ordered set.

The low impedance common mode and differential Receiver terminations values (see Section 4.3.1.4) must be met in Electrical Idle. The Transmitter can be in either a low or high impedance mode during Electrical Idle.

Any time a Transmitter enters Electrical Idle it must remain in Electrical Idle for a minimum of $T_{TX-IDLE-MIN}$ (see Table 4-5). The Receiver should expect the Electrical Idle ordered set followed by a minimum amount of time in Electrical Idle ($T_{TX-IDLE-SET-TO-IDLE}$) to arm its Electrical Idle Exit detector.

When the Transmitter transitions from Electrical Idle to a valid differential signal level it must meet the output return loss specifications in Table 4-5 and transition from zero differential to full differential voltage consistent with the requirements of the Transmitter Compliance Eye Diagram of Figure 4-24 after the allotted debounce time of $T_{TX-IDLE-TO-TO-DIFF-DATA}$ (see Table 4-5).

Electrical Idle Exit occurs when a signal larger than the minimum $V_{RX-IDLE-DET-DIFFp-p}$ is detected at a Receiver.

4.3.2.   Electrical Signal Specifications

A differential signal is defined by taking the voltage difference between two conductors. In this specification, a differential signal or differential pair is comprised of a voltage on a positive conductor, V_{D+}, and a negative conductor, V_{D-}. The differential voltage (V_{DIFF}) is defined as the difference of the positive conductor voltage and the negative conductor voltage ($V_{DIFF} = V_{D+} - V_{D-}$). The Common Mode Voltage (V_{CM}) is defined as the average or mean voltage present on the same differential pair ($V_{CM} = [V_{D+} + V_{D-}]/2$). This document's electrical specifications often refer to peak-to-peak measurements or peak measurements, which are defined by the following equations.

- ❑ $V_{DIFFp-p} = (2*max|V_{D+} - V_{D-}|)$ (This applies to a symmetric differential swing.)

- ❑ $V_{DIFFp-p} = (max|V_{D+} - V_{D-}| \{V_{D+} > V_{D-}\} + max|V_{D+} - V_{D-}| \{V_{D+} < V_{D-}\})$ (This applies to an asymmetric differential swing.)

- ❑ $V_{DIFFp} = (max|V_{D+} - V_{D-}|)$ (This applies to a symmetric differential swing.)

- ❑ $V_{DIFFp} = (max|V_{D+} - V_{D-}| \{V_{D+} > V_{D-}\})$ or $(max|V_{D+} - V_{D-}| \{V_{D+} < V_{D-}\})$ whichever is greater (This applies to an asymmetric differential swing.)

- ❑ $V_{CMp} = (max|V_{D+} + V_{D-}|/2)$

Note: The maximum value is calculated on a per unit interval evaluation. The maximum function as described is implicit for all peak-to-peak and peak equations throughout the rest of this chapter, and thus a maximum function will not appear in any following subsequent representations of these equations.

In this section, DC is defined as all frequency components below $F_{dc} = 30$ kHz. AC is defined as all frequency components at or above $F_{dc} = 30$ kHz. These definitions pertain to all voltage and current specifications.

An example waveform is shown in Figure 4-20. In this waveform the differential peak-peak signal is approximately 0.6 V, the differential peak signal is approximately 0.3 V and the common mode voltage is approximately 0.25 V.
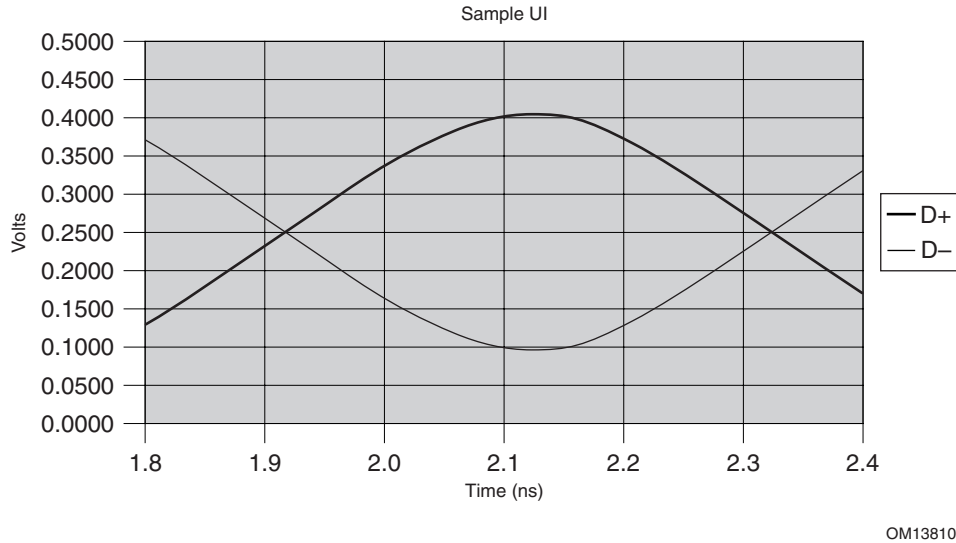
Sample UI

Figure 4-20: Sample Differential Signal

4.3.2.1.   Loss

Loss (attenuation of the differential voltage swing) in this system is a critical parameter that must be properly considered and managed in order to ensure proper system functionality. Failure to properly consider the loss may result in a differential signal swing arriving at the Receiver that does
5   not meet specifications. The interconnect loss is specified in terms of the amount of attenuation or loss that can be tolerated between the Transmitter (TX) and Receiver (RX). The TX is responsible for producing the specified differential eye height at the pins of its package. Together, the TX and the interconnect are responsible for producing the specified differential eye height at the RX pins (see Figure 4-26).

10   An approximate way to understand the worst-case operational loss budget at 1.25 GHz is calculated by taking the minimum output voltage ($V_{TX-DIFFp-p}$ = 800 mV) divided by the minimum input voltage to the Receiver ($V_{RX-DIFFp-p}$ = 175 mV), which results in a maximum loss of 13.2 dB. The approximate way to understand the worst-case operational loss budget at 625 MHz is calculated by taking the minimum de-emphasized output voltage
15   ($V_{TX-DIFFp-p}$ = 505 mV) divided by the minimum input voltage to the Receiver ($V_{RX-DIFFp-p}$ = 175 mV), which results in a maximum loss of 9.2 dB. Although loss vs. frequency is useful in understanding how to design an effective interconnect, the timing and voltage margin measured in the TX and RX eye diagrams end up being the ultimate constraints of insertion Loss. Additional headroom in the loss budget can be achieved by driving a larger differential output
20   voltage (up to the maximum specified in Table 4-5) at the Transmitter.

4.3.2.2. Jitter and BER

Jitter is categorized into random sources (Rj) and deterministic sources (Dj). The total jitter (Tj) is the convolution of the probability density functions for all the independent jitter sources, Rj and Dj. The nature of Rj can be approximated as Gaussian and is used to establish the bit error rate (BER) of the Link.

The UI allocation is given as the allowable Tj at the target BER. The UI allocation must meet a maximum BER of 10^{-12} for the Tj. The allocation to Rj and Dj is not specified.

The methods for measuring the BER compliance are beyond the scope of this specification.

4.3.2.3. De-emphasis

De-emphasis is included (i.e., Generation 1 fundamental band = 250 MHz to 1.25 GHz).

De-emphasis must be implemented when multiple bits of the same polarity are output in succession. Subsequent bits are driven at a differential voltage level 3.5 dB (+/-.5 dB) below the first bit. Note that individual bits, and the first bit from a sequence in which all bits have the same polarity, must always be driven between the Min and Max values as specified by $V_{TX-DIFFp-p}$ in Table 4-5.

The only exception pertains to transmitting the Beacon (see Section 4.3.2.4).

Note: The specified amount of de-emphasis allows for PCI Express designs to optimize maximum interoperability while minimizing complexity of managing configurable de-emphasis values. Thus, the primary benefits of de-emphasis are targeted for worst-case loss budgets of 11-13.2 dB, while being slightly less optimal for lower loss systems. However, this tradeoff is more than offset by the fact that there is inherently more voltage margin in lower loss systems.

An example waveform illustrating de-emphasis and representing the bit sequence (from left to right) of 1001000011 is shown in Figure 4-21.

Sample Data Pattern
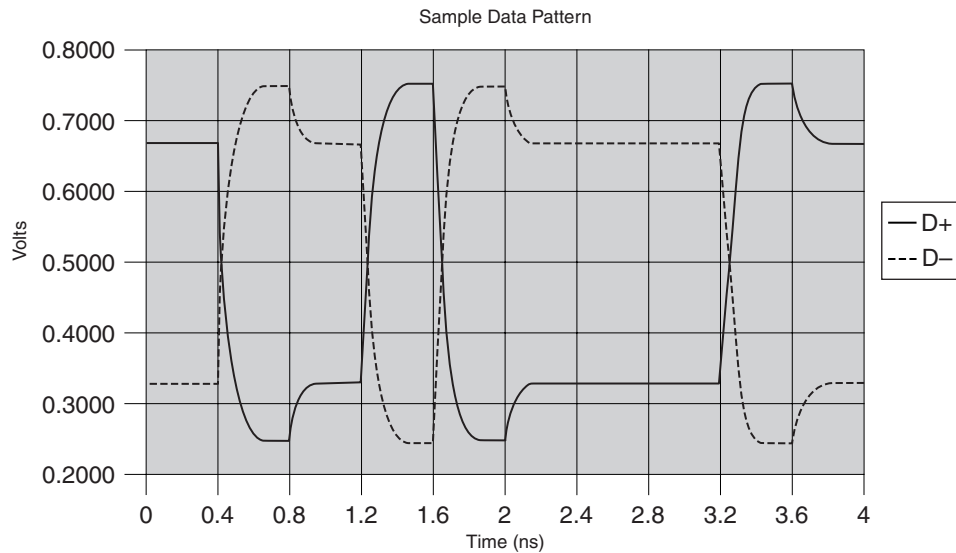
OM14497

**Figure 4-21: Sample Transmitted Waveform Showing -3.5 dB De-emphasis Around a 0.5 V
Common Mode Voltage**

4.3.2.4. Beacon

Support for Beacon is required for all "universal" PCI Express components that support a wakeup
mechanism in order to function in form factors that require the use of Beacon. However, not all
systems and form factor specifications require the use of Beacon, and for components that are
restricted to use in such environments it is not necessary to support Beacon. This section applies to
all components that support Beacon.

The Beacon is a signal sent by a Downstream component to start the exit from an L2 state.

All Transmitter electrical specifications (Table 4-5) must be met while sending a Beacon with the
following exceptions and clarifications.

❑ The Beacon is a DC balanced signal of periodic arbitrary data, which is required to contain some
   pulse widths >= 2 ns but no larger than 16 μs.

❑ The maximum time between qualifying pulses (2 ns <= x <= 16 μs) can be no larger than 16 μs.

❑ DC balance must be always be restored within a maximum time of 32 μs.

❑ Beacon is transmitted in a low impedance mode.

❑ All Beacons must be transmitted and received on at least Lane 0 of multi-Lane Links.[44]

❑ The output Beacon voltage level must be at a -6 dB de-emphasis level for Beacon pulses with a
   width greater than 500 ns.

---

[44] Lane 0 as defined after Link width and Lane reversal negotiations are complete.

❑ The output Beacon voltage level can range between a specified voltage level (see $V_{TX-DIFFp-p}$ in Table 4-5) and a corresponding -3.5 dB de-emphasized voltage levels for Beacon pulses smaller than 500 ns.

❑ The Lane-to-Lane Output Skew (see Table 4-5) and SKP ordered set output (see Section 4.2.7) specifications do not apply.

❑ When any Bridge and/or Switch receives a Beacon at a Downstream Port, that component must propagate a Beacon wakeup indication upstream. This wakeup indication must use the appropriate wakeup mechanism required by the system or form factor associated with the Upstream Port of the Switch (see Section 5.3.3.2).

4.3.2.4.1. Beacon Example

An example Receiver waveform driven at the -6 dB level for a 30 kHz Beacon is shown in Figure 4-22. An example Receiver waveform using the COM character at full speed signaling is shown in Figure 4-23 It should be noted that other waveforms and signaling are possible other than the examples shown in Figure 4-22 and Figure 4-23 (i.e., Polling is another valid Beacon signal).
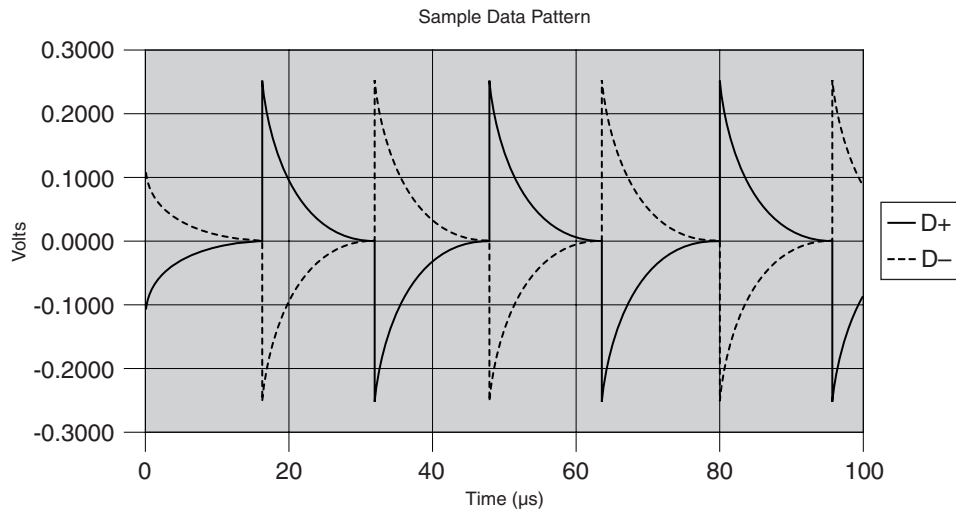
OM13812

Figure 4-22: A 30 kHz Beacon Signaling Through a 75 nF Capacitor

Sample Data Pattern

OM14314A
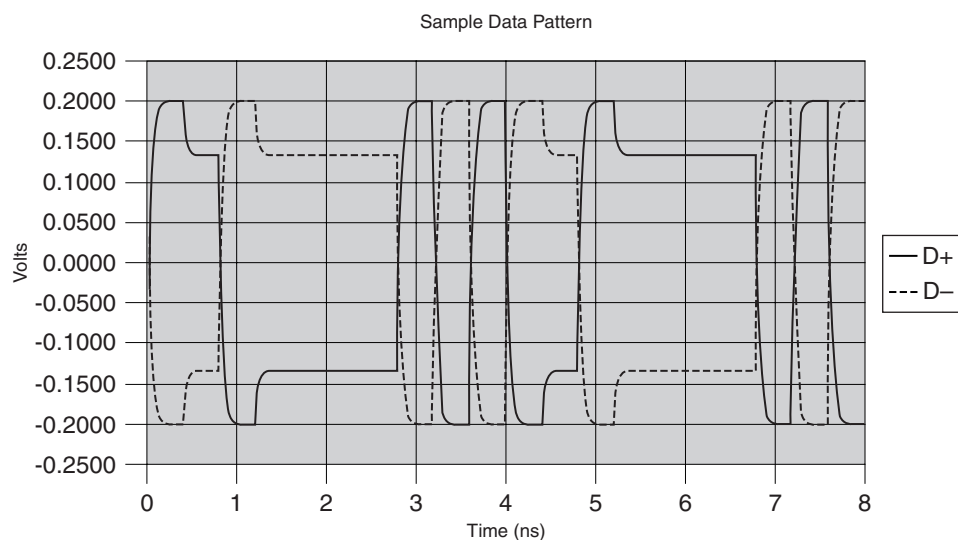
Figure 4-23: Beacon, Which Includes a 2 ns Pulse Through a 75 nF Capacitor

4.3.3.   Differential Transmitter (TX) Output Specifications

The following table defines the specification of parameters for the differential output at all Transmitters (TXs). The parameters are specified at the component pins.

Table 4-5: Differential Transmitter (TX) Output Specifications

Symbol	Parameter	Min	Nom	Max	Units	Comments
UI	Unit Interval	399.88	400	400.12	ps	Each UI is 400 ps +/-300 ppm. UI does not account for SSC dictated variations. See Note 1.
$V_{TX-DIFFp-p}$	Differential Peak to Peak Output Voltage	0.800		1.2	V	$V_{TX-DIFFp-p} = 2*|V_{TX-D+} - V_{TX-D-}|$ See Note 2.
$V_{TX-DE-RATIO}$	De-Emphasized Differential Output Voltage (Ratio)	-3.0	-3.5	-4.0	dB	This is the ratio of the $V_{TX-DIFFp-p}$ of the second and following bits after a transition divided by the $V_{TX-DIFFp-p}$ of the first bit after a transition. See Note 2.
T_{TX-EYE}	Minimum TX Eye Width	0.70			UI	The maximum Transmitter jitter can be derived as $T_{TX-MAX-JITTER} = 1 - T_{TX-EYE} = .3$ UI See Notes 2 and 3.

Symbol	Parameter	Min	Nom	Max	Units	Comments
$T_{TX\text{-}EYE\text{-}MEDIAN\text{-}to\text{-}MAX\text{-}JITTER}$	Maximum time between the jitter median and maximum deviation from the median.			0.15	UI	Jitter is defined as the measurement variation of the crossing points ($V_{TX\text{-}DIFFp\text{-}p.}$ = 0 V) in relation to recovered TX UI. A recovered TX UI is calculated over 3500 consecutive unit intervals of sample data. Jitter is measured using all edges of the 250 consecutive UI in the center of the 3500 UI used for calculating the TX UI. See Notes 2 and 3.
$T_{TX\text{-}RISE}$, $T_{TX\text{-}FALL}$	D+/D- TX Output Rise/Fall Time	0.125			UI	See Notes 2 and 5.
$V_{TX\text{-}CM\text{-}ACp}$	RMS AC Peak Common Mode Output Voltage			20	mV	$V_{TX\text{-}CM\text{-}ACp}$ = RMS($|V_{TX\text{-}D+}$ + $V_{TX\text{-}D\text{-}}|/2 − V_{TX\text{-}CM\text{-}DC}$) $V_{TX\text{-}CM\text{-}DC}$ = DC$_{(avg)}$ of $|V_{TX\text{-}D+}$ + $V_{TX\text{-}D\text{-}}|/2$ See Note 2.
$V_{TX\text{-}CM\text{-}DC\text{-}ACTIVE\text{-}IDLE\text{-}DELTA}$	Absolute Delta of DC Common Mode Voltage During L0 and Electrical Idle.	0		100	mV	$|V_{TX\text{-}CM\text{-}DC\ [during\ L0]} – V_{TX\text{-}CM\text{-}Idle\text{-}DC[During\ Electrical\ Idle.]}| <= 100$ mV $V_{TX\text{-}CM\text{-}DC\ =}$ DC$_{(avg)}$ of $|V_{TX\text{-}D+}$ + $V_{TX\text{-}D\text{-}}|/2$ [L0] $V_{TX\text{-}CM\text{-}Idle\text{-}DC=}$ DC$_{(avg)}$ of $|V_{TX\text{-}D+}$ + $V_{TX\text{-}D\text{-}}|/2$ [Electrical Idle] See Note 2.
$V_{TX\text{-}CM\text{-}DC\text{-}LINE\text{-}DELTA}$	Absolute Delta of DC Common Mode Voltage between D+ and D-	0		25	mV	$|V_{TX\text{-}CM\text{-}DC\text{-}D+} – V_{TX\text{-}CM\text{-}DC\text{-}D\text{-}}| <= 25$ mV $V_{TX\text{-}CM\text{-}DC\text{-}D+\ =}$ DC$_{(avg)}$ of $|V_{TX\text{-}D+}|$ $V_{TX\text{-}CM\text{-}DC\text{-}D\text{-}\ =}$ DC$_{(avg)}$ of $|V_{TX\text{-}D\text{-}}|$ See Note 2.
$V_{TX\text{-}IDLE\text{-}DIFFp}$	Electrical Idle Differential Peak Output Voltage	0		20	mV	$V_{TX\text{-}IDLE\text{-}DIFFp}$ = $|V_{TX\text{-}Idle\text{-}D+}$ - $V_{Tx\text{-}Idle\text{-}D\text{-}}| <= 20$ mV See Note 2.
$V_{TX\text{-}RCV\text{-}DETECT}$	The amount of voltage change allowed during Receiver Detection			600	mV	The total amount of voltage change that a Transmitter can apply to sense whether a low impedance Receiver is present. See Section 4.3.1.8.
$V_{TX\text{-}DC\text{-}CM}$	The TX DC Common Mode Voltage	0		3.6	V	The allowed DC Common Mode voltage under any conditions. See Section 4.3.1.8.
$I_{TX\text{-}SHORT}$	TX Short Circuit Current Limit			90	mA	The total current the Transmitter can provide when shorted to its ground

Symbol	Parameter	Min	Nom	Max	Units	Comments
$T_{TX\text{-}IDLE\text{-}MIN}$	Minimum time spent in Electrical Idle	50			UI	Minimum time a Transmitter must be in Electrical Idle Utilized by the Receiver to start looking for an Electrical Idle Exit after successfully receiving an Electrical Idle ordered set.
$T_{TX\text{-}IDLE\text{-}SET\text{-}TO\text{-}IDLE}$	Maximum time to transition to a valid Electrical Idle after sending an Electrical Idle ordered set			20	UI	After sending an Electrical Idle ordered set, the Transmitter must meet all Electrical Idle specifications within this time. This is considered a debounce time for the Transmitter to meet Electrical Idle after transitioning from L0.
$T_{TX\text{-}IDLE\text{-}TO\text{-}TO\text{-}DIFF\text{-}DATA}$	Maximum time to transition to valid TX specifications after leaving an Electrical Idle condition			20	UI	Maximum time to meet all TX specifications when transitioning from Electrical Idle to sending differential data. This is considered a debounce time for the TX to meet all TX specifications after leaving Electrical Idle.
$RL_{TX\text{-}DIFF}$	Differential Return Loss	12			dB	Measured over 50 MHz to 1.25 GHz. See Note 4.
$RL_{TX\text{-}CM}$	Common Mode Return Loss	6			dB	Measured over 50 MHz to 1.25 GHz. See Note 4.
$Z_{TX\text{-}DIFF\text{-}DC}$	DC Differential TX Impedance	80	100	120	Ω	TX DC Differential Mode Low impedance
$Z_{TX\text{-}DC}$	Transmitter DC Impedance	40			Ω	Required TX D+ as well as D- DC impedance during all states
$L_{TX\text{-}SKEW}$	Lane-to-Lane Output Skew			500 + 2 UI	ps	Static skew between any two Transmitter Lanes within a single Link
C_{TX}	AC Coupling Capacitor	75		200	nF	All Transmitters shall be AC coupled. The AC coupling is required either within the media or within the transmitting component itself.
$T_{crosslink}$	Crosslink Random Timeout	0		1	ms	This random timeout helps resolve conflicts in crosslink configuration by eventually resulting in only one Downstream and one Upstream Port (see 4.2.6.3).

Notes:

1. No test load is necessarily associated with this value.

2. Specified at the measurement point into a timing and voltage compliance test load as shown in Figure 4-25 and measured over any 250 consecutive TX UIs. (Also refer to the Transmitter compliance eye diagram shown in Figure 4-24.)

3. A $T_{TX\text{-}EYE}$ = 0.70 UI provides for a total sum of deterministic and random jitter budget of $T_{TX\text{-}JITTER\text{-}MAX}$ = 0.30 UI for the Transmitter collected over any 250 consecutive TX UIs. The $T_{TX\text{-}EYE\text{-}MEDIAN\text{-}to\text{-}MAX\text{-}JITTER}$ specification ensures a jitter distribution in which the median and the maximum deviation from the

5

median is less than half of the total TX jitter budget collected over any 250 consecutive TX UIs. It should be noted that the median is not the same as the mean. The jitter median describes the point in time where the number of jitter points on either side is approximately equal as opposed to the averaged time value.

4. The Transmitter input impedance shall result in a differential return loss greater than or equal to 12 dB and a common mode return loss greater than or equal to 6 dB over a frequency range of 50 MHz to 1.25 GHz. This input impedance requirement applies to all valid input levels. The reference impedance for return loss measurements is 50 ohms to ground for both the D+ and D- line (i.e., as measured by a Vector Network Analyzer with 50 ohm probes - see Figure 4-25). Note that the series capacitors C_{TX} is optional for the return loss measurement.

5. Measured between 20-80% at Transmitter package pins into a test load as shown in Figure 4-25 for both V_{TX-D+} and V_{TX-D-}.

4.3.3.1. *Transmitter Compliance Eye Diagrams*

The TX eye diagram in Figure 4-24 is specified using the passive compliance/test measurement load (see Figure 4-25) in place of any real PCI Express interconnect + RX component.

There are two eye diagrams that must be met for the Transmitter. Both eye diagrams must be aligned in time using the jitter median to locate the center of the eye diagram. The different eye diagrams will differ in voltage depending whether it is a transition bit or a de-emphasized bit. The exact reduced voltage level of the de-emphasized bit will always be relative to the transition bit.

The eye diagram must be valid for any 250 consecutive UIs.

A recovered TX UI is calculated over 3500 consecutive unit intervals of sample data. The eye diagram is created using all edges of the 250 consecutive UI in the center of the 3500 UI used for calculating the TX UI.[45]

---

[45] It is recommended that the recovered TX UI is calculated using all edges in the 3500 consecutive UI interval with a fit algorithm using a minimization merit function (i.e., least squares and median deviation fits).
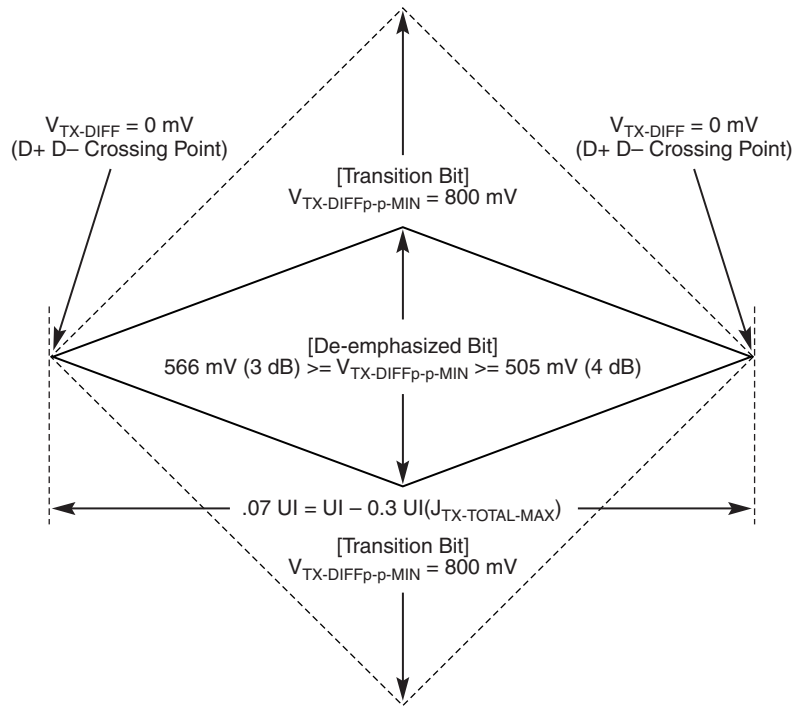
OM13816

Figure 4-24: Minimum Transmitter Timing and Voltage Output Compliance Specifications

4.3.3.2. Compliance Test and Measurement Load

The AC timing and voltage parameters must be verified at the measurement point, as specified by the device vendor within 0.2 inches of the package pins, into a test/measurement load shown in Figure 4-25.

5   Note: The allowance of the measurement point to be within 0.2 inches of the package pins is meant to acknowledge that package/board routing may benefit from D+ and D- not being exactly matched in length at the package pin boundary. If the vendor does not explicitly state where the measurement point is located, the measurement point is assumed to be the D+ and D- package pins.
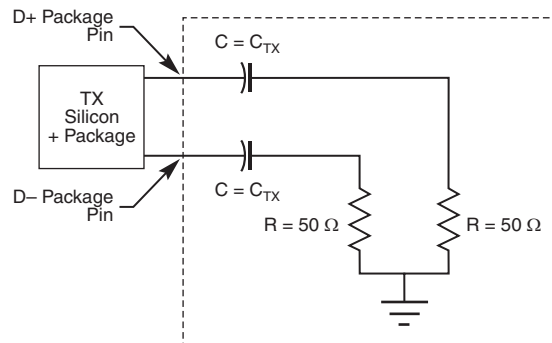
OM13817

Figure 4-25: Compliance Test/Measurement Load

The test load is shown at the Transmitter package reference plane. The same Test/Measurement load is applicable to the Receiver package reference plane at the end of system interconnect in place of the Receiver silicon and package.

When measuring Transmitter output parameters, C_{TX} is an optional portion of the Test/Measurement load. When used, the value of C_{TX} must be in the range of 75 nF to 200 nF. C_{TX} must not be used when the Test/Measurement load is placed in the Receiver package reference plane.

4.3.4.   Differential Receiver (RX) Input Specifications

The following table defines the specification of parameters for all differential Receivers (RXs). The parameters are specified at the component pins.

Table 4-6: Differential Receiver (RX) Input Specifications

Symbol	Parameter	Min	Nom	Max	Units	Comments
	Unit Interval	399.88	400	400.12	ps	The UI is 400 ps +/-300 ppm. UI does not account for SSC dictated variations. See Note 6.
$V_{RX\text{-}DIFFp\text{-}p}$	Differential Input Peak to Peak Voltage	0.175		1.200	V	$V_{RX\text{-}DIFFp\text{-}p} = 2*\lvert V_{RX\text{-}D+} - V_{RX\text{-}D\text{-}}\rvert$ See Note 7.
$T_{RX\text{-}EYE}$	Minimum Receiver Eye Width	0.4			UI	The maximum interconnect media and Transmitter jitter that can be tolerated by the Receiver can be derived as $T_{RX\text{-}MAX\text{-}JITTER} = 1 - T_{RX\text{-}EYE} = 0.6$ UI See Notes 7 and 8.
$T_{RX\text{-}EYE\text{-}MEDIAN\text{-}to\text{-}MAX\text{-}JITTER}$	Maximum time between the jitter median and maximum deviation from the median.			0.3	UI	Jitter is defined as the measurement variation of the crossing points ($V_{RX\text{-}DIFFp\text{-}p} = 0$ V) in relation to a recovered TX UI. A recovered TX UI is calculated over 3500 consecutive unit intervals of sample data. Jitter is measured using all edges of the 250 consecutive UI in the center of the 3500 UI used for calculating the TX UI. See Notes 7 and 8.
$V_{RX\text{-}CM\text{-}ACp}$	AC Peak Common Mode Input Voltage			150	mV	$V_{RX\text{-}CM\text{-}AC} = \lvert V_{RX\text{-}D+} + V_{RX\text{-}D\text{-}}\rvert/2 - V_{RX\text{-}CM\text{-}DC}$ $V_{RX\text{-}CM\text{-}DC} = DC_{(avg)}$ of $\lvert V_{RX\text{-}D+} + V_{RX\text{-}D\text{-}}\rvert/2$ See Note 7.
$RL_{RX\text{-}DIFF}$	Differential Return Loss	15			dB	Measured over 50 MHz to 1.25 GHz with the D+ and D- lines biased at +300 mV and -300 mV, respectively. See Note 9.

Symbol	Parameter	Min	Nom	Max	Units	Comments
RL_{RX-CM}	Common Mode Return Loss	6			dB	Measured over 50 MHz to 1.25 GHz with the D+ and D- lines biased at 0 V. See Note 9.
$Z_{RX-DIFF-DC}$	DC Differential Input Impedance	80	100	120	Ω	RX DC Differential Mode impedance. See Note 10.
Z_{RX--DC}	DC Input Impedance	40	50	60	Ω	Required RX D+ as well as D- DC impedance (50 Ω +/- 20% tolerance). See Notes 7 and 10.
$Z_{RX-HIGH-IMP-DC}$	Powered Down DC Input Impedance	200 k			Ω	Required RX D+ as well as D- DC impedance when the Receiver terminations do not have power. See Note 11.
$V_{RX-IDLE-DET-DIFFp-p}$	Electrical Idle Detect Threshold	65		175	mV	$V_{RX-IDLE-DET-DIFFp-p} = 2*|V_{RX-D+} - V_{RX-D-}|$ Measured at the package pins of the Receiver
$T_{RX-IDLE-DET-DIFF-ENTERTIME}$	Unexpected Electrical Idle Enter Detect Threshold Integration Time			10	ms	An unexpected Electrical Idle ($V_{RX-DIFFp-p} < V_{RX-IDLE-DET-DIFFp-p}$) must be recognized no longer than $T_{RX-IDLE-DET-DIFF-ENTERTIME}$ to signal an unexpected idle condition.
$L_{RX-SKEW}$	Total Skew			20	ns	Skew across all Lanes on a Link. This includes variation in the length of a SKP ordered set (e.g., COM and one to five SKP Symbols) at the RX as well as any delay differences arising from the interconnect itself.

Notes:

6. No test load is necessarily associated with this value.

7. Specified at the measurement point and measured over any 250 consecutive UIs. The test load in Figure 4-25 should be used as the RX device when taking measurements (also refer to the Receiver compliance eye diagram shown in Figure 4-26). If the clocks to the RX and TX are not derived from the same reference clock, the TX UI recovered from 3500 consecutive UI must be used as a reference for the eye diagram.

8. A $T_{RX-EYE} = 0.40$ UI provides for a total sum of 0.60 UI deterministic and random jitter budget for the Transmitter and interconnect collected any 250 consecutive UIs. The $T_{RX-EYE-MEDIAN-to-MAX-JITTER}$ specification ensures a jitter distribution in which the median and the maximum deviation from the median is less than half of the total .6 UI jitter budget collected over any 250 consecutive TX UIs. It should be noted that the median is not the same as the mean. The jitter median describes the point in time where the number of jitter points on either side is approximately equal as opposed to the averaged time value. If the clocks to the RX and TX are not derived from the same reference clock, the TX UI recovered from 3500 consecutive UI must be used as the reference for the eye diagram.

9. The Receiver input impedance shall result in a differential return loss greater than or equal to 15 dB with the D+ line biased to 300 mV and the D- line biased to -300 mV and a common mode return loss greater than or equal to 6 dB (no bias required) over a frequency range of 50 MHz to 1.25 GHz. This input impedance requirement applies to all valid input levels. The reference impedance for return loss measurements for is 50 ohms to ground for both the D+ and D- line (i.e., as measured by a Vector

Network Analyzer with 50 ohm probes - see Figure 4-25). Note: that the series capacitors C_{TX} is optional for the return loss measurement.

10. Impedance during all LTSSM states. When transitioning from a Fundamental Reset to Detect (the initial state of the LTSSM) there is a 5 ms transiton time before Receiver termination values must be met on all un-configured Lanes of a Port.

11. The RX DC Common Mode Impedance that exists when no power is present or Fundamental Reset is asserted. This helps ensure that the Receiver Detect circuit will not falsely assume a Receiver is powered on when it is not. This term must be measured at 300 mV above the RX ground.

12. It is recommended that the recovered TX UI is calculated using all edges in the 3500 consecutive UI interval with a fit algorithm using a minimization merit function. Least squares and median deviation fits have worked well with experimental and simulated data.

The RX eye diagram in Figure 4-26 is specified using the passive compliance/test measurement load (see Figure 4-25) in place of any real PCI Express RX component.

Note: In general, the minimum Receiver eye diagram measured with the compliance/test measurement load (see Figure 4-25) will be larger than the minimum Receiver eye diagram measured over a range of systems at the input Receiver of any real PCI Express component. The degraded eye diagram at the input Receiver is due to traces internal to the package as well as silicon parasitic characteristics which cause the real PCI Express component to vary in impedance from the compliance/test measurement load. The input Receiver eye diagram is implementation specific and is not specified. RX component designer should provide additional margin to adequately compensate for the degraded minimum Receiver eye diagram (shown in Figure 4-26) expected at the input Receiver based on some adequate combination of system simulations and the Return Loss measured looking into the RX package and silicon.[46]

The RX eye diagram must be aligned in time using the jitter median to locate the center of the eye diagram.

The eye diagram must be valid for any 250 consecutive UIs.

A recovered TX UI is calculated over 3500 consecutive unit intervals of sample data. The eye diagram is created using all edges of the 250 consecutive UI in the center of the 3500 UI used for calculating the TX UI.

---

[46] The reference impedance for return loss measurements is 50 Ω to ground for both the D+ and D- line (i.e., as measured by a Vector Network Analyzer with 50 Ω probes - see Figure 4-25). Note that the series capacitors, CTX, are optional for the return loss measurement.
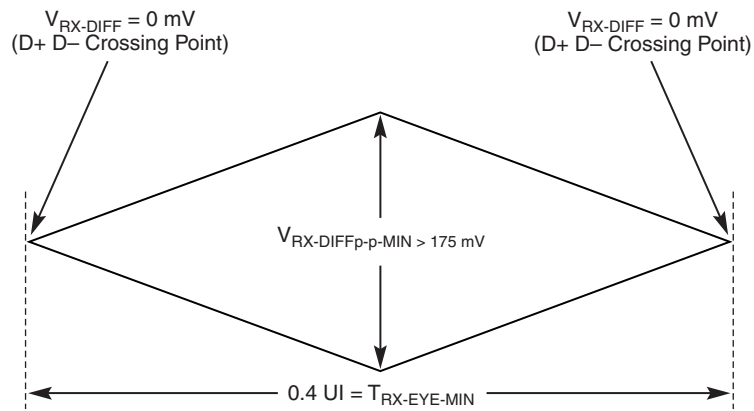
OM13818

Figure 4-26: Minimum Receiver Eye Timing and Voltage Compliance Specification

5

5.   Power Management

This chapter describes PCI Express power management (PCI Express-PM) capabilities and protocols.

5.1.   Overview

5   PCI Express-PM provides the following services:

❑ A mechanism to identify power management capabilities of a given function

❑ The ability to transition a function into a certain power management state

❑ Notification of the current power management state of a function

❑ The option to wakeup the system on a specific event

10   PCI Express-PM is compatible with the *PCI Bus Power Management Interface Specification, Revision 1.1* (PCI-PM), and the *Advanced Configuration and Power Interface Specification, Revision 2.0* (ACPI). This chapter also defines PCI Express native power management extensions. These provide additional power management capabilities beyond the scope of the *PCI Power Management Interface Specification*.

PCI Express-PM defines Link power management states that a PCI Express physical Link is
15   permitted to enter in response to either software driven D-state transitions or active state Link power management activities. PCI Express Link states are not visible directly to legacy bus driver software, but are derived from the power management state of the components residing on those Links. Defined Link states are L0, L0s, L1, L2, and L3. The power savings increase as the Link state transitions from L0 through L3.

20   PCI Express components are permitted to wakeup the system using a wakeup mechanism followed by a power management event (PME) Message. PCI Express systems may provide the optional auxiliary power supply (Vaux) needed for wakeup operation from states where the main power supplies are off. PCI Express-PM extends beyond the PME mechanism defined in conventional PCI-PM as PCI Express PME Messages include the Requestor ID of the requesting agent. These
25   PME Messages are in-band TLPs routed from the requesting device towards the Root Complex.

Another distinction of the PCI Express-PM PME mechanism is in its separation of the following two tasks that are associated with PME:

❑ Reactivation (wakeup) of the associated resources (i.e., re-establishing reference clocks and main power rails to the PCI Express components)

5 ❑ Sending a PME Message to the Root Complex

An autonomous hardware based active state mechanism, Active State Power Management, (ASPM) enables power savings even when the connected components are in the D0 state. After a period of idle Link time, the ASPM mechanism engages in a Physical Layer protocol that places the idle Link into a lower power state. Once in the lower power state, transitions to the fully operative L0 state
10 are triggered by traffic appearing on either side of the Link. Endpoints initiate entry into a low power Link state. This feature may be disabled by software. Refer to Section 5.4.1 for more information on ASPM.

Throughout this document the term Upstream component, or Upstream device, is used to refer to the PCI Express component that is on the end of the PCI Express Link that is hierarchically closer
15 to the root of the PCI Express tree hierarchy. The term Downstream component, or Downstream device, is used to refer to the PCI Express component that is on the end of the Link that is hierarchically further from the root of the PCI Express tree hierarchy.

5.1.1.   Statement of Requirements

All PCI Express components, with exception of the Root Complex, are required to meet or exceed
20 the minimum requirements defined by the PCI-PM Software compatible PCI Express-PM features. Root Complexes are required to participate in Link power management DLLP protocols initiated by a Downstream device, when all functions of a Downstream component enter a PCI-PM Software compatible low power state. For further detail, refer to Section 5.3.2.

ASPM is a required feature (L0s entry at minimum) for all components including Root Complexes,
25 and is configured separately via the native PCI Express configuration mechanisms. Refer to Section 5.4.1 for more information on ASPM.

5.2.   Link State Power Management

PCI Express defines Link power management states, replacing the bus power management states that were defined by the PCI-PM specification. Link states are not visible to PCI-PM legacy
30 compatible software, and are either derived from the power management D-states of the corresponding components connected to that Link or by ASPM protocols (refer to Section 5.4.1).

Note that the PCI Express Physical Layer may define additional intermediate states. See Chapter 4 for more detail on each state and how the Physical Layer handles transitions between states.

PCI Express-PM defines the following Link power management states:

❑ L0 – Active state.

All PCI Express transactions and other operations are enabled.

L0 support is required for both ASPM and PCI-PM compatible power management

❑ L0s – A low resume latency, energy saving "standby" state.

L0s support is required for ASPM. It is not applicable to PCI-PM compatible power management.

All main power supplies, component reference clocks, and components' internal PLLs must be active at all times during L0s. TLP and DLLP communication over a Link that is in L0s is prohibited. The L0s state is used exclusively for ASPM.

The PCI Express Physical Layer provides mechanisms for quick transitions from this state to the L0 state. When common (distributed) reference clocks are used on both sides of a given Link, the transition time from L0s to L0 is typically less than 100 Symbol Times.

It is possible for the Transmit side of one component on a Link to be in L0s while the Transmit side of the other component on the Link is in L0.

❑ L1 – Higher latency, lower power "standby" state.

L1 support is required for PCI-PM compatible power management. L1 is optional for ASPM.

All platform provided main power supplies and component reference clocks must remain active at all times during L1. The component's internal PLLs may be shut off during L1, enabling greater energy savings at a cost of increased exit latency.[47]

The L1 state is entered whenever all functions of a Downstream component on a given PCI Express Link are either programmed to a D-state other than D0, or if the Downstream component requests L1 entry (ASPM) and receives positive acknowledgement for the request.

Exit from L1 is initiated by an upstream initiated transaction targeting the Downstream component, or by the Downstream component's desire to initiate a transaction heading upstream. Transition from L1 to L0 is typically a few microseconds.

TLP and DLLP communication over a Link that is in L1 is prohibited.

❑ L2/L3 Ready – Staging point for L2 or L3

L2/L3 Ready transition protocol support is required

The L2/L3 Ready state is related to PCI-PM D-state transitions. L2/L3 Ready is a transitional pseudo-state (not corresponding directly to the state of the LTSSM) that a given Link enters into when the platform is actively executing the process to remove power and clocks from the downstream component or both attached components. This process is initiated after PM software transitions a device into a D3 state, and subsequently calls power management software to initiate the removal of power and clocks. After the Link enters the L2/L3 Ready state the component(s) are ready for power removal. Depending upon the platform's implementation

---

[47] For example, disabling the internal PLL may be something that is desirable when in D3$_{hot}$, but not so when in D1 or D2.

choices with respect to providing a Vaux supply, after main power has been removed the Link will either settle into L2 (i.e., Vaux is provided), or it will settle into a zero power "off" state (see L3). Note that these are PM pseudo-states for the Link; under these conditions, the LTSSM will in, general, operate only on main power, and so will power off with main power removal.

5   The L2/L3 Ready state entry transition process must begin as soon as possible following the acknowledgment of a PM_Turn_Off Message, (i.e., the injection of a PME_TO_Ack TLP). The Downstream component initiates L2/L3 Ready entry by injecting a PM_Enter_L23 DLLP onto its Transmit Port. Refer to Section 5.6 for further detail on power management system Messages.

10   TLP and DLLP communication over a Link that is in L2/L3 Ready is not possible.

Once the PME_Turn_Off/PME_TO_Ack sequence is initiated, the downstream component is guaranteed that a Fundamental Reset will occur (with or without the removal of power). Exit from L2/L3 Ready back to L0 must only be initiated through the power controller, and must follow entry into D3$_{cold}$ or Fundamental Reset.

15   Note: Exit from L2/L3 ready back to L0 will be through intermediate LTSSM states. Refer to Chapter 4 for detailed information.

❑ L2 – Auxiliary powered Link deep energy saving state.

L2 support is optional, and dependent upon platform support of Vaux.

A component may only consume Vaux power if enabled to do so as described in Section 5.5.1.

20   L2 – The component's main power supply inputs and reference clock inputs are shut off.

- When in L2 (with main power removed), any Link reactivation wakeup logic (Beacon or WAKE#), PME context, and any other "keep alive" logic is powered by Vaux.

  TLP and DLLP communication over a Link that is in L2 is not possible.

❑ L3 – Link Off state.

25   Zero power state.

Refer to Section 4.2 for further detail relating to entering and exiting each of the PCI Express L-states between L0 and L2/L3 Ready (L2.Idle from the Chapter 4 perspective). The L2 state is an abstraction for PM purposes distinguished by the presence of auxiliary power, and should not be construed to imply a requirement that the LTSSM remain active. The L3 state is not defined or

30   discussed in the electrical section (as the component is completely un-powered) with the exception of defining un-powered electrical driver and Receiver specifications.

Figure 5-1 highlights the legitimate L-state transitions that may occur during the course of Link operation.

OM13819
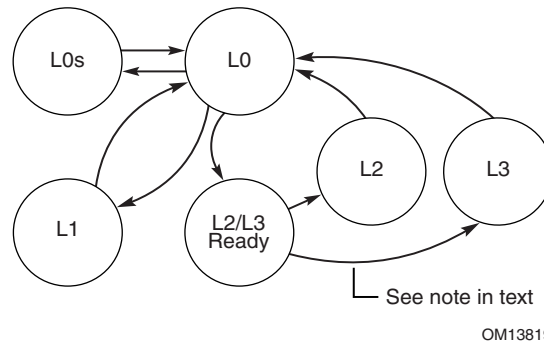
Figure 5-1: Link Power Management State Transitions

Note that these states and state transitions do not correspond directly to the actions of the Physical Layer LTSSM. For example, the LTSSM is typically powered by main power (not Vaux), and so in both the L2 and L3 states will be unpowered. The arc noted in Figure 5-1 indicates the case where the platform does not provide Vaux. In this case, the L2/L3 Ready state transition protocol results in a state of readiness for loss of main power, and once main power is removed the Link settles into the L3 state.

Link PM transitions from any L-state to any other L-state must pass through the L0 state during the transition process with the exception of the L2/L3 Ready to L2 or L3 transitions. In this case, the Link transitions from L2/L3 Ready directly to either L2 or L3 when main power to the component is removed. (This follows along with a corresponding component's D-state transition from $D3_{hot}$ to $D3_{cold}$.)

The following sequence, leading up to entering a system sleep state, illustrates the multi-step Link state transition process:

1. System Software directs all functions of a Downstream component to $D3_{hot}$.

2. The Downstream component then initiates the transition of the Link to L1 as required.

3. System Software then causes the Root Complex to broadcast the PME_Turn_Off Message in preparation for removing the main power source.

4. This Message causes the subject Link to transition back to L0 in order to send it, and to enable the Downstream component to respond with PME_TO_Ack.

5. After the PME_TO_Ack is sent, the Downstream component then initiates the L2/L3 Ready transition protocol.

$$L0 \rightarrow L1 \rightarrow L0 \rightarrow L2/L3 \text{ Ready}$$

Table 5-1 summarizes each L-state, describing when they are used, and the PCI Express platform, and PCI Express component behaviors that correspond to each of them.

A "Yes" entry indicates that support is required (unless otherwise noted). "On" and "Off" entries indicate the required clocking and power delivery. "On/Off" indicates an optional design choice.

Table 5-1: Summary of PCI Express Link Power Management States

	L-State Description	Used by S/W Directed PM	Used by ASPM	Platform Reference Clocks	Platform Main Power	Component Internal PLL	Platform Vaux
L0	Fully active Link	Yes (D0)	Yes (D0)	On	On	On	On/Off
L0s	Standby state	No	Yes [1] (D0)	On	On	On	On/Off
L1	Lower power standby	Yes (D1-D3$_{hot}$)	Yes[2] (opt., D0)	On	On	On/Off [3]	On/Off
L2/L3 Ready	Staging point for power removal	Yes [4]	No	On	On	On/Off	On/Off
L2	Low power sleep state (all clocks, main power off)	Yes [5]	No	Off	Off	Off	On [6]
L3	Off (zero power)	n/a	n/a	Off	Off	Off	Off

Notes:

1. L0s exit latency will be greatest in Link configurations characterized by independent reference clock inputs for components connected to opposite ends of a given Link (vs. a common, distributed reference clock).

2. L1 entry may be requested within ASPM protocol, however its support is optional.

3. L1 exit latency will be greatest for components that internally shut off their PLLs during this state.

4. L2/L3 Ready entry sequence is initiated at the completion of the PME_Turn_Off/PME_TO_Ack protocol handshake . It is not directly affiliated with a D-State transition, or a transition in accordance with ASPM policies and procedures.

5. Depending upon the platform implementation, the system's sleep state may utilize the L2 state or transition to being fully off (L3). L2/L3 Ready state transition protocol is initiated by the Downstream component following reception and TLP acknowledgement of the PME_Turn_Off TLP Message. While platform support for an L2 sleep state configuration is optional (i.e., support for Vaux delivery), PCI Express component protocol support for transitioning the Link to the L2/L3 Ready state is required.

6. L2 is distinguished from the L3 state only by the presence of Vaux. After the completion of the L2/L3 Ready state transition protocol and before main power has been removed, the Link has indicated its readiness for main power removal.

5.3.   PCI-PM Software Compatible Mechanisms

5.3.1.   Device Power Management States (D-States) of a Function

PCI Express supports all PCI-PM device power management states. All functions must support the
D0 and D3 states (both $D3_{hot}$ and $D3_{cold}$). The D1 and D2 states are optional. Refer to the *PCI Bus
Power Management Interface Specification* for further detail relating to the PCI-PM compatible features
described in this specification. Note that where this specification defines detail that departs from
the *PCI Bus Power Management Interface Specification*, this specification takes precedence for PCI Express
components and Link hierarchies.

5.3.1.1.   D0 State

All PCI Express functions must support the D0 state. D0 is divided into two distinct sub-states, the
"un-initialized" sub-state and the "active" sub-state. When a PCI Express component initially has
its power applied, it defaults to the $D0_{uninitialized}$ state. Components that are in this state will be
enumerated and configured by the PCI Express Hierarchy enumeration process. Following the
completion of the enumeration and configuration process the function enters the $D0_{active}$ state, the
fully operational state for a PCI Express function. A function enters the $D0_{active}$ state whenever any
single or combination of the function's Memory Space Enable, I/O Space Enable, or Bus Master
Enable bits have been enabled by system software

5.3.1.2.   D1 State

D1 support is optional. While in the D1 state, a function must not initiate any TLPs on the Link
with the exception of a PME Message as defined in Section 5.3.3. Configuration requests are the
only TLP accepted (as target) by a function that is currently in the D1 state. All other received
Requests must be handled as Unsupported Requests.

Note that a function's software driver participates in the process of transitioning the function from
D0 to D1. It contributes to the process by saving any functional state (if necessary), and otherwise
preparing the function for the transition to D1. As part of this quiescence process the function's
software driver must ensure that any mid-transaction TLPs (i.e., Requests with outstanding
Completions), are terminated prior to handing control to the system configuration software that
would then complete the transition to D1.

5.3.1.3. D2 State

D2 support is optional. While in the D2 state, a function must not initiate any TLPs on the Link with the exception of a PME Message as defined in Section 5.3.3. Configuration requests are the only TLP accepted (as target) by a function that is currently in the D2 state. All other received TLPs must be handled as unsupported packets.

Note that a function's software driver participates in the process of transitioning the function from D0 to D2. It contributes to the process by saving any functional state (if necessary), and otherwise preparing the function for the transition to D2. As part of this quiescence process the function's software driver must ensure that any mid-transaction TLPs (i.e., Requests with outstanding Completions), are terminated prior to handing control to the system configuration software that would then complete the transition to D2.

5.3.1.4. D3 State

D3 support is required, (both the $D3_{cold}$ and the $D3_{hot}$ states). Functions supporting PME generation from D3 must support it for both $D3_{cold}$ and the $D3_{hot}$ states.

Functional context does not need to be maintained by functions in the D3 state. Software is required to re- initialize the function following a $D3 \rightarrow D0$ transition.

The minimum recovery time following a $D3_{hot} \rightarrow D0$ transition is 10 ms. This recovery time may be used by the $D3_{hot} \rightarrow D0$ transitioning component to bootstrap any of its component interfaces (e.g., from serial ROM) prior to being accessible. Attempts to target the function during the recovery time (including configuration request packets) will result in undefined behavior.

5.3.1.4.1. $D3_{hot}$ State

When a function is in $D3_{hot}$, it must respond to configuration accesses targeting it. They must also participate in the PME_Turn_Off/PME_TO_Ack protocol. Refer to Section 5.3.3 details. Once in $D3_{hot}$ the function can later be transitioned into $D3_{cold}$ (by removing power from its host component).

Transitions into the $D3_{hot}$ state are used to establish a standard process for graceful saving of functional state immediately prior to entering a deeper power savings state where power is removed.

Note that a function's software driver participates in the process of transitioning the function from D0 to $D3_{hot}$. It contributes to the process by saving any functional state that would otherwise be lost with removal of main power, and otherwise preparing the function for the transition to $D3_{hot}$. As part of this quiescence process the function's software driver must ensure that any outstanding transactions (i.e., Requests with outstanding Completions), are terminated prior to handing control to the system configuration software that would then complete the transition to $D3_{hot}$.

Note that $D3_{hot}$ is also a useful state for reducing power consumption by idle components in an otherwise running system.

Functions that are in $D3_{hot}$ may be transitioned by software (writing to their PMCSR PowerState field) to the $D0_{uninitialized}$ state. Note that the function is not required to generate an internal hardware reset during or immediately following its transition from $D3_{hot}$ to $D0_{uninitialized}$.

5.3.1.4.2. $D3_{cold}$ State

A function transitions to the $D3_{cold}$ state when its power is removed. A power-on sequence with its associated cold reset transitions a function from the $D3_{cold}$ state to the $D0_{uninititialized}$ state. At this point, software must perform a full initialization of the function in order to re-establish all functional context, completing the restoration of the function to its $D0_{active}$ state.

Functions that support wakeup functionality from $D3_{cold}$ must maintain their PME context (in the Power Management Status/Control register) for inspection by PME service routine software during the course of the resume process.

IMPLEMENTATION NOTE

PME Context

Examples of PME context include, but are not limited to, a function's PME_Status bit, the requesting agent's Requester ID, Caller ID if supported by a modem, IP information for IP directed network packets that trigger a resume event, etc.

A function's PME assertion is acknowledged when system software performs a "write 1 to clear" configuration transaction to the asserting function's PME_Status bit of its PCI-PM compatible PMCSR register.

An auxiliary power source must be used to support PME event detection within a device, Link reactivation, and to preserve PME context from within $D3_{cold}$. Note that once the I/O Hierarchy has been brought back to a fully communicating state, as a result of the Link reactivation, the waking agent then propagates a PME Message to the root of the Hierarchy indicating the source of the PME event. Refer to Section 5.3.3 for further PME specific detail.

5.3.2. PM Software Control of the Link Power Management State

The power management state of a Link is determined by the D-state of its Downstream component.

Table 5-2 depicts the relationships between the power state of a component (Endpoint or Switch) and its Upstream Link.

Table 5-2: Relation Between Power Management States of Link and Components

Downstream Component D-State	Permissible Upstream Component D-State	Permissible Interconnect State
D0	D0	L0, L0s, L1 [1]
D1	D0-D1	L1
D2	D0-D2	L1
$D3_{hot}$	$D0-D3_{hot}$	L1, L2/L3 Ready [2]
$D3_{cold}$	$D0-D3_{cold}$	L2 [3], L3

Notes:

1. All PCI Express components are required to support ASPM with L0s entry during idle at a minimum. The use of L1 within D0 is optional.

2. When all functions within a Downstream component are programmed to $D3_{hot}$ the Downstream component must request the transition of its Link to the L1 state using the PM_Enter_L1 DLLP. Once in $D3_{hot}$, following the execution of a PM_Turn_Off/PME_TO_Ack handshake sequence, the Downstream component must then request a Link transition to L2/L3 Ready using the PM_Enter_L23 DLLP. Following the L2/L3 Ready entry transition protocol the Downstream component must be ready for loss of main power and reference clock.

3. If Vaux is provided by the platform, the Link sleeps in L2. In the absence of V aux, the L-state is L3

The conditions governing Link state transition in the software directed PCI-PM compatible power management scheme are defined as:

❑ A Switch or single function Endpoint device must initiate a Link state transition of its Upstream Port (Switch) or Port (Endpoint), to L1 based solely upon that Port being programmed to D1, D2, or $D3_{hot}$. In the case of the Switch, system software bears the responsibility of ensuring that any D-state programming of a Switch's Upstream Port is done in a compliant manner with respect to PCI Express hierarchy-wide PM policies (i.e., the Upstream Port cannot be programmed to a D-state that is any less active than the most active Downstream Port and Downstream connected component/function(s)).

❑ Multi-function Endpoints must not initiate a Link state transition to L1 until all of their functions have been programmed to a non-D0 D-state.

5.3.2.1. Entry into the L1 State

Figure 5-2 depicts the process by which a Link is transitioned into the L1 state as a direct result of power management software programming the Downstream connected component into a lower power state, (either D1, D2, or $D3_{hot}$ state). This figure and the subsequent description outline the transition process for a single function Downstream component that is being programmed to a non-D0 state.
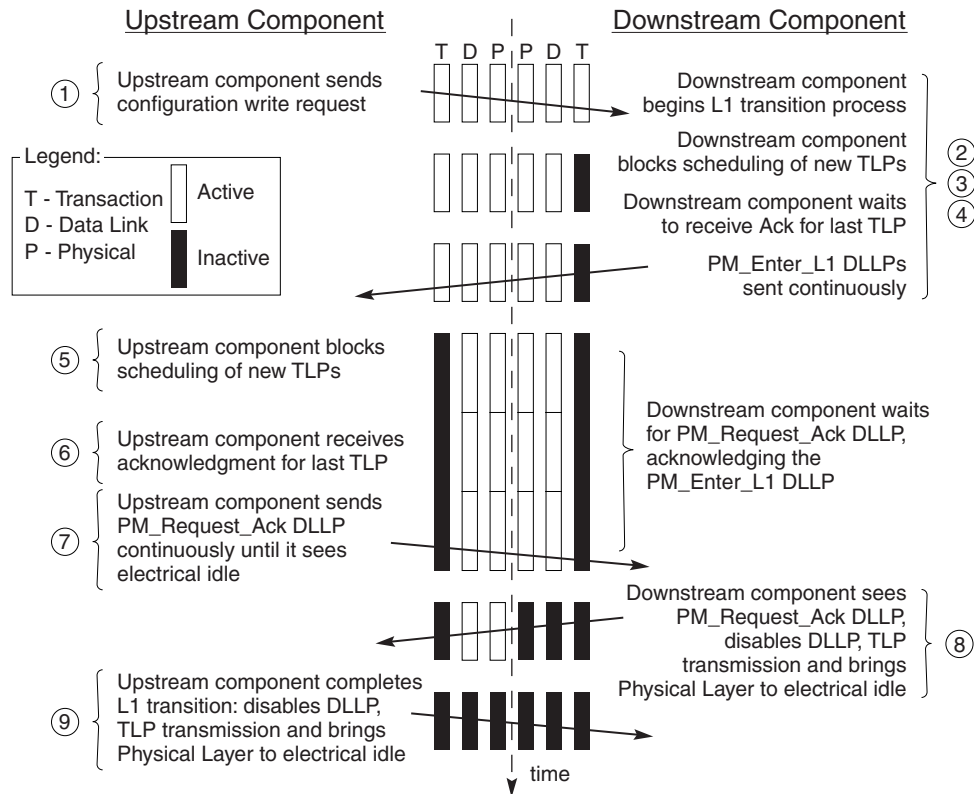
Figure 5-2: Entry into L1 Link State

The following text provides additional detail for the Link state transition process shown in Figure 5-2.

PM Software Request:

1. PM Software (Upstream component) sends a TLP configuration request packet to change the Downstream function's D-state (D1 for example).

Downstream Component Link State Transition Initiation Process:

2. The Downstream component schedules the completion response corresponding to the configuration write to its PMCSR PowerState field. All new TLP scheduling is suspended.

3. The Downstream component then waits until it receives a Link Layer acknowledgement for the PMCSR write completion, and any other TLPs it had previously sent. The component may retransmit a TLP out of its Link Layer Retry buffer if required to do so by Link Layer rules. The Downstream component must also wait until it accumulates at least the minimum number of credits required to send the largest possible packet for any FC type. Note that this is required so that the component can immediately issue a TLP after it exits the L1 state.

4. Once all of the Downstream component's TLPs have been acknowledged, the Downstream component transmits a PM_Enter_L1 DLLP onto its upstream-directed (Transmit) Port. The Downstream component sends this DLLP continuously until it receives a response from the Upstream component[48] (PM_Request_Ack). While waiting for all of its TLPs to be acknowledged, the Downstream component must not initiate any new TLPs. The Downstream component must still, however, continue to accept TLPs and DLLPs from the Upstream component, and it must also continue to respond with DLLPs as needed per Link Layer protocol. (Refer to Chapter 4 for more details on the Physical Layer behavior.)

Upstream Component Link State Transition Process:

5. Upon receiving the PM_Enter_L1 DLLP the Upstream component blocks the scheduling of any future TLPs.

6. The Upstream component then must wait until it receives a Link Layer acknowledgement for the last TLP it had previously sent. The Upstream component may retransmit a TLP from its Link Layer retry buffer if required to do so by the Link Layer rules. The Upstream component must also wait until it accumulates at least the minimum number of credits required to send the largest possible packet for any FC type. Note that this is required so that the component can immediately issue a TLP after it exits the L1 state.

---

[48] If at this point the Downstream component needs to initiate a transfer on the Link, it must first complete the transition to L1 regardless. Once in L1 it is then permitted to initiate an exit L1 to handle the transfer. This corner case represents an event requiring a PME message occurring during the component's transition to L1.

7. Once all of the Upstream component's TLPs have been acknowledged the Upstream component sends a PM_Request_Ack DLLP downstream. The Upstream component sends this DLLP continuously until it observes its receive Lanes enter into the Electrical Idle state. See Chapter 4 for more details on the Physical Layer behavior.[49]

Completing the L1 Link State Transition:

8. Once the Downstream component has captured the PM_Request_Ack DLLP on its Receive Lanes (signaling that the Upstream component acknowledged the transition to L1 request), it then disables DLLP, TLP transmission and brings the upstream directed physical Link into the Electrical Idle state.

9. When the Upstream component observes its Receive Lanes enter the Electrical Idle state, it then stops sending PM_Request_Ack DLLPs, disables DLLP, TLP transmission and brings its Transmit Lanes to Electrical Idle completing the transition of the Link to L1.

    When two components' interconnecting Link is in L1 as a result of the Downstream component being programmed to a non-D0 state, both components suspend the operation of their Flow Control Update, DLLP ACK/NAK Latency, and TLP Completion Timeout counter mechanisms. Refer to the Chapter 4 for more detail on the Physical Layer behavior.

Components on either end of a Link in L1 may optionally disable their internal PLLs in order to conserve more energy. Note, however, that platform supplied main power and reference clocks must continue to be supplied to components on both ends of an L1 Link.

5.3.2.2. Exit from L1 State

L1 exit can be initiated by the component on either end of a PCI Express Link. A Downstream component would initiate an L1 exit transition in order to bring the Link to L0 such that it may then inject a PME Message.

The Upstream component initiates L1 exit to re-establish normal TLP and DLLP communications on the Link.

In either case the physical mechanism for transitioning a Link from L1 to L0 is the same and is described in detail within Chapter 4.

Figure 5-3 outlines a sequence that would trigger an Upstream component to initiate transition of the Link to the L0 state.

---

[49] If, at this point, the Upstream component for any reason needs to initiate a transfer on the Link, it must first complete the transition to L1 regardless. Once in L1, the component is permitted to initiate an exit from L1 to handle the transfer.

OM13821
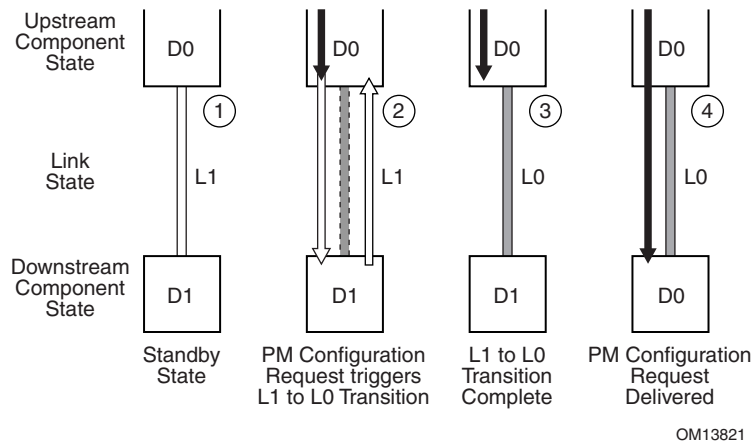
Figure 5-3: Exit from L1 Link State Initiated by Upstream Component

Sequence of events:

1. Power management software initiates a configuration cycle targeting a PM configuration register (the PowerState field of the PMCSR in this example) within a function that resides in the Downstream component (e.g., to bring the function back to the D0 state).

5   2. The Upstream component detects that a configuration cycle is intended for a Link that is currently in a low power state, and as a result, initiates a transition of that Link into the L0 state.

3. In accordance with the Chapter 4 definition, both directions of the Link enter into Link training, resulting in the transition of the Link to the L0 state. The L1 $\rightarrow$ L0 transition is discussed in detail in Chapter 4.

10   4. Once both directions of the Link are back to the active L0 state, the Upstream Port sends the configuration Packet Downstream.

5.3.2.3.   Entry into the L2/L3 Ready State

Transition to the L2/L3 Ready state follows a process that is similar to the L1 entry process. There are some minor differences between the two that are spelled out below.

15   ❑ L2/L3 Ready entry transition protocol does not immediately result in an L2 or L3 Link state. The transition to L2/L3 Ready is effectively a handshake to establish the Downstream component's readiness for power removal. L2 or L3 is ultimately achieved when the platform removes the components' power and reference clock.

❑ The time for L2/L3 Ready entry transition is indicated by the completion of the PME_Turn_Off/PME_TO_Ack handshake sequence. Any actions on the part of the Downstream component necessary to ready itself for loss of power must be completed prior to initiating the transition to L2/L3 Ready. Once all preparations for loss of power and clock are completed, L2/L3 Ready entry is initiated by the Downstream component by sending the PM_Enter_L23 DLLP upstream.

In contrast, the time for L1 entry transition is indicated by programming all of the Downstream component's function(s) to non-D0 states, or by ASPM policies. There are no preparations necessary before initiating a transition to L1.

❑ The Downstream component must be in D3$_{hot}$ prior to being transitioned into the L2/L3 Ready state, i.e., a PME_Turn_Off Message must never be sent unless all functions Downstream of its point of origin are currently in D3$_{hot}$.

In contrast, a Downstream component initiating a transition to L1 would have always initially been in D0, and had just been reprogrammed to D1, D2, or D3$_{hot}$.

❑ L2/L3 Ready entry transition protocol uses the PM_Enter_L23 DLLP.

Note that the PM_Enter_L23 DLLPs are sent continuously until an acknowledgement is received.

5.3.3.   Power Management Event Mechanisms

5.3.3.1. Motivation

The PCI Express PME mechanism is software compatible with the PME mechanism defined by the PCI-PM specification. Power Management Events are generated by PCI Express functions as a means of requesting a PM state change. Power Management Events are typically utilized to revive the system or an individual function from a low power state.

Power management software may transition a PCI Express Hierarchy into a low power state, and transition the Upstream Links of these devices into the non-communicating L2 state.[50] The PCI Express PME generation mechanism is, therefore, broken into two components:

❑ Waking a non-communicating Hierarchy (wakeup). This step is required only if the Upstream Link of the device originating the PME is in the non-communicating L2 state, since in that state the device cannot send a PM_PME Message upstream.

❑ Sending a PM_PME Message to the root of the PCI Express Hierarchy

PME indications are propagated to the Root Complex in the form of TLP Messages. PM_PME Messages include the logical location of the requesting agent within the Hierarchy (in the form of the Requester ID of the PME Message header). Explicit identification within the PM_PME Message is intended to facilitate quicker PME service routine response and, hence, shorter resume time.

---

[50] The L2 state is defined as "non-communicating" since component reference clock and main power supply are removed in that state.

5.3.3.2. Link Wakeup

The Link wakeup mechanisms provide a means of signaling the platform to re-establish power and reference clocks to the components within its domain. There are two defined wakeup mechanisms: Beacon and WAKE#. The Beacon mechanism uses in-band signaling to implement wakeup
functionality, and is described in Section 4.3.2.4. For components that support wakeup functionality, Beacon is the required mechanism for all components, except for components designed exclusively for the following form factors: *PCI Express Card Electromechanical Specification* and *Mini PCI Express Card Electromechanical Specification.* Switch products targeting applications where Beacon is used on some ports of the Switch and WAKE# is used for other ports must translate the
wakeup mechanism appropriately (see the implementation note entitled "Example of WAKE# to Beacon Translation" on page 237). In applications where WAKE# is the only wakeup mechanism used, the Root Complex is not required to support the receipt of Beacon.

The WAKE# mechanism uses sideband signaling to implement wakeup functionality. WAKE# is an "open drain" signal asserted by components requesting wakeup and observed by the associated
power controller. WAKE# is only defined for certain form factors, and the detailed specifications for WAKE# are included in the relevant form factor specifications. Specific form factor specifications may require the use of either Beacon or WAKE# as the wakeup mechanism. All form factors that require WAKE# as the wakeup mechanism must permit components to also generate Beacon, although the Beacon may not be observed.

When WAKE# is used as a wakeup mechanism, once WAKE# has been asserted, the asserting PCI Express function must continue to drive the signal low until main power has been restored to the component as indicated by Fundamental Reset going inactive.

WAKE# is not intended to be used as an input by any Endpoint, and the system is not required to route or buffer it in such a way that a Endpoint is guaranteed to be able to detect that the signal has
been asserted by another function.

Before using any wakeup mechanism, functions must be enabled by software to do so by setting the device's PME_En bit in the Power Management Control/Status register (PMCSR). The PME_Status bit is sticky, and devices must maintain the value of the PME_Status bit through reset if aux power is available and they are enabled for wakeup events.

Systems that allow PME generation from $D3_{cold}$ state must provide auxiliary power to support Link wakeup when the main system power rails are off. A component may only consume auxiliary power if software has enabled it to do so as described in Section 5.5.1. Software is required to enable auxiliary power consumption in all components that participate in Link wakeup, including all components that must propagate the Beacon signal. In the presence of legacy system software, this
is the responsibility of system firmware.

Regardless of the wakeup mechanism used, once the Link has been re-activated and trained, the requesting agent then propagates a PM_PME Message upstream to the Root Complex. From a power management point of view, the two wakeup mechanisms provide the same functionality, and are not distinguished elsewhere in this chapter.

✐ IMPLEMENTATION NOTE

Example of WAKE# to Beacon Translation

Switch products targeting applications that connect "Beacon domains" and "WAKE# domains" must translate the wakeup mechanism appropriately. Figure 5-4 shows two example systems, each including slots that use the WAKE# wakeup mechanism. In Case 1, WAKE# is input directly to the Power Management Controller, and no translation is required. In Case 2, WAKE# is an input to the Switch, and in response to WAKE# being asserted the Switch must generate a Beacon that is propagated to the Root Complex/Power Management Controller.

Case 1: WAKE# routed directly to system
Power Management Controller

Case 2: WAKE# routed to Switch;
Switch in turn generates Beacon
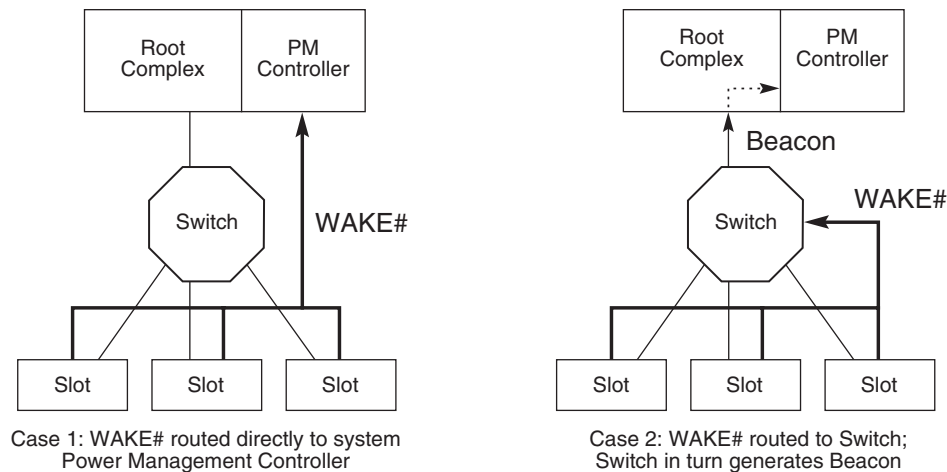
A-0334

Figure 5-4: Conceptual Digrams Showing Two Example Cases of WAKE# Routing

5.3.3.2.1. PME Synchronization

PCI Express devices need to be notified before their reference clock and main power may be removed so that they can prepare for that eventuality. PCI Express-PM introduces a fence mechanism that serves to initiate the power removal sequence while also coordinating the behavior of the platform's power management controller and PME handling by PCI Express agents.

Potential race conditions exist where a Downstream agent, if not somehow coordinated with the platform's power manager, could initiate a PM_PME Message while the power manager was in the process of turning off the main power source to the Link Hierarchy. The net result of hitting this corner condition would be loss of the PME indication. The fence mechanism ensures this does not happen.

PME_Turn_Off Broadcast Message

Before main component power and reference clocks are turned off, the Root Complex or Switch Downstream Port must issue a broadcast Message that instructs all agents downstream of that point within the hierarchy to cease initiation of any subsequent PM_PME Messages, effective immediately upon receipt of the PME_Turn_Off Message.

Each PCI Express agent is required to respond with a TLP "acknowledgement" Packet, PME_TO_Ack that is, as in the case of a PME Message, always routed upstream. In all cases, the PME_TO_Ack Message must terminate at the PME_Turn_Off Message's point of origin.[51]

Note that PM_PME and PME_TO_Ack, like all other PCI Express Message packets, are handled as posted transactions. It is their posted transaction nature that ensures that any previously injected PM_PME Messages will be pushed ahead of the fence acknowledgement assuring full in-order delivery of any previously initiated PM_PME Messages before the "Turn off" acknowledgement ever reaches the initiator of the PME_Turn_Off Message.

For the case where a PME_Turn_Off Message is initiated upstream of a PCI Express Switch, the PCI Express Switch's Upstream Port must report an "aggregate" acknowledgement only after having received PME_TO_Ack packets from each of their Downstream Ports individually. Once a PME_TO_Ack Packet has arrived on all Downstream Ports, the Switch then sends a PME_TO_Ack packet on its Upstream Port.

All PCI Express components must accept[52] and acknowledge the PME_Turn_Off Packet from within the $D3_{hot}$ State. Once an Endpoint has sent a PME_TO_Ack Packet on its Transmit Link, it must then prepare for removal of its power and reference clocks by initiating a transition to the L2/L3 Ready state.

A Switch must also transition its Upstream Link to the L2/L3 Ready state in the same manner as described in the previous paragraph for Endpoints. However, the Switch initiates this transition only after all of its Downstream Ports have entered L2/L3 Ready state.

The Links attached to the originator of the PME_Turn_Off Message are the last to assume the L2/L3 Ready state. This serves as an indication to the power delivery manager[53] that all Links within that portion of the PCI Express hierarchy have:

❑ Successfully retired all in flight PME Messages to the point of PME_Turn_Off Message origin

❑ Performed any necessary local conditioning in preparation for power removal

The power delivery manager must wait a minimum of 100 ns after observing all Links corresponding to the point of origin of the PME_Turn_Off Message enter L2/L3 Ready before removing the components' reference clock and main power.

---

[51] Point of origin for the PME_Turn_Off Message could be all of the Root Ports for a given Root Complex (full platform sleep state transition), an individual Root Port, or a Switch Downstream Port.

[52] FC credits permitting.

[53] Power delivery control within this context relates to control over the entire PCI Express Link hierarchy, or over a subset of PCI Express Links ranging down to a single PCI Express Link and associated Endpoing for sub hierarchies supporting independently managed power and clock distribution.

📝 IMPLEMENTATION NOTE

PME_TO_Ack Message Proxy by Switch Devices

One of the PME_Turn_Off/PME_TO_Ack handshake's key roles is to ensure that all in flight PME Messages are flushed from the PCI Express fabric prior to sleep state power removal. This is guaranteed to occur because PME Messages and the PME_TO_Ack Messages both use the posted request queue within vc0 and so all previously injected PME Messages will be made visible to the system before the PME_TO_Ack is received at the Root Complex. Once all Downstream Ports of the Root Complex receive a PME_TO_Ack Message the Root Complex can then signal the power manager that it is safe to remove power without loss of any PME Messages.

Switches create points of hierarchical expansion and, therefore, must wait for all of their connected Downstream Ports to receive a PME_TO_Ack Message before they can send a PME_TO_Ack Message upstream on behalf of the sub-hierarchy that it has created downstream. This can be accomplished very simply using common score boarding techniques. For example, once a PME_Turn_Off broadcast Message has been broadcast downstream of the Switch, the Switch simply checks off each Downstream Port having received a PME_TO_Ack. Once the last of its active Downstream Ports receives a PME_TO_Ack, the Switch will then send a single PME_TO_Ack Message upstream as a proxy on behalf of the entire sub-hierarchy downstream of it. Note that once a Downstream Port receives a PME_TO_Ack Message and the Switch has scored its arrival, the Port is then free to drop the packet from its internal queues and free up the corresponding posted request queue FC credits.

📝 IMPLEMENTATION NOTE

PME_TO_Ack Deadlock Avoidance

As specified earlier, any device that detects a PME_Turn_Off Message must reply with a PME_TO_Ack Message. However, system behavior must not depend on the correct behavior of any single device. In order to avoid deadlock in the case where one or more devices do not respond with a PME_TO_Ack Message, the power manager must not depend on the acceptance of a PME_TO_Ack Message. For example, the power manager may timeout after waiting for the PME_TO_Ack Message for a given time, after which it proceeds as if the Message was accepted.

5.3.3.3. PM_PME Messages

PM_PME Messages are posted Transaction Layer Packets (TLPs) that inform the power management software which agent within the PCI Express Hierarchy requests a PM state change. PM_PME Messages, like all other Power Management system Messages, must use the general purpose Traffic Class, TC #0.

PM_PME Messages are always routed in the direction of the Root Complex. To send a PM_PME Message on its Upstream Link, a device must transition the Link to the L0 state (if the Link was not in that state already). Unless otherwise noted, the device will keep the Link in the L0 state following the transmission of a PM_PME Message.

5.3.3.3.1. PM_PME "Backpressure" Deadlock Avoidance

A PCI Express Root Complex is typically implemented with local buffering to temporarily store a finite number of PM_PME Messages that could potentially be simultaneously propagating through the PCI Express Hierarchy at any given time. Given a limited number of PM_PME Messages that can be stored within the Root Complex, there can be backpressure applied to the upstream directed posted queue in the event that the capacity of this temporary PM_PME Message buffer is exceeded.

Deadlock can occur according to the following example scenario:

1. Incoming PM_PME Messages fill the Root Complex's temporary storage to its capacity while there are additional PM_PME Messages still in the Hierarchy making their way upstream.

2. The Root Complex, on behalf of system software, issues a configuration read request targeting one of the PME requester's PMCSR (e.g., reading its PME_Status bit).

3. The corresponding split completion Packet is required, as per producer/consumer ordering rules, to push all previously posted PM_PME Messages out ahead of it, which in this case are PM_PME Messages that have no place to go.

4. The PME service routine cannot make progress; the PM_PME Message storage situation does not improve.

5. Deadlock occurs.

Precluding potential deadlocks requires the Root Complex to always enable forward progress under these circumstances. This must be done by accepting any PM_PME Messages that posted queue flow control credits allow for, and discarding any PM_PME Messages that create an overflow condition. This required behavior ensures that no deadlock will occur in these cases, however PM_PME Messages will be discarded and hence lost in the process.

To ensure that no PM_PME Messages are lost permanently, all agents that are capable of generating PM_PME must implement a PME Service Timeout mechanism to ensure that their PME requests are serviced in a reasonable amount of time.

If after 100 ms (+ 50%/- 5%), the PME_Status bit of a requesting agent has not yet been cleared, the PME Service Timeout mechanism expires triggering the PME requesting agent to re-send the temporarily lost PM_PME Message. If at this time the Link is in a non-communicating state, then, prior to re-sending the PM_PME Message, the agent must reactivate the Link as defined in Section 5.3.3.2.

5.3.3.4. PME Rules

❑ All PCI Express components supporting PCI Express-PM must implement the PCI-PM PMC and PMCSR registers in accordance with the PCI-PM specification. These registers reside in the PCI-PM compliant PCI Capability List format.

5 • PME capable functions must implement the PME_Status bit, and underlying functional behavior, in their PMCSR configuration register.

• When a function initiates Link wakeup, or issues a PM_PME Message, it must set its PME_Status bit.

❑ Switches must route a PM_PME received on any Downstream Port to their Upstream Port

10 ❑ PME capable agents must comply with PME_Turn_Off and PME_TO_Ack fence protocols

❑ Before a Link or a portion of a Hierarchy is transferred into a non-communicating state (i.e., a state it cannot issue a PM_PME Message from), a PME_Turn_Off Message must be broadcast Downstream.

5.3.3.5. PM_PME Delivery State Machine

15 The following diagram conceptually outlines the PM_PME delivery control state machine. This state machine determines the ability of a Link to service PME events by issuing PM_PME immediately vs. requiring Link wakeup.
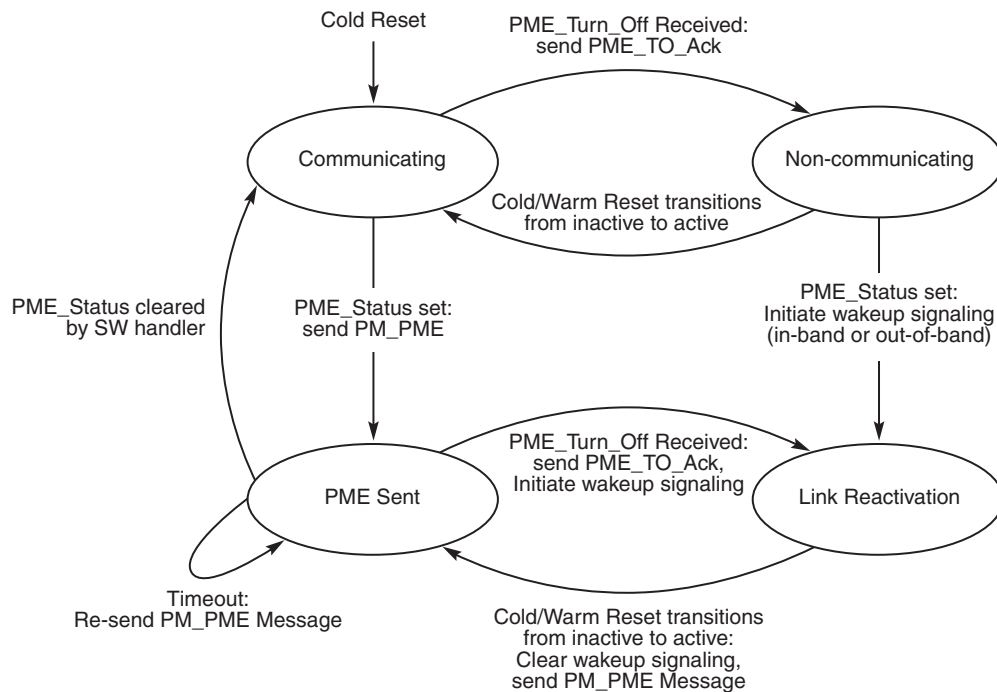
OM13822A

Figure 5-5: A Conceptual PME Control State Machine

Communicating State:

At initial power-up and associated reset, the Upstream Link enters the Communicating state

❑ If PME_Status is asserted (assuming PME delivery is enabled), a PM_PME Message will be issued upstream, terminating at the root of the PCI Express Hierarchy. The next state is the
⁵ PME Sent state

❑ If a PME_Turn_Off Message is received, the Link enters the Non-communicating state following its acknowledgment of the Message and subsequent entry into the L2/L3 Ready state.

Non-communicating State:

❑ Following the restoration of power and clock, and the associated reset, the next state is the
¹⁰ Communicating state.

❑ If PME_Status is asserted, the Link will transition to the Link Reactivation state, and activate the wakeup mechanism.

PME Sent State

❑ If PME_Status is cleared, the function becomes PME Capable again. Next state is the
¹⁵ Communicating state.

❑ If the PME_Status bit is not cleared by the time the PME service timeout expires, a PM_PME Message is re-sent upstream. See Section 5.3.3.3.1 for an explanation of the timeout mechanism.

❑ If a PME Message has been issued but the PME_Status has not been cleared by software when the Link is about to be transitioned into a messaging incapable state (a PME_Turn_Off Message
²⁰ is received), the Link transitions into Link Reactivation state after sending a PME_TO_Ack Message. The device also activates the wakeup mechanism.

Link Reactivation State

❑ Following the restoration of power and clock, and the associated reset, the Link resumes a transaction-capable state. The device clears the wakeup signaling, if necessary, and issues a
²⁵ PM_PME Upstream and transitions into the PME Sent state.

5.4. Native PCI Express Power Management Mechanisms

The following sections define power management features that require new software. While the presence of these features in new PCI Express designs will not break legacy software compatibility,
³⁰ taking the full advantage of them requires new code to manage them.

These features are enumerated and configured using PCI Express native configuration mechanisms as described in Chapter 7 of this specification. Refer to Chapter 7 for specific register locations, bit assignments, and access mechanisms associated with these PCI Express-PM features.

5.4.1.   Active State Power Management (ASPM)

All PCI Express components are required to support the minimum requirements defined herein for Active State Power Management (ASPM). This feature must be treated as being orthogonal to the PCI-PM Software compatible features from a minimum requirements perspective. For example, the Root Complex is exempt from the PCI-PM Software compatible features requirements, however it must implement ASPM's minimum requirements.

Components in the D0 state (i.e., fully active state) normally keep their Upstream Link in the active L0 state, as defined in Section 5.3.2. ASPM defines a protocol for components in the D0 state to reduce Link power by placing their Links into a low power state and instructing the other end of the Link to do likewise. This capability allows hardware-autonomous, dynamic Link power reduction beyond what is achievable by software-only controlled (i.e., PCI-PM Software driven) power management.

Two low power "standby" Link states are defined for ASPM. The L0s low power Link state is optimized for short entry and exit latencies, while providing substantial power savings. If the L0s state is enabled in a device, it is required to bring any Transmit Link into L0s state whenever that Link is not in use (refer to Section 5.4.1.1.1 for details relating to the L0s invocation policy). All PCI Express components must support the L0s Link state from within the D0 device state.

The L1 Link state is optimized for maximum power savings at a cost of longer entry and exit latencies. L1 reduces Link power beyond the L0s state for cases where very low power is required and longer transition times are acceptable. ASPM support for the L1 Link state is optional.

Each PCI Express component must report its level of support for ASPM in the ASPM Support configuration field.

Each PCI Express component shall also report its L0s and L1 exit latency (the time that they require to transition from the L0s or L1 state to the L0 state). Endpoints must also report the worst-case latency that they can withstand before risking, for example, internal fifo overruns due to the transition latency from L0s or L1 to the L0 state. Power management software can use the provided information to then enable the appropriate level of ASPM.

The L0s exit latency may differ significantly if the reference clock for opposing sides of a given Link is provided from the same source, or delivered to each component from a different source. PCI Express-PM software informs each PCI Express device of its clock configuration via the "common clock configuration" bit in their PCI Express Capability structure's Link Control register. This bit serves as the determining factor in the L0s exit latency value reported by the device. ASPM may be enabled or disabled by default depending on implementation specific criteria and/or the requirements of the associated form factor specification(s). Software can enable or disable ASPM using a process described in Section 5.4.1.3.1.

Power management software enables (or disables) ASPM in each Port of a component by programming the ASPM Control field. Note that new BIOS code can effectively enable or disable ASPM functionality even when running with a legacy operating system.

IMPLEMENTATION NOTE

Isochronous Traffic and ASPM

Isochronous traffic requires bounded service latency. ASPM may add latency to isochronous transactions beyond expected limits. A possible solution would be to disable ASPM for devices that are configured with an Isochronous Virtual Channel.

Multi-function Endpoints may be programmed with different values in their respective Active_PM_En registers of each function. The policy for such a component will be dictated by the most active common denominator among all D0 functions according to the following rules:

❑ Functions in non-D0 state (D1 and deeper) are ignored in determining the ASPM policy

❑ If any of the functions in D0 state has its ASPM disabled, (ASPM Control field = 00b), then ASPM is disabled for the entire component

❑ Else, if at least one of the functions in D0 state is enabled for L0s only (ASPM Control field = 01b), then ASPM is enabled for L0s only

❑ Else, ASPM is enabled for both L0s and L1 states

Note that the components must be capable of changing their behavior during runtime as devices enter and exit low power device states. For example, if one function within a multi-function component is programmed to disable ASPM, then ASPM must be disabled for that component while that function is in the D0 state. Once the function transitions to a non-D0 state, ASPM can be enabled to at least the L0s state if all other functions are enabled for ASPM.

5.4.1.1. L0s ASPM State

All PCI Express devices must support the L0s low power Link state.

Transaction Layer and Link Layer timers are not affected by a transition to the L0s state (i.e., they must follow the rules as defined in their respective chapters).

IMPLEMENTATION NOTE

Minimizing L0s Exit Latency

L0s exit latency depends mainly on the ability of the Receiver to quickly acquire bit and Symbol synchronization. Different approaches exist for high-frequency clocking solutions which may differ significantly in their L0s exit latency, and therefore in the efficiency of ASPM. To achieve maximum power savings efficiency with ASPM, L0s exit latency should be kept low by proper selection of the clocking solution.

5.4.1.1.1. Entry to the L0s State

Entry into the L0s state is managed separately for each direction of the Link. It is the responsibility of each device at either end of the Link to initiate an entry into the L0s state on its transmitting Lanes.

5   A Port that is disabled for the L0s state must not transition its transmitting Lanes to the L0s state. It must still however be able to tolerate having its Receiver Port Lanes entering L0s, (as a result of the device at the other end bringing its transmitting Lanes into L0s state), and then later returning to the L0 state.

L0s Invocation Policy

10   PCI Express Ports that are enabled for L0s entry must transition their Transmit Lanes to the L0s state if the defined idle conditions (below) are met for a period of time not to exceed 7 µs. Within this time period, the policy used by the Port to determine when to enter L0s is implementation specific.

Definition of Idle

15   The definition of "idle" varies with device category.

An Endpoint Port, Root Port is determined to be idle if the following conditions are met:

❑ No TLP is pending to transmit over the Link, or no FC credits are available to transmit any TLPs

❑ No DLLPs are pending for transmission

20   A Switch's Upstream Port is determined to be idle if the following conditions are met:

❑ All of the Switch's Downstream Port Receive Lanes are in the L0s state

❑ No pending TLPs to transmit, or no FC credits are available to transmit anything

❑ No DLLPs are pending for transmission

A Switch's Downstream Port is determined to be idle if the following conditions are met:

25   ❑ The Switch's Upstream Port's Receive Lanes are in the L0s state

❑ No pending TLPs to transmit on this Link, or no FC credits are available

❑ No DLLPs are pending for transmission

See Section 4.2 for details on L0s entry by the Physical Layer.

5.4.1.1.2. Exit from L0s State

30   A component with its Transmitter in L0s must initiate L0s exit when it has a TLP or DLLP to transmit across the Link. Note that a transition from the L0s Link state does not depend on the status (or availability) of FC credits. The Link must be able to reach the L0 state, and to exchange FC credits across the Link. For example, if all credits of some type were consumed when the Link entered L0s, then any component on either side of the Link must still be able to transition the Link

to the L0 state when new credits need to be sent across the Link. Note that it may be appropriate for a component to anticipate the end of the idle condition and initiate L0s transmit exit; for example, when a NP request is received.

Downstream Initiated Exit

5   An Endpoint or Switch is permitted to initiate an exit from the L0s low power state on its Transmit Link, (Upstream Port Transmit Lanes in the case of a Downstream Switch), if it needs to communicate through the Link. The component initiates a transition to the L0 state on Lanes in the upstream direction as described in Section 4.2.

If the Upstream component is a Switch (i.e., it is not the Root Complex), then it must initiate a
10   transition on its Upstream Port Transmit Lanes (if the Upstream Port's Transmit Lanes are in a low power state) as soon as it detects an exit from L0s on any of its Downstream Ports.

Upstream Initiated Exit

The Root Complex or Switch (Downstream Port) is permitted to initiate an exit from L0s low power state on any of its Transmit Links if it needs to communicate through the Link. The
15   component initiates a transition to the L0 state on Lanes in the downstream direction as described in Chapter 4.

If the Downstream component is a Switch (i.e., it is not an Endpoint), it must initiate a transition on all of its Downstream Port Transmit Lanes that are in L0s at that time as soon as it detects an exit from L0s on its Upstream Port. Links that are already in the L0 state are not affected by this
20   transition. Links whose Downstream component is in a low power state (i.e., D1-D3$_{hot}$ states) are also not affected by the exit transitions.

For example, consider a Switch with an Upstream Port in L0s and a Downstream device in a D1 state. A configuration request packet travels downstream to the Switch, intending to ultimately reprogram the Downstream device from D1 to D0. The Switch's Upstream Port Link must
25   transition to the L0 state to allow the packet to reach the Switch. The Downstream Link connecting to the device in D1 state will not transition to the L0 state yet; it will remain in the L1 state. The captured packet is checked and routed to the Downstream Port that shares a Link with the Downstream device that is in D1. As described in Section 4.2, the Switch now transitions the Downstream Link to the L0 state. Note that the transition to the L0 state was triggered by the
30   packet being routed to that particular Downstream L1 Link, and not by the transition of the Upstream Port's Link to the L0 state. If the packet's destination was targeting a different Downstream Link, then that particular Downstream Link would have remained in the L1 state.

5.4.1.2.   L1 ASPM State

A component may optionally support the ASPM L1 state; a state that provides greater power
35   savings at the expense of longer exit latency. L1 exit latency is visible to software, and reported via the Link Capabilities register defined in Section 7.8.6.

When supported, L1 entry is disabled by default in the ASPM Control configuration field.

5.4.1.2.1. Entry to L1 State

An Endpoint enabled for L1 ASPM entry may initiate entry into the L1 Link state.

🗒 IMPLEMENTATION NOTE

Initiating L1

5 This specification does not dictate when an Endpoint must initiate a transition to the L1 state on its Transmit Lanes. The interoperable mechanisms for transitioning into and out of L1 are defined within this specification, however the specific ASPM policy governing when to transition into L1 is left to the implementer.

One possible approach would be for the Downstream device to initiate a transition to the L1 state
10 once the device has both its Receiver and Transmitter in the L0s state (RxL0s and TxL0s) for a set amount of time.

Three power management Messages provide support for ASPM of the L1 state:

❑ PM_ Active_State_Request_L1   (DLLP)

❑ PM_ Request_Ack   (DLLP)

15 ❑ PM_Active_State_Nak   (TLP)

Endpoints that have their ASPM L1 entry enabled negotiate for the resultant L-state with the component on the upstream end of the Link. If the Endpoint receives a negative acknowledgement in response to its issuance of a PM_Active_State_Request_L1 DLLP, the Endpoint must enter the L0s state as soon as possible.[54] Note that the component on the upstream side of the Link may not
20 support L1 ASPM, or it may be disabled and so could legitimately respond to the L1 entry request with a negative acknowledgement.

A Root Complex Root Port, or Switch Downstream Port must accept a request to enter a low power L1 state if all of the following conditions are true:

❑ The Port supports ASPM L1 entry, and ASPM L1 entry is enabled.

25 ❑ No TLP is scheduled for transmission

❑ No Ack or Nak DLLP is scheduled for transmission

---

[54] Assuming L0s is enabled and that the conditions for L0s entry are met.

A Switch Upstream Port may request L1 entry on its Link provided all of the following conditions are true:

❑ The Upstream Port supports ASPM L1 entry and it is enabled

❑ All of the Switch's Downstream Port Links are in the L1 state (or deeper)

❑ No pending TLPs to transmit

❑ No pending DLLPs to transmit

❑ The Upstream Port's Receive Lanes are idle for an implementation specific set amount of time

If the Switch's Upstream Port receives a negative acknowledgement in response to its issuance of a PM_Active_State_Request_L1 DLLP, then the Switch's Upstream Port Transmit Lanes must instead transition to the L0s state as soon as possible.[55]

Note that it is legitimate for a Switch to be enabled for the ASPM L1 Link state on any of its Downstream Ports and to be disabled or not even supportive of ASPM L1 on its Upstream Port. In that case, Downstream Ports may enter the L1 Link state, but the Switch will never initiate an ASPM L1 entry transition on its Upstream Port.

ASPM L1 Negotiation Rules (see Figure 5-6 and Figure 5-7)

❑ Upon deciding to enter a low power Link state, the Downstream component must block scheduling of any TLPs (including completion packets).

❑ The Downstream component must wait until it receives a Link Layer acknowledgement for the last TLP it had previously sent (i.e., the retry buffer is empty). The component may retransmit a TLP if required by the Data Link Layer rules.

❑ The Downstream component must also wait until it accumulates at least the minimum number of credits required to send the largest possible packet for any FC type. Note that this is required so that the component can immediately issue a TLP after it exits the L1 state.

❑ The Downstream component then initiates ASPM negotiation by sending a PM_Active_State_Request_L1 DLLP onto its Transmit Lanes. The Downstream component sends this DLLP continuously until it receives a response from the Upstream device (see below). The Downstream component remains in this loop waiting for a response from the Upstream Agent.

• During this waiting period, the Downstream component must not initiate any Transaction Layer transfers. It must still accept TLPs and DLLPs from the Upstream component. It also responds with DLLPs as needed by the Link Layer protocol. All received TLPs must be acknowledged before entering the L1 state (i.e., DLLP ACK/NAK Latency timers are reset before entering the L1 state).

• If the Downstream component for any reason needs to initiate a transfer on the Link, it must first complete the transition to the low power Link state. Once in a lower power Link state, the Downstream component is then permitted to exit the low power Link state to handle the transfer.

---

[55] Assuming L0s is enabled and that the conditions for L0s entry are met.

❑ The Upstream component must immediately respond to the request with either an acceptance or a rejection of the request.

Rules in case of rejection:

❑ In the case of a rejection, the Upstream component must schedule, as soon as possible, a rejection (Nak) by sending the PM_Active_State_Nak Message to the Downstream requesting agent. Once the PM_Active_State_Nak Message is sent, the Upstream component is permitted to initiate any TLP or DLLP transfers.

❑ If the request was rejected, the Downstream component must immediately transition its Transmit Lanes into the L0s state, provided L0s is enabled and that conditions for L0s entry are met.

Rules in case of acceptance:

❑ If the Upstream agent is ready to accept the request, it must block scheduling of any TLPs.

❑ The Upstream component then must wait until it receives a Link Layer acknowledgement for the last TLP it had previously sent. The Upstream component may retransmit a TLP if required by the Link Layer rules.

❑ The Upstream component must also wait until it accumulates at least the minimum number of credits required to send the largest possible packet for any FC type. Note that this is required so that the component can immediately issue a TLP after it exits the L1 state.

❑ Once all TLPs have been acknowledged and enough FC credits accumulated, the Upstream component sends a PM_Request_Ack DLLP downstream. The Upstream component sends this DLLP continuously until it observes its Receive Lanes enter into the Electrical Idle state. See Chapter 4 for more details on the Physical Layer behavior.

❑ If the Upstream component needs, for any reason, to initiate a transfer on the Link after it sends a PM_Request_Ack DLLP, it must first complete the transition to the low power state. It is then permitted to exit the low power state to handle the transfer once the Link is back to L0.

❑ When the Downstream component detects a PM_Request_Ack DLLP on its Receive Lanes (signaling that the Upstream device acknowledged the transition to L1 request), the Downstream component then ceases sending the PM_Active_State_Request_L1 DLLP, disables DLLP, TLP transmission and brings its Transmit Lanes into the Electrical Idle state.

❑ When the Upstream component detects an Electrical Idle on its Receive Lanes (signaling that the Downstream component has entered the L1 state), it then ceases to send the PM_Request_Ack DLLP, disables DLLP, TLP transmission and brings the downstream direction of the Link into the Electrical Idle state.

Notes:

1. The transaction Layer Completion Timeout mechanism is not affected by transition to the L1 state (i.e., it must keep counting).

2. Flow Control Update timers are frozen while the Link is in L1 state to prevent a timer expiration that will unnecessarily transition the Link back to the L0 state.
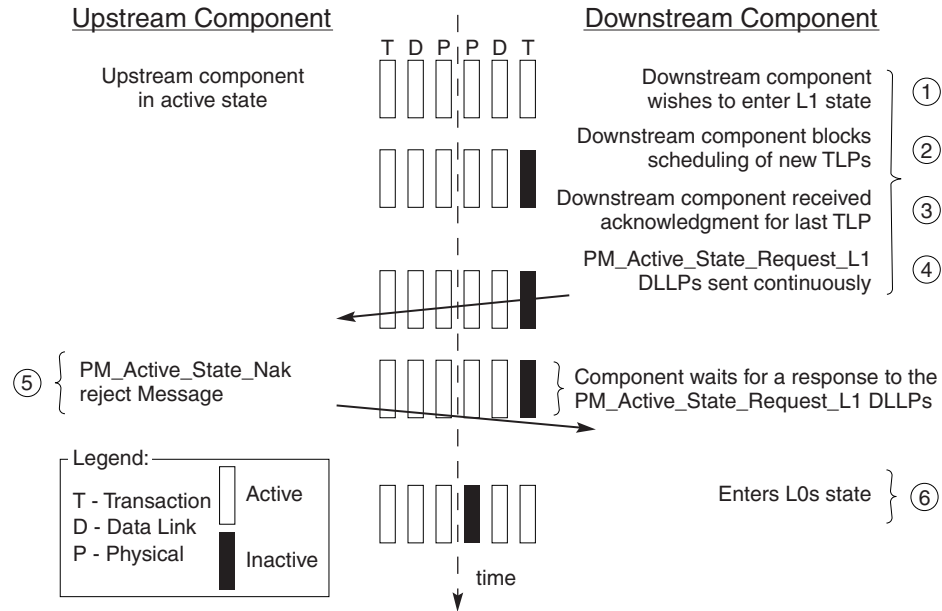
Figure 5-6: L1 Transition Sequence Ending with a Rejection (L0s Enabled)
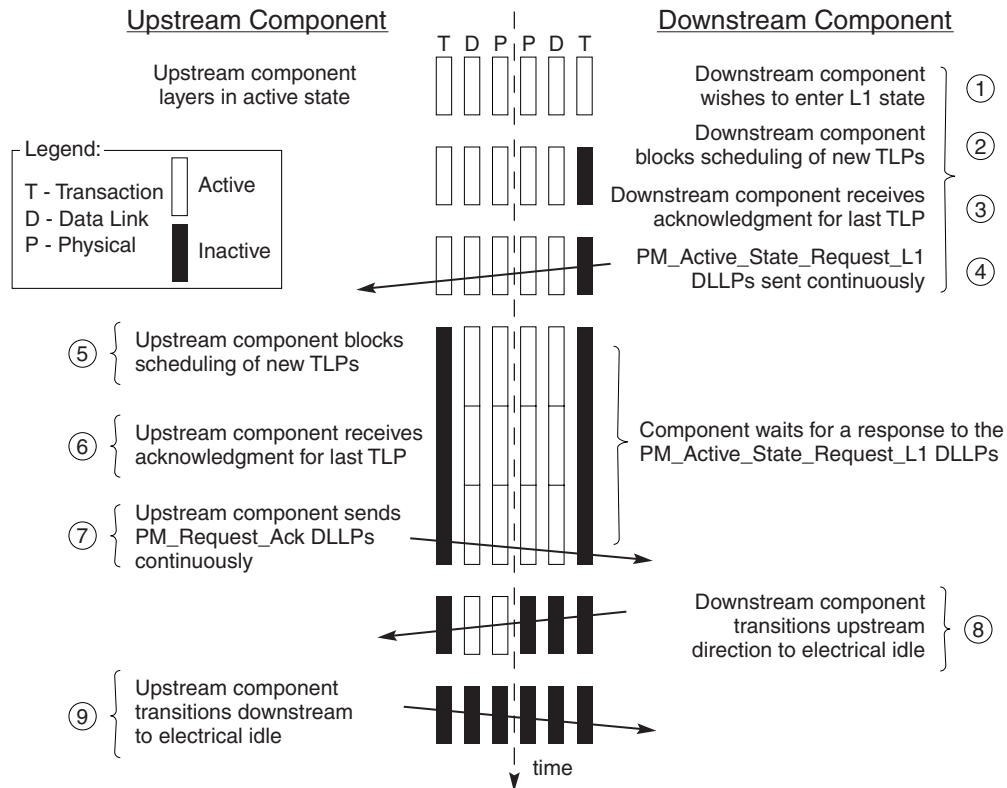
Figure 5-7: L1 Successful Transition Sequence

5.4.1.2.2. Exit from L1 State

Components on either end of a PCI Express Link may initiate an exit from the L1 Link state. Unlike the L1 entry protocol, where both ends negotiate for the resultant Link state, L1 exit does not require any negotiation.

5 *Downstream Initiated Exit*

An Endpoint or Switch Upstream Port is permitted to initiate an exit from L1 on its Transmit Lanes if it needs to communicate through the Link. The component initiates a transition to the L0 state as described in Chapter 4. The Upstream component must respond by initiating a similar transition of its Transmit Lanes.

10 If the Upstream component is a Switch Downstream Port, (i.e., it is not a Root Complex Root Port), the Switch must initiate an L1 exit transition on its Upstream Port's Transmit Lanes, (if the Upstream Port's Link is in the L1 state), as soon as it detects the L1 exit activity on any of its Downstream Port Links. Since L1 exit latencies are relatively long, a Switch must not wait until its Downstream Port Link has fully exited to L0 before initiating an L1 exit transition on its Upstream
15 Port Link. Waiting until the Downstream Link has completed the L0 transition will cause a Message traveling through several PCI Express Switches to experience accumulating latency as it traverses each Switch.

A Switch is required to initiate an L1 exit transition on its Upstream Port Link after no more than 1 μs from the beginning of an L1 exit transition on any of its Downstream Port Links. Refer to
20 Section 4.2 for details of the Physical Layer signaling during L1 exit.

Consider the example in Figure 5-8. The numbers attached to each Port represent the corresponding Port's reported Transmit Lanes L1 exit latency in units of microseconds.

Links 1, 2, and 3 are all in the L1 state, and Endpoint C initiates a transition to the L0 state at time T. Since Switch B takes 32 μs to exit L1 on its Ports, Link 3 will transition to the L0 state at T+32
25 (longest time considering T+8 for the Endpoint C, and T+32 for Switch B).

Switch B is required to initiate a transition from the L1 state on its Upstream Port Link (Link 2) after no more than 1 μs from the beginning of the transition from the L1 state on Link 3. Therefore, transition to the L0 state will begin on Link 2 at T+1. Similarly, Link 1 will start its transition to the L0 state at time T+2.

30 Following along as above, Link 2 will complete its transition to the L0 state at time T+33 (since Switch B takes longer to transition and it started at time T+1). Link 1 will complete its transition to the L0 state at time T+34 (since the Root Complex takes 32 μs to transition and it started at time T+2).

Therefore, among Links 1, 2, and 3, the Link to complete the transition to the L0 state last is Link 1
35 with a 34 μs delay. This is the delay experienced by the packet that initiated the transition in Endpoint C.
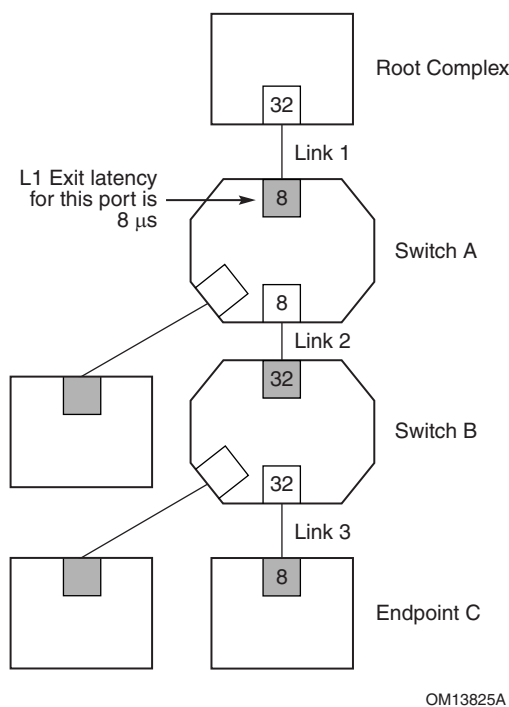
OM13825A

Figure 5-8: Example of L1 Exit Latency Computation

Switches are not required to initiate an L1 exit transition on any other of their Downstream Port Links.

Upstream Initiated Exit

A Root Complex, or a Switch is permitted to initiate an exit from L1 on any of its Root Ports, or Downstream Port Links if it needs to communicate through that Link. The component initiates a transition to the L0 state as described in Chapter 4. The Downstream component must respond by initiating a similar transition on its Transmit Lanes.

If the Downstream component is a Switch (i.e., it is not an Endpoint), it must initiate a transition on all of its Downstream Links (assuming the Downstream Link is in an ASPM L1 state) as soon as it detects an exit from L1 state on its Upstream Port Link. Since L1 exit latencies are relatively long, a Switch must not wait until its Upstream Port Link has fully exited to L0 before initiating an L1 exit transition on its Downstream Port Links. If that were the case, a Message traveling through multiple PCI Express Switches would experience accumulating latency as it traverses each Switch.

A Switch is required to initiate a transition from L1 state on all of its Downstream Port Links that are currently in L1 after no more than 1 μs from the beginning of a transition from L1 state on its Upstream Port. Refer to Section 4.2 for details of the Physical Layer signaling during L1 exit. Downstream Port Links that are already in the L0 state do not participate in the exit transition. Downstream Port Links whose Downstream component is in a low power D-state (D1-D3$_{hot}$) are also not affected by the L1 exit transitions (i.e., such Links must not be transitioned to the L0 state).

5.4.1.3. ASPM Configuration

All PCI Express functions must implement the following configuration bits in support of ASPM. Refer to Chapter 7 for configuration register assignment and access mechanisms.

Each PCI Express component reports its level of support for ASPM in the ASPM Support configuration field below. All PCI Express components must support transition to the L0s Link state. Support for transition to the L1 Link state while in $D0_{active}$ state is optional.

Table 5-3: Encoding of the ASPM Support Field

Field	Description
ASPM Support	00b – Reserved
	01b – L0s supported
	10b – Reserved
	11b – L0s and L1 supported

Each PCI Express component reports the source of its reference clock in its Slot Clock Configuration bit located in its PCI Express Capability structure's Link Status register.

Table 5-4: Description of the Slot Clock Configuration Field

Field	Description
Slot Clock Configuration	This bit indicates that the component uses the same physical reference clock that the platform provides on the connector. If the device uses an independent clock irrespective of the presence of a reference on the connector, this bit must be clear. For Root and Switch Downstream Ports, this bit, when set, indicates that the Downstream Port is using the same reference clock as the Downstream device or the slot. For Switch and Bridge Upstream Ports, this bit when set, indicates that the Upstream Port is using the same reference clock that the platform provides. Otherwise it is clear.

Each PCI Express component must support the Common Clock Configuration bit in their PCI Express Capability structure's Link Status register. Software writes to this register bit to indicate to the device whether it is sharing the same clock source as the device on the other end of the Link.

Table 5-5: Description of the Common Clock Configuration Field

Field	Description
Common Clock Configuration	This bit when set indicates that this component and the component at the opposite end of the Link are operating with a common clock source. A value of 0 indicates that this component and the component at the opposite end of the Link are operating with separate reference clock sources. Default value of this field is 0. Components utilize this common clock configuration information to report the correct L0s and L1 Exit Latencies.

Each PCI Express Port reports the L0s and L1 exit latency (the time that they require to transition their Receive Lanes from the L0s or L1 state to the L0 state) in the L0s Exit Latency and the L1 Exit Latency configuration fields, respectively.

If L1 state is not supported for ASPM (as reported in the ASPM Support field), the L1 Exit latency field is ignored.

Table 5-6: Encoding of the L0s Exit Latency Field

Field	Description
L0s Exit Latency	000b – Less than 64 ns
	001b – 64 ns to less than 128 ns
	010b – 128 ns to less than 256 ns
	011b – 256 ns to less than 512 ns
	100b – 512 ns to less than 1 μs
	101b – 1 μs to less than 2 μs
	110b – 2 μs to 4 μs
	111b – More than 4 μs

Table 5-7: Encoding of the L1 Exit Latency Field

Field	Description
L1 Exit Latency	000b – Less than 1 μs
	001b – 1 μs to less than 2 μs
	010b – 2 μs to less than 4 μs
	011b – 4 μs to less than 8 μs
	100b – 8 μs to less than 16 μs
	101b – 16 μs to less than 32 μs
	110b – 32 μs to 64 μs
	111b – More than 64 μs

Endpoints also report the additional latency that they can absorb due to the transition from L0s state or L1 state to the L0 state. This is reported in the Endpoint L0s Acceptable Latency and Endpoint L1 Acceptable Latency fields, respectively.

Power management software, using the latency information reported by all components in the PCI Express Hierarchy, can enable the appropriate level of ASPM by comparing exit latency for each given path from root to Endpoint against the acceptable latency that each corresponding Endpoint can withstand.

Table 5-8: Encoding of the Endpoint L0s Acceptable Latency Field

Field	Description
Endpoint L0s Acceptable Latency	000b – Less than 64 ns
	001b – 64 ns to less than 128 ns
	010b – 128 ns to less than 256 ns
	011b – 256 ns to less than 512 ns
	100b – 512 ns to less than 1 µs
	101b – 1 µs to less than 2 µs
	110b – 2 µs to less than 4 µs
	111b – More than 4 µs

Table 5-9: Encoding of the Endpoint L1 Acceptable Latency Field

Field	Description
Endpoint L1 Acceptable Latency	000b – Less than 1 µs
	001b – 1 µs to less than 2 µs
	010b – 2 µs to less than 4 µs
	011b – 4 µs to less than 8 µs
	100b – 8 µs to less than 16 µs
	101b – 16 µs to less than 32 µs
	110b – 32 µs to 64 µs
	111b – More than 64 µs

Power management software enables (or disables) ASPM in each component by programming the ASPM Control field.

Table 5-10: Encoding of the ASPM Control Field

Field	Description
ASPM Control	00b – Disabled
	01b – L0s Entry Enabled
	10b – L1 Entry Enabled
	11b – L0s and L1 Entry enabled

ASPM Control = 00

Port must not bring a Link into L0s state.

5 Ports connected to the Downstream end of the Link must not issue a PM_Active_State_Request_L1 DLLP on its Upstream Link.

Ports connected to the Upstream end of the Link receiving a L1 request must respond with negative acknowledgement.

ASPM Control = 01

Port must bring a Link into L0s state if all conditions are met.

Ports connected to the Downstream end of the Link must not issue a PM_Active_State_Request_L1 DLLP on its Upstream Link.

⁵ Ports connected to the Upstream end of the Link receiving a L1 request must respond with negative acknowledgement.

ASPM Control = 10

Port's Transmitter must not enter L0s

Ports connected to the Downstream end of the Link may issue PM_Active_State_Request_L1
¹⁰ DLLPs.

Ports connected to the Upstream end of the Link must respond with positive acknowledgement to a L1 request and transition into L1 if the conditions for Root Complex Root Port or Switch Downstream Port in Section 5.4.1.2.1 are met.

ASPM Control = 11

¹⁵ Port must bring a Link into the L0s state if all conditions are met.

Ports connected to the Downstream end of the Link may issue PM_Active_State_Request_L1 DLLPs.

Ports connected to the Upstream end of the Link must respond with positive acknowledgement to a L1 request and transition into L1 if the conditions for Root Complex Root Port or Switch
²⁰ Downstream Port in Section 5.4.1.2.1 are met.

5.4.1.3.1. Software Flow for Enabling or Disabling ASPM

Following is an example software algorithm that highlights how to enable or disable ASPM in a PCI Express component.

❑ PCI Express components power up with an appropriate value in their Slot Clock Configuration
²⁵ bit. The method by which they initialize this bit is device-specific

❑ PCI Express system software scans the Slot Clock Configuration bit in the components on both ends of each Link to determine if both are using the same reference clock source or reference clocks from separate sources. If the Slot Clock Configuration bits in both devices are set, they are both using the same reference clock source, otherwise they're not.

³⁰ ❑ PCI Express software updates the Common Clock Configuration bits in the components on both ends of each Link to indicate if those devices share the same reference clock and triggers Link retraining by writing 1 to the Retrain Link bit in the Link Control register of one of the components.

❑ Devices must reflect the appropriate L0s/L1 exit latency in their L0s/L1 Exit Latency register
³⁵ bits, per the setting of the Common Clock Configuration bit.

❑ PCI Express system software then reads and calculates the L0s/L1 exit latency for each Endpoint based on the latencies reported by each Port. See Section 5.4.1.2.2 for an example.

❑ For each Endpoint component, PCI Express system software examines the Endpoint L0s/L1 Acceptable Latency, as reported by the Endpoint component in their Link Capabilities register, and enables or disables L0s/L1 entry (via the ASPM Control bits in the Link Control register) accordingly in some or all of the intervening device Ports on that hierarchy.

5.5. Auxiliary Power Support

5.5.1. Auxiliary Power Enabling

The PCI-PM specification requires that a function must support PME generation in order to consume the maximum allowance of auxiliary current (375 mA vs. 20 mA). However, there are instances where functions need to consume power even if they are "PME Disabled," or PME incapable by design. One example is a component with its system management mode active during a system low power state.

PCI Express PM provides a new control bit in the Device Control register, Aux Power PM Enable, that provides the means for enabling a function to draw the maximum allowance of auxiliary current independent of its level of support for PME generation.

A PCI Express function requests aux power allocation by specifying a non-zero value in the Aux_Current field of the Power Management Capabilities register (PMC). Refer to Chapter 7 for the Aux Power PM Enable register bit assignment, and access mechanism.

Legacy PCI-PM software is unaware of this new bit and will only be able to enable aux current to a given function based on the function's reported PME support, the Aux_Current field value and the function's PME_Enable bit.

Allocation of aux power using Aux Power PM Enable is determined as follows:

Aux Power PM Enable = 1b:

Aux power is allocated as requested in the Aux_Current field of the Power Management Capabilities register (PMC), independent of the PME_En bit in the Power Management Control/Status register (PMCSR). The PME_En bit still controls the ability to master PME.

Aux Power PM Enable = 0b:

Aux power allocation is controlled by the PME_En bit as defined in the PCI-PM specification.

The Aux Power PM Enable bit is sticky (see Section 7.4) so its state is preserved in the $D3_{cold}$ state, and is not affected by the transitions from the $D3_{cold}$ state to the $D0_{Uninitlaized}$ state.

Typically, aux power is required to support the Beacon wakeup mechanism, and components supporting Beacon must not consume aux power unless enabled by software to do so. To enable finer granularity of power consumption, Root Ports and Switch Ports should support independent control of aux power consumption for each Port. However, it is permitted to logically OR the aux power enables for multiple ports and combine the aux power control for those ports so long as the combined aux power consumption does not exceed the sum of the amounts enabled by software.

5.6. Power Management System Messages and DLLPs

Table 5-11 defines the location of each PM packet in the PCI Express stack.

Table 5-11: Power Management System Messages and DLLPs

Packet	Type
PM_Enter_L1	DLLP
PM_Enter_L23	DLLP
PM_Active_State_Request_L1	DLLP
PM_Request_Ack	DLLP
PM_Active_State_Nak	Transaction Layer Message
PM_PME	Transaction Layer Message
PME_Turn_Off	Transaction Layer Message
PME_TO_Ack	Transaction Layer Message

For information on the structure of the power management DLLPs, refer to Section 3.4.

Power management Messages follow the general rules for PCI Express system Messages. Message fields follow the following rules:

❑ Length field is reserved.

❑ Attribute field must be set to the default values (all 0's).

❑ Address field is reserved.

❑ Requester ID

- PM_PME Message

    ♦ Endpoints report their Upstream Link bus number and the device and function number where the PME originated.

- All other devices report their Upstream Link bus and device numbers, and the function number must be zero.

❑ Traffic Class field must use the default class (TC0)

6

6.   System Architecture

This chapter addresses various aspects of PCI Express interconnect architecture in a platform context. It covers the details of interrupt support, error signaling and logging, Virtual Channel and isochronous support, device synchronization, lock, reset and Hot-Plug.

6.1.   Interrupt and PME Support

The PCI Express interrupt model supports two mechanisms:

❑ INTx emulation

❑ Message Signaled Interrupt (MSI) Support.

For legacy compatibility, PCI Express provides a PCI INTx emulation mechanism to signal interrupts to the system interrupt controller (typically part of the Root Complex). This mechanism is compatible with existing PCI software, and provides the same level and type of service as corresponding PCI interrupt signaling mechanism and is independent of system interrupt controller specifics. This legacy compatibility mechanism allows boot device support without requiring complex BIOS-level interrupt configuration/control service stacks. It virtualizes PCI physical interrupt signals by using an in-band signaling mechanism.

In addition to PCI INTx compatible interrupt emulation, PCI Express requires support of the Message Signaled Interrupt (MSI) mechanism. The PCI Express MSI mechanism is compatible with the MSI capability defined in the PCI 2.3 Specification.

6.1.1.   Rationale for PCI Express Interrupt Model

PCI Express takes an evolutionary approach from PCI with respect to interrupt support.

As required for PCI/PCI-X interrupt mechanisms, each device is required to differentiate between INTx (legacy) and MSI (native) modes of operation. The PCI Express device complexity required to support both schemes is no different than that for PCI/PCI-X devices today. The advantages of this approach include:

❑ Compatibility with existing PCI Software Models

❑ Direct support for boot devices

❑ Easier End of Life (EOL) for INTx legacy mechanisms.

Existing software model is used to differentiate legacy (INTx) vs. MSI modes of operation; thus, no special software support is required for PCI Express.

6.1.2. PCI Compatible INTx Emulation

PCI Express supports the PCI interrupts as defined in the PCI Specification, rev. 2.3 including the Interrupt Pin and Interrupt Line registers of the PCI configuration space for PCI devices. PCI Express devices support these registers for backwards compatibility; actual interrupt signaling uses in-band Messages rather than being signaled using physical pins.

Two Messages are defined, Assert_INTx and Deassert_INTx, for emulation of PCI INTx signaling, where x is A, B, C, and D for respective PCI interrupt signals. These Messages are used to provide "virtual wires" for signaling interrupts across a Link. Switches collect these virtual wires and present a combined set at the Switch's Upstream Port. Ultimately, the virtual wires are routed to the Root Complex which maps the virtual wires to system interrupt resources. PCI Express devices must use assert/de-assert Messages in pairs to emulate PCI interrupt level-triggered signaling. Actual mapping of PCI Express INTx emulation to system interrupts is implementation specific as is mapping of physical interrupt signals in PCI today.

The legacy INTx emulation mechanism may be depreciated in a future version of this specification.

6.1.3. INTx Emulation Software Model

The software model for legacy INTx emulation matches that of PCI. The system BIOS reporting of chipset/platform interrupt mapping and the association of a device's interrupt with PCI interrupt lines is handled in exactly the same manner as with previous PCI systems. Legacy software reads from the device's Interrupt Pin register to determine if the device is interrupt driven. A value between 01 and 04 indicates that the device uses interrupt pin to generate an interrupt.

Note that similarly to physical interrupt signals, the INTx emulation mechanism may potentially cause spurious interrupts that must be handled by the system software.

6.1.4. Message Signaled Interrupt (MSI) Support

The Message Signaled Interrupt (MSI) capability is defined in the PCI 2.3 Specification.

MSI interrupt support, which is optional for PCI 2.3 devices, is required for PCI Express devices. MSI-capable devices deliver interrupts by performing memory write transactions. MSI is an edge-triggered interrupt; neither the PCI 2.3 Specification nor this specification support level-triggered MSI interrupts. Certain PCI devices and their drivers rely on INTx-type level-triggered interrupt behavior (addressed by the PCI Express legacy INTx emulation mechanism). To take advantage of the MSI capability and edge-triggered interrupt semantics, these devices and their drivers may have to be redesigned.

A legacy Endpoint is required to support either the 32-bit or 64-bit Message Address version of the MSI capability structure. A PCI Express Endpoint is required to support the 64-bit Message Address version of the MSI capability structure.

The requestor of an MSI transaction must set the No Snoop and Relaxed Ordering attributes of the Transaction Descriptor to 0.

Note that, unlike INTx emulation Messages, MSIs are not restricted to TC0 traffic class.

📝 IMPLEMENTATION NOTE

Synchronization of Data Traffic and Message Signaled Interrupts

MSIs are permitted to use the TC that is most appropriate for the device's programming model. This is generally the same TC as is used to transfer data; for legacy I/O, TC0 should be used.

If a device uses more than one TC, it must explicitly ensure that proper synchronization is maintained between data traffic and interrupt Message(s) not using the same TC. Methods for ensuring this synchronization are implementation specific. One option is for a device to issue a zero length Read (as described in Section 2.2.5) using each additional TC used for data traffic prior to issuing the MSI. Other methods are also possible. Note, however, that platform software (e.g., a device driver) is generally only capable of issuing transactions using TC0.

6.1.5. PME Support

PCI Express supports power management events from native PCI Express devices as well as PME-capable PCI devices.

PME signaling is accomplished using an in-band Transaction Layer PME Message (PM_PME) as described in Chapter 5.

6.1.6. Native PME Software Model

PCI Express-aware software can enable a mode where the Root Complex signals PME via an interrupt. When configured for native PME support, a Root Port receives the PME Message and sets the PME Status bit in its Root Status register. If software has set the PME Interrupt Enable bit in the Root Control register to 1b, the Root Port then generates an interrupt.

The software handler for this interrupt can determine which device sent the PME Message by reading the PME Requester ID field in the Root Status register in a Root Port. It dismisses the interrupt by writing a 1b to the PME Status bit in the Root Status register. See Section 7.8.13 for more details.

6.1.7. Legacy PME Software Model

Legacy software, however, will not understand this mechanism for signaling PME. In the presence of legacy system software, the system power management logic in the Root Complex receives the PME Message and informs system software through an implementation specific mechanism. The Root Complex may utilize the PCI Express Requestor ID in the PM_PME to inform system software which device caused the power management event.

Because it is delivered by a Message, PME has edge-triggered semantics in PCI Express, which differs from the level-triggered PME mechanism used for PCI. It is the responsibility of the Root

Complex to abstract this difference from system software to maintain compatibility with PCI systems.

6.1.8.   Operating System Power Management Notification

In order to maintain compatibility with non-PCI Express-aware system software, system power
management logic must be configured by firmware to use the legacy mechanism of signaling PME
by default. PCI Express-aware system software must notify the firmware prior to enabling native,
interrupt-based PME signaling. In response to this notification, system firmware must, if needed,
reconfigure the Root Complex to disable legacy mechanisms of signaling PME. The details of this
firmware notification are beyond the scope of this specification, but since it will be executed at
system run-time, the response to this notification must not interfere with system software.
Therefore, the firmware must not write to system available memory or I/O, or the configuration
space headers of any PCI device.

6.1.9.   PME Routing Between PCI Express and PCI Hierarchies

PME-capable conventional PCI and PCI-X devices assert the PME# pin to signal a power
management event. The PME# signal from PCI devices may either be converted to a PCI Express
in-band PME Message by a PCI Express-PCI Bridge or routed directly to the Root Complex.

If the PME# signal from a PCI device is routed directly to the Root Complex, it signals system
software using the same mechanism used in present PCI systems. A Root Complex may optionally
provide support for signaling PME from PCI devices to system software via an interrupt. In this
scenario, it is recommended for the Root Complex to detect the bus, device and function number of
the PCI device that asserted PME#, and use this information to fill in the PME Requester ID field
in the Root Port that originated the hierarchy containing the PCI device. If this is not possible, the
Root Complex may optionally write the Requester ID of the Root Port to this field.

6.2.　Error Signaling and Logging

In this document, errors which must be checked and errors which may optionally be checked are identified. Each such error is associated either with the Port or with a specific device (or function in a multi-function device), and this association is given along with the description of the error. This
5　section will discuss how errors are classified and reported.

6.2.1.　Scope

This section explains the error signaling and logging requirements for PCI Express components. This includes errors which occur on the PCI Express interface itself and those errors which occur on behalf of transactions initiated on PCI Express. This section does not focus on errors which
10　occur within the component that are unrelated to a particular PCI Express transaction. This type of error signaling is better handled through proprietary methods employing device-specific interrupts.

PCI Express defines two error reporting paradigms: the baseline capability and the Advanced Error Reporting capability. The baseline error reporting capabilities are required of all PCI Express devices and define the minimum error reporting requirements. The Advanced Error Reporting
15　capability is defined for more robust error reporting and is implemented with a specific PCI Express capability structure (refer to Chapter 7 for a definition of this optional capability). This section explicitly calls out all error handling differences between the baseline and the Advanced Error Reporting capability.

All PCI Express devices support existing, non-PCI Express-aware, software for error handling by
20　mapping PCI Express errors to existing PCI reporting mechanisms, in addition to the PCI Express-specific mechanisms.

6.2.2.　Error Classification

PCI Express errors can be classified as two types: Uncorrectable errors and Correctable errors. This classification separates those errors resulting in functional failure from those errors resulting in
25　degraded performance. Uncorrectable errors can further be classified as Fatal or Non-Fatal (see Figure 6-1).

ERR_COR   ERR_NONFATAL   ERR_FATAL

Physical

Physical

Fatal

Data Link   Transaction   Data Link   Transaction

Correctable Errors   Uncorrectable Errors

OM13827
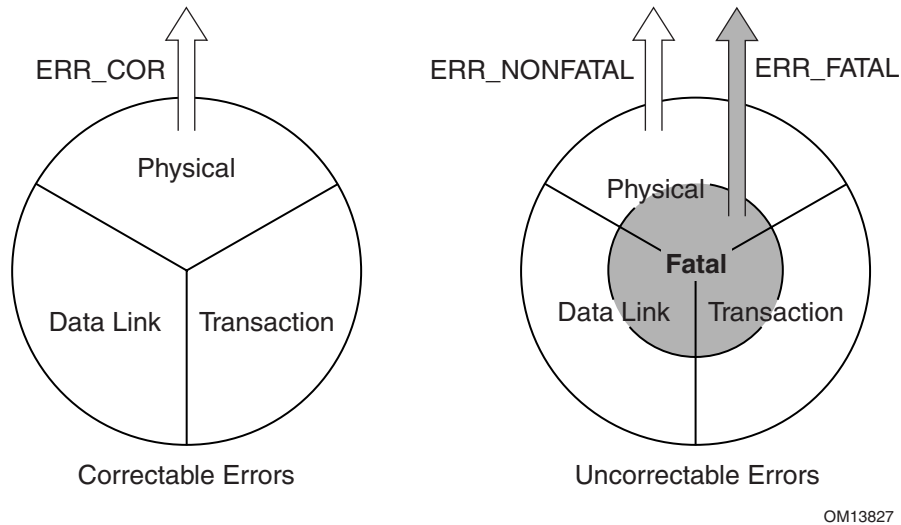
Figure 6-1: Error Classification

Classification of error severity as Fatal, Uncorrectable, and Correctable provides the platform with mechanisms for mapping the error to a suitable handling mechanism. For example, the platform might choose to respond to correctable errors with low priority, performance monitoring software. Such software could count the frequency of correctable errors and provide Link integrity

5   information. On the other hand, a platform designer might choose to map Fatal errors to a system-wide reset. It is the decision of the platform designer to map these PCI Express severity levels onto platform level severities.

6.2.2.1.   Correctable Errors

Correctable errors include those error conditions where the PCI Express protocol can recover

10   without any loss of information. Hardware corrects these errors and software intervention is not required. For example, an LCRC error in a TLP which is corrected by Data Link Level Retry is considered a correctable error. Logging the frequency of correctable errors may be helpful for profiling the integrity of a Link.

6.2.2.2.   Uncorrectable Errors

15   Uncorrectable errors are those error conditions that impact functionality of the interface. There is no PCI Express mechanism defined in this specification to correct these errors. For more robust error handling by the system, PCI Express further classifies uncorrectable errors as Fatal and Non-fatal.

6.2.2.2.1. Fatal Errors

Fatal errors are uncorrectable error conditions which render the particular PCI Express Link and related hardware unreliable. For Fatal errors, a reset of the components on the Link may be required to return to reliable operation. Platform handling of Fatal errors, and any efforts to limit the effects of these errors, is platform implementation specific.

Comparing with PCI/PCI-X, reporting a Fatal error is somewhat analogous to asserting SERR#.

6.2.2.2.2. Non-Fatal Errors

Non-fatal errors are uncorrectable errors which cause a particular transaction to be unreliable but the Link is otherwise fully functional. Isolating Non-fatal from Fatal errors provides system management software the opportunity to recover from the error without resetting the components on the Link and disturbing other transactions in progress. Devices not associated with the transaction in error are not impacted by the error.

6.2.3.   Error Signaling

There are two complementary mechanisms in PCI Express which allow the agent detecting an error to alert the system or the initiating device that an error has occurred. The first mechanism is through a Completion Status and the second method is with in-band error Messages.

Note that it is the responsibility of the agent detecting the error to signal the error appropriately.

Section 6.2.6 describes all the errors and how the hardware is required to respond when the error is detected.

6.2.3.1. Completion Status

The Completion Status field in the Completion header indicates when the associated Request failed (refer to Section 2.2.9). This is the only method of error reporting in PCI Express which enables the Requestor to associate an error with a specific Request. In other words, since Non-Posted Requests are not considered complete until after the Completion returns, the Completion Status field gives the initiator an opportunity to "fix" the problem at some higher level protocol (outside the scope of this specification). For example, if a Read is issued to prefetchable memory space and the Completion returns with a Unsupported Request Completion Status, perhaps due to a temporary condition, the initiator may choose to reissue the Read Request without side effects. Note that from a PCI Express point of view, the reissued Read Request is a distinct Request, and there is no relationship (on PCI Express) between the initial Request and the reissued Request.

6.2.3.2.   Error Messages

Error Messages are sent to the Root Complex for reporting the detection of errors according to the severity of the error.

5   When multiple errors of the same severity are detected, the corresponding error Messages may be merged for different errors of the same severity. At least one error Message must be sent for detected errors of each severity level.

Table 6-1: Error Messages

Error Message	Description
ERR_COR	This Message is issued when the component or device detects a correctable error on the PCI Express interface. Refer to Section 6.2.2.1 for the definition of a correctable error.
ERR_NONFATAL	This Message is issued when the component or device detects a Non-fatal, uncorrectable error on the PCI Express interface. Refer to Section 6.2.2.2.2 for the definition of a Non-fatal, uncorrectable error.
ERR_FATAL	This Message is issued when the component or device detects a Fatal, uncorrectable error on the PCI Express interface. Refer to Section 6.2.2.2.1 for the definition of a Fatal, uncorrectable error.

For these Messages, the Root Complex identifies the initiator of the Message by the Requester ID of the Message header. The Root Complex translates these error Messages into platform level events.

6.2.3.2.1.   Uncorrectable Error Severity Programming (Advanced Error
10   Reporting)

For devices implementing the Advanced Error Reporting capability, the Uncorrectable Errors Severity register allows each uncorrectable error to be programmed to Fatal or Non-Fatal. Uncorrectable errors are not recoverable using defined PCI Express mechanisms. However, some platforms or devices might consider a particular error fatal to a Link or device while another
15   platform considers that error non-fatal. The default value of the Uncorrectable Errors Severity register serves as a starting point for this specification but the register can be reprogrammed if the device driver or platform software requires more robust error handling.

Baseline error handling does not support severity programming.

6.2.3.2.2.   Masking Error Messages

20   Section 6.2.6 lists all the errors governed by this specification and describes when each of the above error Messages are issued. The transmission of these error Messages may be masked using the Reporting Enable fields of the Device Control register (see Section 7.8.4).

For devices implementing the Advanced Error Reporting capability the Uncorrectable Errors Mask register and Correctable Errors Mask register allows each error condition to be masked
25   independently. If Messages for a particular type of error are blocked by the setting in the Device

Control register, then no Messages of that type will be sent regardless of the values for the corresponding mask register.

When an error is masked, the error status bits are still affected but no error reporting Message is sent to the Root Complex, and the Header Log and First Error Pointer registers are unmodified.

6.2.3.2.3. Error Pollution

Error pollution can occur if error conditions for a given transaction are not isolated to the error's first occurrence. For example, assume the Physical Layer detects a Receiver Error. This error is detected at the Physical Layer and an error is reported to the Root Complex. To avoid having this error propagate and cause subsequent errors at upper layers (for example, a TLP error at the Data Link Layer), making it more difficult to determine the root cause of the error, subsequent errors which occur for the same packet will not be signaled at the Data Link or Transaction layers. Similarly, when the Data Link Layer detects an error, subsequent errors which occur for the same packet will not be signaled at the Transaction Layer. This behavior applies only to errors that are associated with a particular packet – other errors are reported for each occurrence.

6.2.4. Error Logging

Section 6.2.6 lists all the errors governed by this specification and for each error, the logging requirements are specified. Devices that do not support the Advanced Error Reporting capability log only the Device Status register bits indicating that an error has been detected. Note that some errors are also reported using the reporting mechanisms in the PCI compatible (Type 00h and 01h) configuration registers. Section 7.5 describes how these register bits are affected by the different types of error conditions described in this section.

For devices supporting the Advanced Error Reporting capability, each of the errors in Table 6-2, Table 6-3, and Table 6-4 corresponds to a particular bit in the Uncorrectable Error Status register or Correctable Error Status register. These registers are used by software to determine more precisely which error and what severity occurred. For specific Transaction Layer errors the associated TLP header is recorded in the Header Log register if the error is the first uncorrectable error detected (corresponding to the setting of the First Error Pointer register).

In a multi-function device, PCI Express errors that are not related to any specific function within the device, are logged in the corresponding status and logging registers of all functions in that device.

The following PCI Express errors are not function-specific:

❑ All Physical Layer errors

❑ All Data Link Layer errors

❑ The following are Transaction Layer errors:

- ECRC Fail

- UR, when caused by no function claiming a TLP

- Receiver Overflow

- Flow Control Protocol Error

- Malformed TLP

On the detection of one of these errors, a multi-function device should generate at most one error reporting Message of a given severity, where the Message must report the Requestor ID of a
5 function of the device that is enabled to report that specific type of error. If no function is enabled to send a reporting Message, the device does not send a reporting Message. If all reporting-enabled functions have the same severity level set for the error, only one error Message is sent. If all reporting-enabled functions do not have the same severity level set for the error, one error Message for each severity level is sent. Software is responsible for scanning all functions in a multi-function
10 device when it detects one of those errors.

6.2.4.1. Root Complex Considerations (Advanced Error Reporting)

In addition to the above logging, a Root Complex that supports the Advanced Error Reporting capability is required to implement the Error Source Identification register, which records the
15 Requester ID of the first ERR_NONFATAL/ERR_FATAL (uncorrectable errors) and ERR_COR (correctable errors) Messages received by the Root Complex. System software written to support Advanced Error Reporting can use the Root Port Error Status register to determine which fields hold valid information.

6.2.4.2. Multiple Error Handling (Advanced Error Reporting
20 Capability)

For the Advanced Error Reporting capability, the Uncorrectable Error Status register and Correctable Error Status register accumulate the collection of errors which occur on that particular PCI Express interface. The bits remain set until explicitly cleared by software or reset. Since multiple bits might be set in the Uncorrectable Error Status register, the First Error Pointer register
25 points to the unmasked uncorrectable error that occurred first. The First Error Pointer register is valid when the corresponding bit of the Uncorrectable Error Status register is set. The First Error Pointer value is not meaningful when the corresponding bit of the Uncorrectable Error Status register is not set, or is an unimplemented or undefined bit. For errors which require header logging, the Header Log register loaded according to the same rules as the First Error Pointer
30 register (such that the Header Log register will correspond to the error indicated in the First Error Pointer register, when the First Error Pointer register is valid).

Since the First Error Pointer and Header Log registers are only loaded for the first occurrence of an uncorrectable error, it is important that software services these registers in a timely manner, to limit the risk of missing this information for subsequent errors.

6.2.5. Sequence of Device Error Signaling and Logging Operations

Figure 6-2 shows the sequence of operations related to signaling and logging of errors detected by a device.
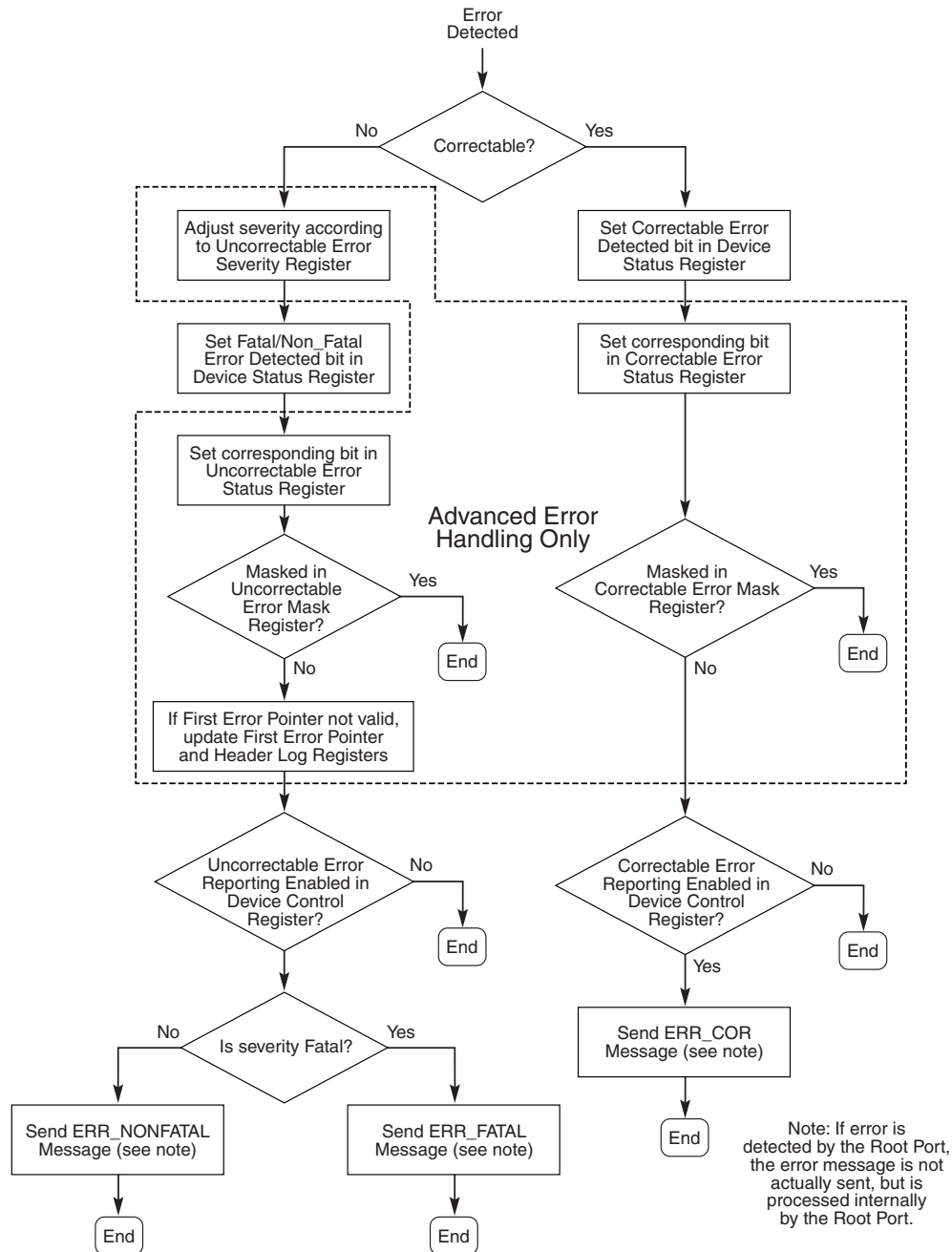
Figure 6-2: Flowchart Showing Sequence of Device Error Signaling and Logging Operations

269

6.2.6.   Error Listing and Rules

Table 6-2 and Table 6-3 list all of the PCI Express errors that are defined by this specification. Each error is listed with a short-hand name, how the error is detected in hardware, the default severity of the error, and the expected action taken by the agent which detects the error. These actions form the rules for PCI Express error reporting and logging.

The Default Severity column specifies the default severity for the error without any software reprogramming. For devices supporting the Advanced Error Reporting capability, the uncorrectable errors are programmable to Fatal or Non-fatal with the Error Severity register. Devices without Advanced Error Reporting capability use the default associations and are not reprogrammable.

<p align="center">Table 6-2: Physical Layer Error List</p>

Error Name	Default Severity	Detecting Agent Action[56]
Receiver Error	Correctable	*Receiver (if checking):* Send ERR_COR to Root Complex.
Training Error	Uncorrectable (Fatal)	If checking, send ERR_FATAL to Root Complex.[57]

<p align="center">Table 6-3: Data Link Layer Error List</p>

Error Name	Severity	Detecting Agent Action[56]
Bad TLP	Correctable	*Receiver:* Send ERR_COR to Root Complex.
Bad DLLP		*Receiver:* Send ERR_COR to Root Complex.
Replay Timeout		*Transmitter:* Send ERR_COR to Root Complex.
REPLAY NUM Rollover		*Transmitter:* Send ERR_COR to Root Complex.
Data Link Layer Protocol Error	Uncorrectable (Fatal)	If checking, send ERR_FATAL to Root Complex.

---

[56] For these tables, detecting agent action is given as if all enable bits are set to "enable" and, for Advanced Error Handling, mask bits are disabled and severity bits are set to their default values. Actions must be modified according to the actual settings of these bits.

[57] Only the component closer to the Root Complex is typically capable of sending the error Message.

Table 6-4: Transaction Layer Error List

Error Name	Severity	Detecting Agent Action[56]
Poisoned TLP Received	Uncorrectable (Non-Fatal)	*Receiver (if data poisoning is supported):* Send ERR_NONFATAL to Root Complex. Log the header of the Poisoned TLP.[58]
ECRC Check Failed		*Receiver:* Send ERR_NONFATAL to Root Complex. Log the header of the TLP that encounter the ECRC error.
Unsupported Request (UR)		*Request Receiver:* Send ERR_NONFATAL to Root Complex. Log the header of the TLP that caused the error.
Completion Timeout		*Requester:* Send ERR_NONFATAL to Root Complex.
Completer Abort		*Completer:* Send ERR_NONFATAL to Root Complex. Log the header of the Request that encountered the error.
Unexpected Completion		*Receiver:* Send ERR_NONFATAL to Root Complex. Log the header of the Completion that encountered the error. Note that if the Unexpected Completion is a result of misrouting, the Completion Timeout mechanism will be triggered at the corresponding Requester.
Receiver Overflow	Uncorrectable (Fatal)	*Receiver (if checking):* Send ERR_FATAL to Root Complex.
Flow Control Protocol Error		*Receiver (if checking):* Send ERR_FATAL to Root Complex.
Malformed TLP		*Receiver:* Send ERR_FATAL to Root Complex. Log the header of the TLP that encountered the error.

For all errors listed above, the appropriate status bit(s) must be set upon detection of the error. For Unsupported Request (UR), the logging and signaling control/status mechanisms are more complicated than for the other types of errors, and so are clarified below. Note that upon receiving a Non-Posted (NP) Request that is UR, the return of a Completion with UR status is unconditional, and the reporting settings in the Device Control (and Uncorrectable Error Mask register if

---

[58] Advanced Error Handling only.

Advanced Error Reporting (AER) is implemented) register(s) do not affect the return of this completion.

In the Device Status register:

❑ Set Unsupported Request Detected any time a received Request is determined to be UR.

5 ❑ Set Non-Fatal Error Detected any time a received Request is determined to be UR, and either AER is not implemented or AER Unsupported Request Error Severity is set to Non-Fatal.

❑ Set Fatal Error Detected any time a received Request is determined to be UR, and AER is implemented, and AER Unsupported Request Error Severity is set to Fatal.

UR is also reported through the AER structure if implemented.

10 If the Unsupported Request Reporting Enable bit in the Device Control register is not set, the device must not send an error Message in the case of a UR, although the device will set status bits as described above (regardless of AER settings if AER is implemented).

If AER is not implemented, and if the Unsupported Request Reporting Enable bit in the Device Control register is set, an ERR_NONFATAL error Message will be sent to the Root Complex (or 15 signaled within the Root Complex, in the case of a Root Port) on detection of a UR.

If AER is implemented and the Unsupported Request Reporting Enable bit in the Device Control register is set, an error Message is sent if the Unsupported Request Error Mask bit is not set. The severity of the error Message is determined by the setting of the Unsupported Request Error Severity bit.

20 ## 6.2.6.1. PCI Mapping

In order to support PCI driver and software compatibility, PCI Express error conditions, where appropriate, must be mapped onto the PCI Status register bits for error reporting.

In other words, when certain PCI Express errors are detected, the appropriate PCI Status register bit is set alerting the error to legacy PCI software. While the PCI Express error results in setting the 25 PCI Status register, clearing the PCI Status register will not result in clearing bits in the Uncorrectable Error Status register and Correctable Error Status register. Similarly, clearing bits in the Uncorrectable Error Status register and Correctable Error Status register will not result in clearing the PCI Status register.

The PCI command register has bits which control PCI error reporting. However, the PCI 30 Command register does not affect the setting of the PCI Express error register bits.

6.2.7.   Virtual PCI Bridge Error Handling

Virtual PCI Bridge configuration headers are associated with each PCI Express Port in a Root Complex or a Switch. For these cases, PCI Express error concepts require appropriate mapping to the PCI error reporting structures.

6.2.7.1.   *Error Forwarding and PCI Mapping for Bridge - Rules*

In general, a TLP is either passed from one side of the Virtual PCI Bridge to the other, or is handled at the ingress side of the Bridge according to the same rules which apply to the ultimate recipient of a TLP. The following rules cover PCI Express specific error related cases:

❑ If a Request does not address a space mapped to the egress side of the Bridge, the Request is terminated at the ingress side as an Unsupported Request

❑ Poisoned TLPs are forwarded according to the same rules as non-Poisoned TLPs

  • When forwarding a Poisoned TLP from Primary to Secondary:

    ♦ the Receiving side must set the Detected Parity Error bit in the Status register

    ♦ the Transmitting side must set the Master Data Parity Error bit in the Secondary Status register if the Parity Error Response bit in the Bridge Control register is set

  • When forwarding a Poisoned TLP from Secondary to Primary:

    ♦ the Receiving side must set the Detected Parity Error bit in the Secondary Status register

    ♦ the Transmitting side must set the Master Data Parity Error bit in the Status register if the Parity Error Response bit in the Bridge Control register is set

❑ ERR_COR, ERR_NONFATAL and ERR_FATAL are forwarded from the secondary interface to primary interface, if the SERR# Enable bit in the Command and Bridge Control register is set

6.3.   Virtual Channel Support

6.3.1.   Introduction and Scope

Virtual Channel mechanism provides a foundation for supporting differentiated services within the PCI Express fabric. It enables deployment of independent physical resources that together with traffic labeling are required for optimized handling of differentiated traffic. Traffic labeling is supported using Transaction Class TLP-level labels. Exact policy for traffic differentiation is determined by the TC/VC mapping and by the VC-based arbitration. The TC/VC mapping depends on the platform application requirements. These requirements drive the choice of VC arbitration algorithm and configurability/programmability of arbiters allows detailed tuning of the traffic servicing policy.

Basic definition of Virtual Channel mechanism and associated Traffic Class labeling mechanism is covered in Chapter 2. The VC configuration/programming model is defined in Section 7.11.

The remaining sections of this chapter cover VC mechanisms from the system perspective. They address the next level details on:

❑ Supported TC/VC configurations

❑ VC-based arbitration – algorithms and rules

5 ❑ Traffic ordering considerations

❑ Isochronous support as a specific usage model

6.3.2. TC/VC Mapping and Example Usage

A Virtual Channel is established when one or more TC labels are associated with a physical resource designated by a VC ID. Every Traffic Class that is supported on a given path within the fabric must
10 be mapped to one of the enabled Virtual Channels. Every Port must support the default TC0/VC0 pair – this is "hardwired." Any additional TC label mapping or additional VC resource enablement is optional and is controlled by system software using the programming model described in Section 7.11.

The number of VC resources provisioned within a component or enabled within a given fabric may
15 vary due to implementation and usage model requirements, due to Hot-Plug of disparate components with varying resource capabilities, or due to system software restricting what resources may be enabled on a given path within the fabric.

Some examples to illustrate:

❑ A set of components (Root Complex, Endpoints, Switches) may only support the mandatory
20 VC0 resource that must have TC0 mapped to VC0. System software may, based on application usage requirements, map one or all non-zero TC labels to VC0 as well on any or all paths within the fabric.

❑ A set of components may support two VC resources, e.g., VC0 and VC1. System software must map TC0/VC0 and in addition, may map one or all non-zero TC labels to either VC0 or VC1.
25 As above, these mappings may be enabled on any or all paths within the fabric. See the examples below for additional information.

❑ A Switch may be implemented with eight Ports – seven x1 Links with two VC resources and one x16 Link with one VC resource. System software may enable both VC resources on the x1 Links and assign one or more additional TC labels to either VC thus allowing the Switch to
30 differentiate traffic flowing between any Ports. The x16 Link must be also configured to map any non-TC0 traffic to VC0 if such traffic is to flow on this Link. Note: multi-Port components (Switches and Root Complex) are required to support independent TC/VC mapping per PCI Express Port.

In any of the above examples, system software has the ability to map one, all, or a subset of the TC
35 labels to a given VC. Should system software wish to restrict the number of traffic classes that may flow through a given Link, it may configure only a subset of the TC labels to the enabled VC resources. Any TLP that does not contain a TC label that has been mapped to an enabled VC resource shall be treated as a malformed TLP and dropped by the receiving Port. This is referred to as TC Filtering.

A graphical example of TC filtering is illustrated in Figure 6-3, where TC labels [2:6] are not mapped to the Link that connects Endpoint A and the Switch. This means that the TLPs with TC labels [2:6] are not allowed between the Switch and Endpoint A.
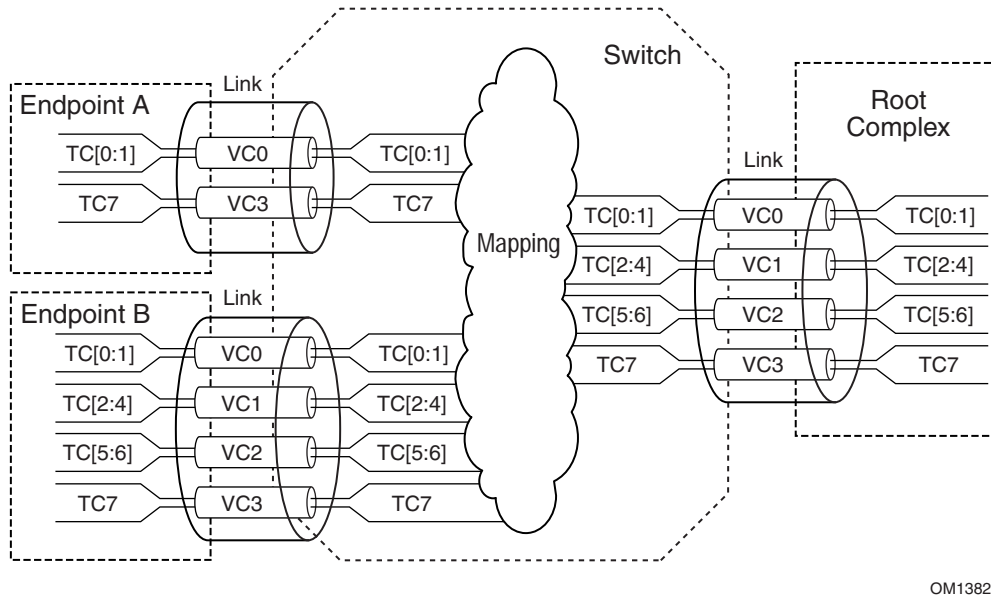
OM13828

Figure 6-3: TC Filtering Example

Figure 6-4 shows an example of TC to VC mapping. A simple Switch with one Downstream Port
5 and one Upstream Port connects an Endpoint to a Root Complex. At the Upstream Port, two VCs (VC0 and VC1) are enabled with the following mapping: TC(0-6)/VC0, TC7/VC1. At the Downstream Port, only VC0 is enabled and all TCs are mapped to VC0. In this example while TC7 is mapped to VC0 at the Downstream Port, it is re-mapped to VC1 at the Upstream Port. Although the Endpoint device only supports VC0, when it labels transactions with different TCs, transactions
10 with TC7 label from/to the Endpoint device can take advantage of the two Virtual Channels enabled between the Switch and the Root Complex.
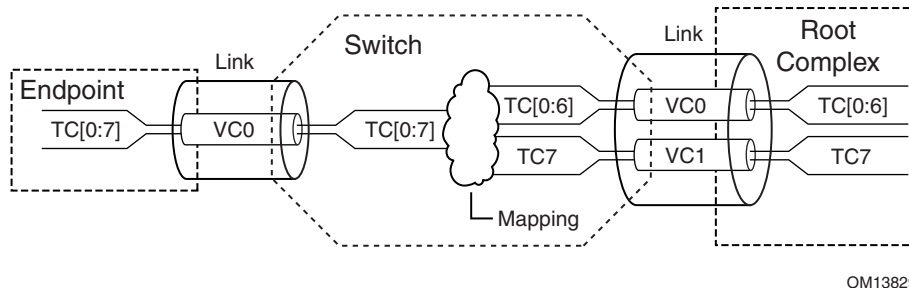
OM13829

Figure 6-4: TC to VC Mapping Example

✍ IMPLEMENTATION NOTE

Multiple TCs Over a Single VC

A single VC implementation may benefit from using multiple TC labels. TC labels provide ordering domains that may be used to differentiate traffic within the Endpoint or the Root Complex independent of the number of VCs supported.

In a simple configuration, where only VC0 is supported, traffic differentiation may not be accomplished in an optimum manner since the different traffic classes cannot be physically segregated. However, the benefits of carrying multiple TC labels can still be exploited particularly in the small and "shallow" topologies where Endpoints are connected directly to Root Complex rather than through cascaded Switches. In these topologies traffic that is targeting Root Complex only needs to traverse a single Link, and an optimized scheduling of packets on both sides (Endpoint and Root Complex) based on TC labels may accomplish significant improvement over the case when a single TC label is used. Still, inability to route differentiated traffic through separate resources with fully independent flow-control and independent ordering exposes all of the traffic to the potential head-of-line blocking conditions. Optimizing Endpoint internal architecture to minimize the exposure to the blocking conditions can reduce those risks.

6.3.3. VC Arbitration

Arbitration is one of the key aspects of the Virtual Channel mechanism and is defined in a manner that fully enables configurability to the specific application. In general, the definition of the PCI Express VC-based arbitration mechanism is driven by the following objectives:

❑ To prevent false transaction timeouts and to guarantee data flow forward progress

❑ To provide differentiated services between data flows within the fabric

❑ To provide guaranteed bandwidth with deterministic (and reasonably small) end-to-end latency between components

PCI Express Links are bidirectional, i.e., each PCI Express Port can be an Ingress or an Egress Port depending on the direction of traffic flow. This is illustrated by the example of a 3-Port Switch in Figure 6-5, where traffic flows between Switch Ports are highlighted with different types of lines. In the following sections, PCI Express VC Arbitration is defined using a Switch arbitration model since a Switch is the PCI Express element that represents a functional superset from the arbitration perspective.

In addition, one-directional data flow is used in the description.
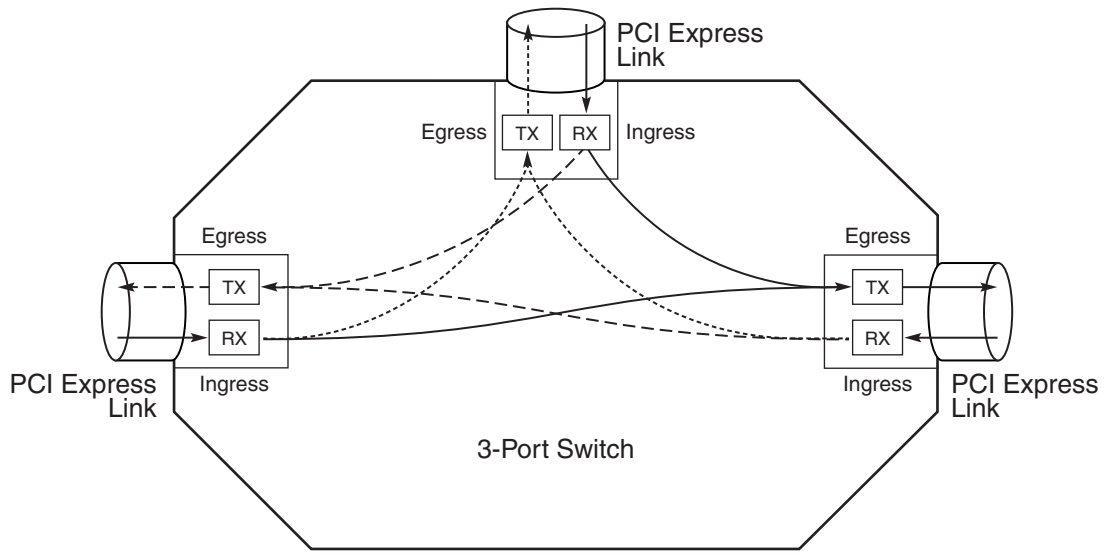
OM13830

Figure 6-5: An Example of Traffic Flow Illustrating Ingress and Egress

6.3.3.1. Traffic Flow and Switch Arbitration Model

The following set of figures (Figure 6-6 and Figure 6-7) illustrates traffic flow through the Switch and summarizes the key aspects of the arbitration.

OM14284
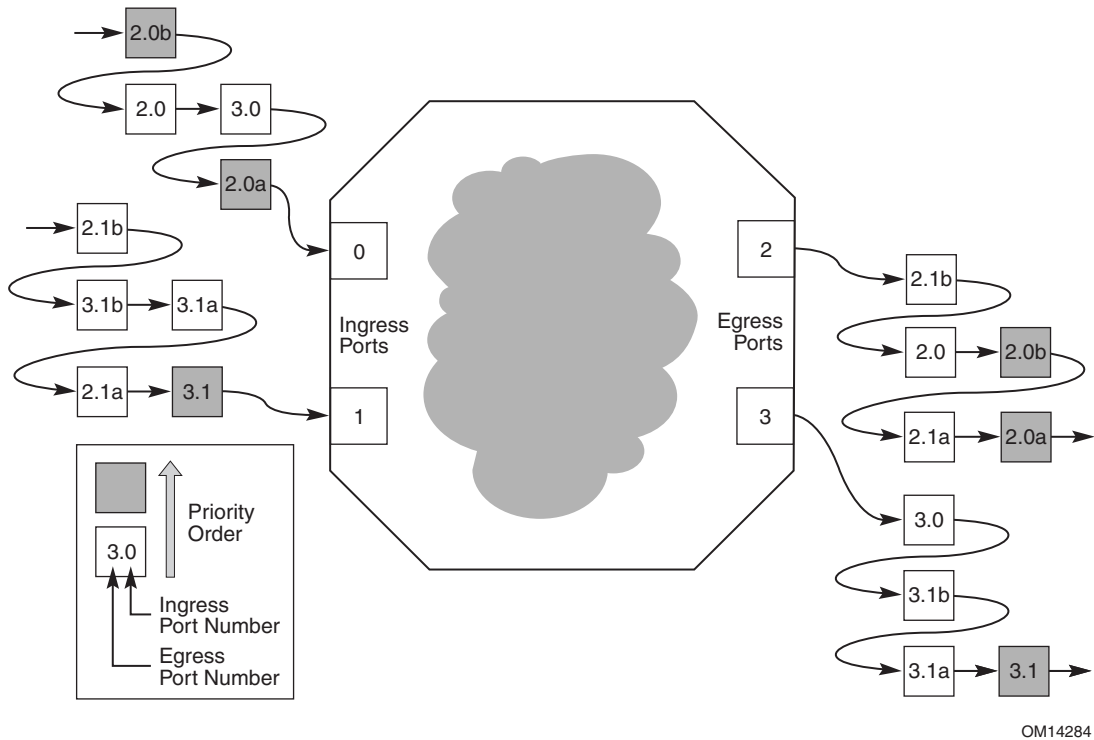
Figure 6-6: An Example of Differentiated Traffic Flow Through a Switch

At each Ingress Port an incoming traffic stream is represented in Figure 6-6 by small boxes. These boxes represent packets that are carried within different VCs that are distinguished using different levels of gray. Each of the boxes that represents a packet belonging to different VC includes designation of Ingress and Egress Ports to indicate where the packet is coming from and where it is going to. For example, designation "3.0" means that this packet is arriving at Port #0 (Ingress) and is destined to Port #3 (Egress). Within the Switch packets are routed and serviced based on Switch internal arbitration mechanisms.

Switch arbitration model defines a required arbitration infrastructure and functionality within a Switch. This functionality is needed to support a set of arbitration policies that control traffic contention for an Egress Port from multiple Ingress Ports.

Figure 6-7 shows a conceptual model of a Switch highlighting resources and associated functionality in ingress to egress direction. Note that each Port in the Switch can have the role of an Ingress or Egress Port. Therefore, this figure only shows one particular scenario where the 4-Port Switch in this example has ingress traffic on Port #0 and Port #1, that targets Port #2 as an Egress Port. A different example may show different flow of traffic implying different roles for Ports on the Switch. PCI Express architecture enables peer-to-peer communication through the Switch and, therefore,

possible scenarios using the same example may include multiple separate and simultaneous ingress to egress flows (e.g., Port 0 to Port 2 and Port 1 to Port 3).
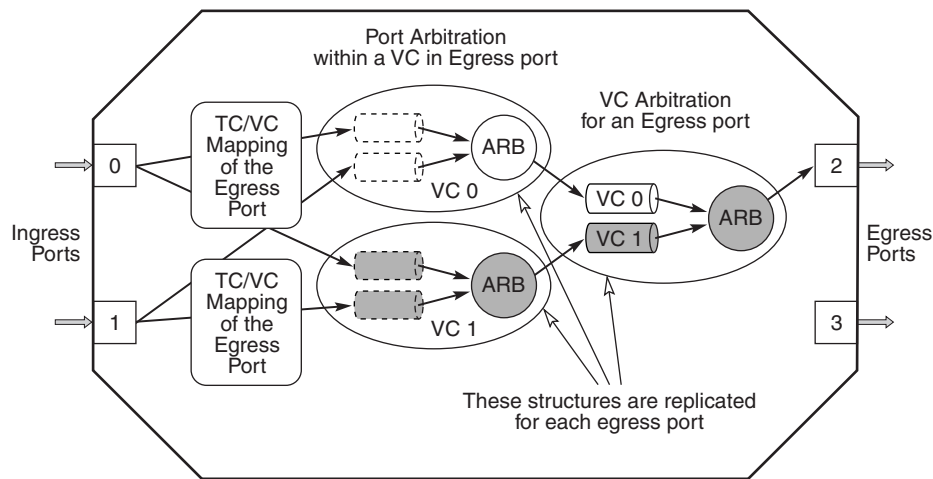
OM14493

Figure 6-7: Switch Arbitration Structure

The following two steps conceptually describe routing of traffic received by the Switch on Port 0 and Port 1 and destined to Port 2. First, the target Egress Port is determined based on address/routing information in the TLP header. Secondly, the target VC of the Egress Port is determined based on the TC/VC map of the Egress Port. Transactions that target the same VC in the Egress Port but are from different Ingress Ports must be arbitrated before they can be forwarded to the corresponding resource in the Egress Port. This arbitration is referred to as the Port Arbitration.

Once the traffic reaches the destination VC resource in the Egress Port, it is subject to arbitration for the shared Link. From the Egress Port point of view this arbitration can be conceptually defined as a simple form of multiplexing where the multiplexing control is based on arbitration policies that are either fixed or configurable/programmable. This stage of arbitration between different VCs at an Egress Port is called the VC Arbitration of the Egress Port.

Independent of VC arbitration policy, a management/control logic associated with each VC must observe transaction ordering and flow control rules before it can make pending traffic visible to the arbitration mechanism.

✎ IMPLEMENTATION NOTE

VC Control Logic at the Egress Port

VC control logic at every Egress Port includes:

5 ❑ VC Flow Control logic

❑ VC Ordering Control logic

Flow control credits are exchanged between two Ports connected to the same Link. Availability of flow-control credits is one of the qualifiers that VC control logic must use to decide when a VC is allowed to compete for the shared Link resource (i.e., Data Link Layer transmit/retry buffer). If a
10 candidate packet cannot be submitted due to the lack of an adequate number of flow control credits, VC control logic must mask presence of pending packet to prevent blockage of traffic from other VCs. Note that since each VC includes buffering resources for Posted, Non-Posted Requests and Completion packets, the VC control logic must also take into account availability of flow control credits for the particular candidate packet. In addition, VC control logic must observe ordering rules
15 (see Section 2.4 for more details) for Posted/Non-Posted/Completion transactions to prevent deadlocks and violation of producer-consumer ordering model.

✎ IMPLEMENTATION NOTE

Arbitration for Multi-Function Endpoints

The arbitration of data flows from different functions of a multi-function Endpoint is beyond the
20 scope of this specification. Mapping of different data flows (within a multi-function Endpoint) to different TCs and VCs is implementation specific. Multi-function Endpoints, however, should support PCI Express VC-based arbitration control mechanisms if multiple VCs are implemented for the PCI Express Link.

When a common VC on the PCI Express Link is shared by multiple functions, the aggregated traffic
25 over the VC is subject to the bandwidth and latency regulations for that VC on the PCI Express Link. The multi-function Endpoints should implement proper arbitration for data flows from different functions in order to share the Link resources and achieve desired end-to-end services.

6.3.3.2. VC Arbitration – Arbitration Between VCs

PCI Express defines a default VC prioritization via the VC Identification (VC ID) assignment, i.e.,
30 the VC ID are arranged in ascending order of relative priority in the Virtual Channel Capability structure. The example in Figure 6-8 illustrates a Port that supports eight VCs with VC0 treated as the lowest priority and VC7 as the highest priority.

VC Resource	VC ID	Relative Priority	VC Arbitration	Usage Example
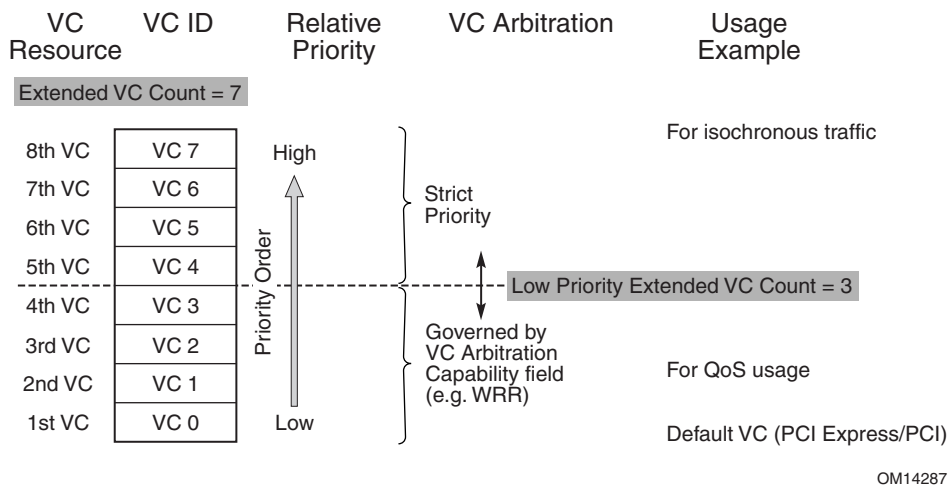
Figure 6-8: VC ID and Priority Order – An Example

The availability of default prioritization does not restrict the type of algorithms that may be implemented to support VC arbitration – either implementation-specific or one of the architecture-defined methods:

❑ Strict Priority – Based on inherent prioritization, i.e., VC0=lowest, VC7=highest

5 ❑ Round Robin (RR) – Simplest form of arbitration where all VCs have equal priority

❑ Weighted RR – Programmable weight factor determines the level of service

If strict priority arbitration is supported by the hardware for a subset of the VC resources, software can configure the VCs into two priority groups – a lower and an upper group. The upper group is treated as a strict priority arbitration group while the lower group that is arbitrated to only when

10 there are no packets to process in the upper group. Figure 6-8 illustrates an example configuration that supports eight VCs separated into two groups – the lower group consisting of VC0-VC3 and the upper group consisting of VC4-VC7. The arbitration within the lower group can be configured to one of the supported arbitration methods. The Low Priority Extended VC Count field in the Port VC Capability register 1 indicates the size of this group. The arbitration methods are listed in

15 the VC Arbitration Capability field in the Port VC Capability register 2. See Section 7.11 for details. When the Low Priority Extended VC Count field is set to zero, all VCs are governed by the strict-priority VC arbitration; when the field is equal to the Extended VC Count, all VCs are governed by the VC arbitration indicated by the VC Arbitration Capability field.

6.3.3.2.1. Strict Priority Arbitration Model

20 Strict priority arbitration enables minimal latency for high-priority transactions. However, there is potential danger of bandwidth starvation should it not be applied correctly. Using strict priority requires all high-priority traffic to be regulated in terms of maximum peak bandwidth and Link usage duration. Regulation must be applied either at the transaction injection Port or within subsequent Egress Ports where data flows contend for a common Link. System software must

25 configure traffic such that lower priority transactions will be serviced at a sufficient rate to avoid transactions timeouts.

6.3.3.2.2. Round Robin Arbitration Model

Round Robin arbitration is used to provide, at the transaction level, equal[59] opportunities to all traffic. Note that this scheme is used where different unordered streams need to be serviced with the same priority.

5   In the case where differentiation is required, a Weighted Round Robin scheme can be used. The WRR scheme is commonly used in the case where bandwidth regulation is not enforced by the sources of traffic and therefore it is not possible to use the priority scheme without risking starvation of lower priority traffic. The key is that this scheme provides fairness during traffic contention by allowing at least one arbitration win per arbitration loop. Assigned weights regulate

10   both minimum allowed bandwidth and maximum burstiness for each VC during the contention. This means that it bounds the arbitration latency for traffic from different VCs. Note that latencies are also dependent on the maximum packet sizes allowed for traffic that is mapped onto those VCs.

One of the key usage models of the WRR scheme is support for QoS policy where different QoS levels can be provided using different weights.

15   Although weights can be fixed (by hardware implementation) for certain applications, to provide more generic support for different applications, PCI Express components that support the WRR scheme are recommended to implement programmable WRR. Programming of WRR is controlled using the software interface defined in Section 7.11.

6.3.3.3.   Port Arbitration – Arbitration Within VC

20   Arbitration within a VC refers to the arbitration between the traffic that is mapped onto the same VC but is coming from different Ingress Ports. Inherent prioritization scheme that makes sense when talking about arbitration among VCs in this context is not applicable since it would imply strict arbitration priority for different Ports. Traffic from different Ports can be arbitrated using the following supported schemes:

25   ❑ Hardware-fixed arbitration scheme, e.g., Round Robin

❑ Programmable WRR arbitration scheme

❑ Programmable Time-based WRR arbitration scheme

Hardware-fixed RR or RR-like scheme is the simplest to implement since it does not require any programmability. It makes all Ports equal priority, which is acceptable for applications where no

30   software-managed differentiation or per-Port-based bandwidth budgeting is required.

Programmable WRR allows flexibility since it can operate as flat RR or if differentiation is required, different weights can be applied to traffic coming from different Ports in the similar manner as described in Section 6.3.3.2. This scheme is used where different allocation of bandwidth needs to be provided for different Ports.

35   A Time-based WRR is used for applications where not only different allocation of bandwidth is required but also a tight control of usage of that bandwidth. This scheme allows control of the amount of traffic that can be injected from different Ports within certain fixed period of time. This

---

[59] Note that this does not imply equivalence and fairness in the terms of bandwidth usage.

is required for certain applications such as isochronous services, where traffic needs to meet a strict deadline requirement. Section 6.3.4 provides basic rules to support isochronous applications. For more details on time-based arbitration and on the isochronous service as a usage model for this arbitration scheme refer to Appendix A.

6.3.4.   Isochronous Support

Servicing isochronous data transfer requires a system to provide not only guaranteed data bandwidth but also deterministic service latency. The isochronous support mechanisms in PCI Express are defined to ensure that isochronous traffic receives its allocated bandwidth over a relevant period of time while also preventing starvation of the other traffic in the system. Isochronous support mechanisms apply to communication between Endpoint and Root Complex as well as to peer-to-peer communication.

Isochronous service is realized through proper use of PCI Express mechanisms such as TC transaction labeling, VC data-transfer protocol, and TC-to-VC mapping. End-to-end isochronous service requires software to set up proper configuration along the path between the Requester to Completer. This section describes the rules for software configuration and the rules hardware components must follow to provide end-to-end isochronous services. More information and background material regarding isochronous applications and isochronous service design guidelines can be found in Appendix A.

6.3.4.1.   Rules for Software Configuration

System software must obey the following rules to configure PCI Express fabric for isochronous traffic:

❑ Software must designate one or more TC labels for isochronous transactions.

❑ The setting of the Attribute fields of all isochronous requests targeting the same Completer must be fixed and identical.

❑ Software must configure all VC resources used to support isochronous traffic to be serviced (arbitrated) at the requisite bandwidth and latency to meet the application objectives. This may be accomplished using strict priority, WRR, or hardware-fixed arbitration.

❑ For each Switch Egress Port and RCRB that supports isochronous data flows, the associated Port arbitration must support time-based arbitration for the configured VC.

❑ Software should not intermix isochronous traffic with non-isochronous traffic on a given VC.

❑ Software must observe the Maximum Time Slots capability reported by the PCI Express Port or RCRB.

❑ Software must not assign all PCI Express Link capacity to isochronous traffic. This is required to ensure the requisite forward progress of other non-isochronous transactions to avoid false transaction timeouts.

❑ Software must limit the Max_Payload_Size for each path that supports isochronous to meet the isochronous latency. For example, all traffic flowing on a path from an isochronous capable device to the Root Complex should be limited to packets that do not exceed the Max_Payload_Size required to meet the isochronous latency requirements.

5 ❑ Software must set Max_Read_Request_Size of an isochronous-configured device with a value that does not exceed the Max_Payload_Size set for the device.

6.3.4.2. Rules for Requesters

A Requester requiring isochronous services must obey the following rules:

❑ The value in the Length field of read requests must never exceed Max_Payload_Size.

10 ❑ If isochronous traffic targets the Root Complex and the RCRB indicates it cannot meet the isochronous bandwidth and latency requirements without requiring all transactions to set the No Snoop attribute bit, indicated by setting the Reject Snoop Transactions field, then this bit must be set within the TLP header else the transaction will be rejected.

6.3.4.3. Rules for Completers

15 A Completer providing isochronous services must obey the following rules:

❑ A Completer should not apply flow control induced backpressure to uniformly injected isochronous requests under normal operating conditions.

❑ A Completer must report its isochronous bandwidth capability in the Maximum Time Slots field in the VC Resource Capability register. Note that a Completer must account for partial writes.

20 ❑ A Completer must observe the maximum isochronous transaction latency.

❑ A Root Complex as a Completer must implement RCRB and support time-based Port Arbitration mechanism for the associated VCs. Note that the time-based Port Arbitration only applies to request transactions.

6.3.4.4. Rules for Switch Components

25 A Switch component providing isochronous services must obey the following rules:

❑ An isochronous-configured Switch Port should not apply flow control induced backpressure to uniformly injected isochronous requests under normal operating conditions.

❑ An isochronous-configured Switch must observe the maximum isochronous transaction latency.

❑ A Switch component must support time-based Port Arbitration mechanism for each Port that
30 supports one or more VCs capable of supporting isochronous traffic. Note that the time-based Port Arbitration only applies to request transactions but not to completion transactions.

6.4.   Device Synchronization

System software requires a "stop" mechanism for ensuring that there are no outstanding transactions for a particular device in a system. For example, without such a mechanism renumbering bus numbers during system operation may cause the Requestor ID (which includes the bus number) for a given device to change while Requests or Completions for that device are still in flight, and may thus be rendered invalid due to the change in the Requester ID. It is also desirable to be able to ensure that there are no outstanding transactions during a Hot-Plug orderly removal.

The details of stop mechanism implementation depend on the device hardware, device driver software, and system software. However, the fundamental requirements which must be supported to allow system software management of the PCI Express fabric include the abilities to:

❑ Block the device from generating new Requests

❑ Block the generation of Requests issued to the device

❑ Determine that all Requests being serviced by the device have been completed

❑ Determine that all non-posted Requests initiated by the device have completed

❑ Determine that all posted Requests initiated by the device have reached their destination

The ability of the driver and/or system software to block new Requests from the device is supported by the Bus Master Enable, SERR Enable, and Interrupt Disable bits in the Command register (Section 7.5.1.1), and other such control bits.

Requests issued to the device are generally under the direct control of the driver, so system software can block these Requests by directing the driver to stop generating them (the details of this communication are system software specific). Similarly, Requests serviced by the device are normally under the device driver's control, so determining the completion of such requests is usually trivial.

The Transaction Pending, or TP bit, provides a consistent way for software to determine that all non-posted Requests issued by the device have been completed (see Chapter 7).

Determining that posted Requests have reached their destination is handled by generating a transaction to "flush" any outstanding Requests. Writes to system memory using TC0 will be flushed by host reads of the device, and so require no explicit flush protocol. Writes using TCs other than TC0 require some type of flush synchronization mechanism. The mechanism itself is implementation specific to the device and its driver software. However, in all cases the device hardware and software implementers should thoroughly understand the ordering rules described in Section 2.4. This is especially true if the Relaxed Ordering flag is set for any Requests initiated by the device.

IMPLEMENTATION NOTE

Flush Mechanisms

In a simple case such as that of an Endpoint device communicating only with host memory through TC0, "flush" can be implemented simply by reading from the device. If the device issues writes to main memory using TCs other than TC0, "flush" can be implemented with a memory read on the corresponding TCs directed to main memory. The memory read needs to be performed on all TCs that the device is using.

If a memory read is used to "flush" outstanding transactions, but no actual read is required, it may be desirable to use the zero-length read semantic described in Section 2.2.5.

Peer-to-peer interaction between devices requires an explicit synchronization protocol between the involved devices, even if all communication is through TC0. For a given system, the model for managing peer-to-peer interaction must be established. System software, and device hardware and software must then conform to this model. The requirements for blocking Request generation and determining completion of Requests match the requirements for non-peer interaction, however the determination that Posted Requests have reached peer destination device(s) requires an explicit synchronization mechanism. The mechanism itself is implementation specific to the device, its driver software, and the model used for the establishment and disestablishment of peer communications.

6.5.   Locked Transactions

6.5.1.   Introduction

Locked Transaction support is required to prevent deadlock in systems that use legacy software which causes the accesses to I/O devices. Note that some CPUs may generate locked accesses as a result of executing instructions that implicitly trigger lock. Some legacy software misuses these transactions and generates locked sequences even when exclusive access is not required. Because locked accesses to I/O devices introduce potential deadlocks apart from those mentioned above, as well as serious performance degradation, PCI Express Endpoints are prohibited from supporting locked accesses, and new software must not use instructions which will cause locked accesses to I/O devices. Legacy Endpoints support locked accesses only for compatibility with existing software.

Only the Root Complex is allowed to initiate Locked Requests on PCI Express. Locked Requests initiated by Endpoints and Bridges are not supported. This is consistent with limitations for locked transaction use outlined in the *PCI Local Bus Specification, Rev 2.3* (Appendix F- Exclusive Accesses).

This section specifies the rules associated with supporting locked accesses from the Host CPU to legacy Endpoints, including the propagation of those transactions through Switches and PCI Express/PCI Bridges.

6.5.2.   Initiation and Propagation of Locked Transactions - Rules

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). When a lock is established, all other traffic is blocked
⁵ from using the path between the Root Complex and the locked legacy Endpoint or Bridge.

❑ A locked transaction sequence or attempted locked transaction sequence is initiated on PCI Express using the "lock"–type Read Request/Completion (MRdLk/CplDLk) and terminated with the Unlock Message

- Locked Requests which are completed with a status other than Successful Completion do
¹⁰   not establish lock (explained in detail in the following sections)

- Regardless of the status of any of the Completions associated with a locked sequence, all locked sequences and attempted locked sequences must be terminated by the transmission of an Unlock Message.

- MRdLk, CplDLk and Unlock semantics are allowed only for the default Traffic Class (TC0)

¹⁵ - Only one locked transaction sequence attempt may be in progress at a given time within a single hierarchy domain

❑ The Unlock Message is sent from the Root Complex down the locked transaction path to the Completer, and may be broadcast from the Root Complex to all Endpoints and Bridges

- Any device which is not involved in the locked sequence must ignore this Message

²⁰ The initiation and propagation of a locked transaction sequence through PCI Express is performed as follows:

❑ A locked transaction sequence is started with a MRdLk Request

- Any successive reads for the locked transaction sequence must also use MRdLk Requests

- The Completions for any successful MRdLk Request use the CplDLk Completion type, or
²⁵   the CPlLk Completion type for unsuccessful Requests

❑ If any read associated with a locked sequence is completed unsuccessfully, the Requester must assume that the atomicity of the lock is no longer assured, and that the path between the Requester and Completer is no longer locked

❑ All writes for the locked sequence use MWr Requests

³⁰ ❑ The Unlock Message is used to indicate the end of a locked sequence

- A Switch propagates Unlock Messages to the locked Egress Port

❑ Upon receiving an Unlock Message, a legacy Endpoint or Bridge must unlock itself if it is in a locked state

- If not locked, or if the Receiver is a PCI Express Endpoint or Bridge which does not
³⁵   support lock, the Unlock Message is ignored and discarded

6.5.3.   Switches and Lock - Rules

Switches must distinguish transactions associated with locked sequences from other transactions to prevent other transactions from interfering with the lock and potentially causing deadlock. The following rules cover how this is done. Note that locked accesses are limited to TC0, which is
⁵ always mapped to VC0.

❑ When a Switch propagates a MRdLk Request from the Ingress Port (closest to the Root Complex) to the Egress Port, it must block all Requests which map to the default Virtual Channel (VC0) from being propagated to the Egress Port

• If a subsequent MRdLk Request is Received at this Ingress Port addressing a different
¹⁰   Egress Port, the behavior of the Switch is undefined

Note: This sort of split-lock access is not supported by PCI Express and software must not cause such a locked access. System deadlock may result from such accesses.

❑ When the CplDLk for the first MRdLk Request is returned, if the Completion indicates a Successful Completion status, the Switch must block all Requests from all other Ports from
¹⁵ being propagated to either of the Ports involved in the locked access, except for Requests which map to non-VC0 on the Egress Port

❑ The two Ports involved in the locked sequence must remain blocked as described above until the Switch receives the Unlock Message (at the Ingress Port for the initial MRdLk Request)

• The Unlock Message must be forwarded to the locked Egress Port

²⁰ • The Unlock Message may be broadcast to all other Ports

• The Ingress Port is unblocked once the Unlock Message arrives, and the Egress Port(s) which were blocked are unblocked following the Transmission of the Unlock Message out of the Egress Ports

♦ Ports which were not involved in the locked access are unaffected by the Unlock
²⁵   Message

6.5.4.   PCI Express/PCI Bridges and Lock - Rules

The requirements for PCI Express/PCI Bridges are similar to those for Switches, except that, because PCI Express/PCI Bridges use only the default Virtual Channel and Traffic Class, all other traffic is blocked during the locked access. The requirements on the PCI bus side of the PCI
³⁰ Express/PCI Bridge match the requirements for a PCI/PCI Bridge (see the *PCI-to-PCI Bridge Architecture Specification, Rev. 1.1* and the *PCI Express to PCI/PCI-X Bridge Specification, Rev. 1.0*).

6.5.5.   Root Complex and Lock - Rules

A Root Complex is permitted to support locked transactions as a Requestor. If locked transactions are supported, a Root Complex must follow the sequence described in Section 6.5.2 to perform a
³⁵ locked access. The mechanisms used by the Root Complex to interface PCI Express to the Host CPU(s) are outside the scope of this document.

6.5.6.   Legacy Endpoints

Legacy Endpoints are permitted to support locked accesses, although their use is discouraged. If locked accesses are supported, legacy Endpoints must handle them as follows:

❑ The legacy Endpoint becomes locked when it Transmits the first Completion for the first Read
   Request of the locked access with a Successful Completion status

   • If the completion status is not Successful Completion, the legacy Endpoint does not become
     locked

   • Once locked, the legacy Endpoint must remain locked until it receives the Unlock Message

❑ While locked, a legacy Endpoint must not issue any Requests using Traffic Classes which map to
   the default Virtual Channel (VC0)

   Note that this requirement applies to all possible sources of Requests within the Endpoint, in
   the case where there is more than one possible source of Requests.

   • Requests may be issued using Traffic Classes which map to VCs other than the default
     Virtual Channel

6.5.7.   PCI Express Endpoints

PCI Express Endpoints do not support lock. A PCI Express Endpoint must treat a MRdLk Request as an Unsupported Request (see Chapter 2).

6.6.   PCI Express Reset - Rules

This section specifies the behavior of PCI Express reset. This section covers the relationship between the architectural mechanisms defined in this document and the reset mechanisms defined in this document. Any relationship between the PCI Express reset and component or platform reset is component or platform specific (respectively).

In all form factors and system hardware configurations, there must, at some level, be a hardware mechanism for setting or returning all Port states to the initial conditions specified in this document – this mechanism is called "Fundamental Reset". This mechanism can take the form of an auxiliary signal provided by the system to a component or add-in card, in which case the signal must be called PERST#, and must conform to the rules specified in Section 4.2.4.5.1. When PERST# is provided to a component or add-in card, this signal must be used by the component or add-in card to cause a Fundamental Reset. When PERST# is not provided to a component or add-in card, Fundamental Reset is generated autonomously by the component or an add-in card, and the details of how this is done are outside the scope of this document. If a Fundamental Reset is generated autonomously by the component or add-in card, and if power is supplied by the platform to the component/add-in card, the component/add-in card must generate a Fundamental Reset to itself if the supplied power goes outside of the limits specified for the form factor or system.

❑ There are three distinct types of reset: cold, warm, and hot:

- A Fundamental Reset must occur following the application of power to the component. This is called a cold reset.

- In some cases, it may be possible for the Fundamental Reset mechanism to be triggered by hardware without the removal and re-application of power to the component. This is called a warm reset. This document does not specify a means for generating a warm reset.

- There is an in-band mechanism for propagating reset across a Link. This is called a hot reset and is described in Section 4.2.4.5.

   Note also that the Data Link Layer reporting DL_Down is in some ways identical to a hot reset – see Section 2.9.

❑ On exit from any type of reset (cold, warm, or hot), all Port registers and state machines must be set to their initialization values as specified in this document, except for sticky registers (see Section 7.4).

❑ On exit from a Fundamental Reset, the Physical Layer will attempt to bring up the Link (see Section 4.2.5). Once both components on a Link have entered the initial Link Training state, they will proceed through Link initialization for the Physical Layer and then through Flow Control initialization for VC0, making the Data Link and Transaction Layers ready to use the Link

- Following Flow Control initialization for VC0, it is possible for TLPs and DLLPs to be transferred across the Link

Following a reset, some devices may require additional time before they are able to respond to Requests they receive. Particularly for Configuration Requests it is necessary that components and devices behave in a deterministic way, which the following rules address.

The first set of rules addresses requirements for components and devices:

❑ A component must enter the initial active Link Training state within 80 ms of the end of Fundamental Reset (Link Training is described in Section 4.2.4)

- Note: In some systems, it is possible that the two components on a Link may exit Fundamental Reset at different times. Each component must observe the requirement to enter the initial active Link Training state within 80 ms of the end of Fundamental Reset from its own point of view.

❑ On the completion of Link Training (entering the DL_Active state, see Section 3.2), a component must be able to receive and process TLPs and DLLPs

The second set of rules addresses requirements placed on the system:

❑ To allow components to perform internal initialization, system software must wait for at least 100 ms from the end of a reset of one or more device before it is permitted to issue Configuration Requests to those devices

- A system must guarantee that all components intended to be software visible at boot time are ready to receive Configuration Requests within 100 ms of the end of Fundamental Reset at the Root Complex – how this is done is beyond the scope of this specification

❑ The Root Complex and/or system software must allow 1.0s (+50%/-0%) after a reset of a device, before it may determine that a device which fails to return a Successful Completion status for a valid Configuration Request is a broken device

Note: This delay is analogous to the Trhfa parameter specified for PCI/PCI-X, and is intended to allow an adequate amount of time for devices which require self initialization.

❑ When attempting a Configuration access to devices on a PCI or PCI-X segment behind a PCI Express/PCI(-X) Bridge, the timing parameter T_{rhfa} must be respected

When a Link is in normal operation, the following rules apply:

❑ If, for whatever reason, a normally operating Link goes down, the Transaction and Data Link Layers will enter the DL_Inactive state (see Sections 2.9 and 3.2.1)

❑ For any Root or Switch Downstream Port, setting the Secondary Bus Reset bit of the Bridge Control register associated with the Port must cause a hot reset to be sent (see Section 4.2.4.5).

❑ For a Switch, the following must cause a hot reset to be sent on all Downstream Ports:

• Setting the Secondary Bus Reset bit of the Bridge Control register associated with the Upstream Port

• The Data Link Layer of the Upstream Port reporting DL_Down

• Receiving a hot reset on the Upstream Port

Certain aspects of Fundamental Reset are specified in this document and others are specific to a platform, form factor and/or implementation. Specific platforms, form factors or application spaces may require the additional specification of the timing and/or sequencing relationships between the components of the system for Fundamental Reset. For example, it might be required that all PCI Express components within a chassis observe the assertion and deassertion of Fundamental Reset at the same time (to within some tolerance). In a multi-chassis environment, it might be necessary to specify that the chassis containing the Root Complex be the last to exit Fundamental Reset.

In all cases where power and PERST# are supplied, the following parameters must be defined:

❑ T_{pvperl} – PERST# must remain active at least this long after power becomes valid

❑ T_{perst} – When asserted, PERST# must remain asserted at least this long

❑ T_{fail} – When power becomes invalid, PERST# must be asserted within this time

Additional parameters may be specified.

In all cases where a reference clock is supplied, the following parameter must be defined:

❑ $T_{perst-clk}$ – PERST# must remain active at least this long after any supplied reference clock is stable

Additional parameters may be specified.

✎ IMPLEMENTATION NOTE

Use of PERST#

The deassertion of PERST# may not be sufficient in all implementations to say that fundamental reset is exited. Such implementations may require other additional conditions to be met; for
5   example, the I/O PLL locking, before exiting fundamental reset.

6.7.　　PCI Express Native Hot-Plug Support

The PCI Express architecture is designed to natively support both Hot-Plug and hot-removal of devices. This section defines the standard usage model defined for all PCI Express form factors supporting Hot-Plug and hot removal of devices. This usage model provides the foundation for
10 how indicators and push buttons should behave if implemented in a system. The definitions of indicators and push buttons apply to all PCI Express Hot-Plug models.

6.7.1.　　PCI Express Hot-Plug Usage Model

6.7.1.1.　Usage Model Rationale

A standard usage model is beneficial to customers who buy systems with Hot-Plug slots because
15 many customers utilize hardware and software from different vendors. A standard usage model allows customers to use the Hot-Plug slots on all of their systems without having to retrain operators. The PCI Express Hot-Plug standard usage model is derived from the standard usage model defined in the *PCI Standard Hot-Plug Controller and Subsystem Specification, Rev 1.0* and is identical from the user perspective; note that only slight changes were made in register definitions.
20 Conformance to the standard usage model is required by all PCI Express form factors that implement Hot-Plug and use indicators and buttons. Specific requirements for individual elements of the standard usage model are form factor specific, and are defined in the individual form factor specifications.

All PCI Express form factors that Support Hot-Plug/Remove are required to conform to the
25 Standard Usage Model. Deviating from the Standard Usage Model causes the solution to be non-PCI Express compliant and will create issues that would not exist otherwise, such as:

- ❑ User confusion

- ❑ More extensive hardware testing

- ❑ Functional incompatibilities with system software

30 - ❑ Encountering untested paths in system software

6.7.1.2. Elements of the Standard Usage Model

Table 6-5: Elements of the Standard Usage Model

Element	Purpose
Indicators	Shows the power and attention state of the slot
Manually-operated Retention Latch (MRL)	Holds add-in cards in place
MRL Sensor	Allows the Port and system software to detect the MRL being opened
Electromechanical Interlock	Prevents removal of add-in cards while slot is powered
Attention Button	Allows user to request Hot-Plug operations
Software User Interface	Allows user to request Hot-Plug operations
Slot Numbering	Provides visual identification of slots

6.7.1.2.1. Indicators

The Standard Usage Model defines two indicators; the Power Indicator and the Attention Indicator. The platform can provide the two indicators per slot or module bay and the indicators can be
implemented either on the chassis or on the module. Each indicator is in one of three states: on, off, or blinking. Hot-Plug system software has exclusive control of the indicator states by writing the command registers associated with the indicator.

The Hot-Plug capable Port controls blink frequency, duty cycle, and phase. Blinking indicators operate at a frequency of 1 to 2 Hz and 50% (+/- 5%) duty cycle. Blinking indicators are not
required to be synchronous and in-phase between Ports.

Indicators must be placed in close proximity to their associated Hot-Plug slot if indicators are implemented on the chassis so that the association between the indicators and the Hot-Plug slot is clear.

Both indicators are completely under the control of system software. The Switch device or Root
Port never changes the state of an indicator in response to an event such as a power fault or unexpected MRL opening unless commanded to do so by software. An exception is granted to platforms capable of detecting stuck-on power faults. In the specific case of a stuck-on power fault, the platform is permitted to override the Switch device or Root Port and force the Power Indicator to be on (as an indication that the add-in card should not be removed). In all cases, the Ports
internal state for the Power Indicator must match the software selected state. The handling by system software of stuck-on faults is optional and not described in this specification. Therefore, the platform vendor must ensure that this optional feature of the Standard Usage Model is addressed via other software, platform documentation, or by other means.

6.7.1.2.1.1. Attention Indicator

The Attention Indicator is yellow or amber in color and is used to indicate that an operational problem exists or that the Hot-Plug slot is being identified so that a human operator can locate it easily.

Table 6-6: Attention Indicator States

Indicator Appearance	Meaning
Off	Normal - Normal operation
On	Attention - Operational problem at this slot
Blinking	Locate - Slot is being identified at the user's request

Attention Indicator Off

When the Attention Indicator is off, it means that neither the add-in card (if one is present) nor the Hot-Plug slot requires attention.

Attention Indicator On

When the Attention Indicator is on, it means an operational problem exists at the card or slot.

An operational problem is a condition that prevents continued operation of an add-in card. The operating system or other system software determines whether a specific condition prevents continued operation of an add-in card and whether lighting the Attention Indicator is appropriate. Examples of operational problems include problems related to external cabling, add-in cards, software drivers, and power faults. In general, when the Attention Indicator is on, it means that an operation was attempted and failed or that an unexpected event occurred.

The Attention Indicator is not used to report problems detected while validating the request for a Hot-Plug operation. Validation is a term applied to any check that system software performs to assure that the requested operation is viable, permitted, and will not cause problems. Examples of validation failures include denial of permission to perform a Hot-Plug operation, insufficient power budget, and other conditions that may be detected before an operation begins.

Attention Indicator Blinking

When the Attention Indicator is blinking, it means that system software is identifying this slot for a human operator to find. This behavior is controlled by a user (for example, from a software user interface or management tool).

6.7.1.2.1.2. Power Indicator

The Power Indicator is green in color and is used to indicate the power state of the slot. Table 6-7 lists the Power Indicator states.

Table 6-7: Power Indicator States

Indicator Appearance	Meaning
Off	Power Off - Insertion or removal of add-in cards is permitted. All supply voltages (except Vaux) have been removed from the slot if required for add-in card removal. Note that Vaux is removed when the MRL is open.
On	Power On - The slot is powered on. Insertion or removal of add-in cards is not permitted.
Blinking	Power Transition - The slot is in the process of powering up or down. Insertion or removal of add-in cards is not permitted.

Power Indicator Off

5   When the Power Indicator is off, it means that insertion or removal of an add-in card is permitted. Main power to the slot is off if required by the form factor. An example of main power removal is the PCI Express card form factor. If the platform provides Vaux to Hot-Plug slots and the MRL is closed, any signals switched by the MRL are connected to the slot even when the Power Indicator is off. Signals switched by the MRL are disconnected when the MRL is opened. System software
10   must cause a slot's Power Indicator to be turned off when the slot is not powered and/or it is permissible to insert or remove add-in cards. See the appropriate electromechanical specifications for form factor details.

Power Indicator On

When the Power Indicator is on, it means that main power to the slot is on and that insertion or
15   removal of an add-in card is not permitted.

Power Indicator Blinking

When the Power Indicator is blinking, it means that the slot is powering up or powering down and that insertion or removal of an add-in card is not permitted. A blinking Power Indicator also provides visual feedback to the human operator when the Attention Button is pressed.

20   ## 6.7.1.2.2. Manually-operated Retention Latch (MRL)

An MRL is a manually-operated retention mechanism that holds an add-in card in the slot and prevents the user from removing the card. The MRL rigidly holds the card in the slot so that cables may be attached without the risk of creating intermittent contact. MRLs that hold down two or more add-in cards simultaneously are permitted in Platforms that do not provide MRL Sensors.

6.7.1.2.3. MRL Sensor

The MRL Sensor is a switch, optical device, or other type of sensor that reports the position of a slot's MRL to the Port. The MRL Sensor reports closed when the MRL is fully closed and open at all other times (that is, fully open and intermediate positions).

Signals switched by the MRL (such as Vaux and SMBus) must be automatically removed from the slot when the MRL Sensor indicates that the MRL is open and must be restored to the slot when the MRL Sensor indicates that MRL has closed again.

The MRL Sensor allows the Port to monitor the position of the MRL and therefore allows the Port to detect unexpected openings of the MRL. When an unexpected opening of the MRL associated with a slot is detected, the Port changes the state of that slot to disabled and notifies system software. The Port does not autonomously change the state of either the Power Indicator or the Attention Indicator.

In the absence of an MRL sensor, staggered presence detect pins could be used to handle the switched signals. In this case, when the presence pins break contact, the switched signals will be automatically removed from the slot.

IMPLEMENTATION NOTE

MRL Sensor Handling

If an MRL Sensor is implemented without a corresponding MRL Sensor input on the PCI Express Hot-Plug Controller, it is recommended that the MRL Sensor be routed to power fault input of the Hot-Plug Controller. This allows an active PCI Express card or module to be powered off when MRL is opened.

6.7.1.2.4. Electromechanical Interlock

An electromechanical interlock is a mechanism for physically locking the add-in card or MRL in place until the system software and Port release it. Implementation of the interlock is optional. There is no mechanism in the programming interface for explicit control of electromechanical interlocks. The Standard Usage Model assumes that if electromechanical interlocks are implemented, they are controlled by the same Port output signal that enables main power to the slot. Systems may optionally expand control of interlocks to provide physical security of the add-in cards.

6.7.1.2.5. Attention Button

An Attention Button is a momentary-contact push button switch, located adjacent to each Hot-Plug slot or on a module that is pressed by the user to initiate a hot-insertion or a hot removal at that slot.

The Power Indicator provides visual feedback to the human operator (if the system software accepts the request initiated by the Attention Button) by blinking. Once the Power Indicator begins

blinking, a 5-second abort interval exists during which a second depression of the Attention Button cancels the operation.

If an operation initiated by an Attention Button fails for any reason, it is recommended that system software present a Message explaining the failure via a software user interface or add the Message to a system log.

6.7.1.2.6. Software User Interface

System software provides a user interface that allows hot insertions and hot removals to be initiated and that allows occupied slots to be monitored. A detailed discussion of Hot-Plug user interfaces is operating system specific and is therefore beyond the scope of this document.

On systems with multiple Hot-Plug slots, the system software must allow the user to initiate operations at each slot independent of the states of all other slots. Therefore, the user is permitted to initiate a Hot-Plug operation on one slot using either the software user interface or the Attention Button while a Hot-Plug operation on another slot is in process, regardless of which interface was used to start the first operation.

6.7.1.2.7. Slot Numbering

A Physical Slot Identifier (as defined in PCI HP 1.1, Section 1.5) consists of an optional chassis number and the physical slot number of the Hot-Plug slot. System software determines the physical slot number from registers in the Port. The chassis number is 0 for the main chassis. The chassis number for other chassis must be a non-zero value obtained from a PCI-to-PCI Bridge's Chassis Number register (see PCI Bridge 1.1, Section 13.4).

The Standard Usage Model also requires that each physical slot number is globally unique within a chassis.

6.7.2. Event Behavior

Depending on the power state of the Switch device or Root Port, a Hot-Plug removal event may generate either a system interrupt or a PME (see Table 6-8).

Table 6-8: Event Behavior

Event	Register Bit Set When Detected	Cleared by	Port Optionally Generates the Following When Event is Detected:
Presence Detect Change	Presence Detect Event Status	Writing a 1 to the detected bit	System Interrupt, PME
Attention Button Pressed	Attention Button Pressed Event	Writing a 1 to the detected bit	System Interrupt, PME
MRL Sensor Changed	MRL Sensor Change Detected Event	Writing a 1 to the detected bit	System Interrupt, PME
Power Fault	Power Fault Detected Event	Writing a 1 to the detected bit.	System Interrupt, PME

6.7.3. Port Registers Grouped by Device Association

5   The registers described in this section are grouped by device to convey all registers associated with implementing each device in Ports. These registers are unique to each Switch device or Root Port implementing Hot-Plug slots, and are located in the PCI Express Capability structure (Section 7.8).

6.7.3.1. Attention Button Registers

Attention Button Present – This bit indicates if an Attention Button is implemented on the
10   chassis.

Attention Button Pressed – This bit is set when the Attention Button is pressed. This register is set by the debounced output of an Attention Button. This bit is also set by the Port receiving the Attention_Button_Pressed Message from the end device.

Attention Button Pressed Enable – This bit when set enables the generation of the Hot-Plug
15   interrupt or a wakeup event on an Attention Button Pressed event.

6.7.3.2.   Attention Indicator Registers

Attention Indicator Present – This bit indicates if an Attention Indicator is implemented on the chassis.

Attention Indicator Control – When read this register returns the current state of the Attention Indicator; when written the Attention Indicator is set to this state. If an Attention Indicator is implemented on the card, when written, the Port will send the appropriate Attention Indicator Message (determined by the decoding) to the device on the card.

6.7.3.3.   Power Indicator Registers

Power Indicator Present – This bit indicates if a Power Indicator is implemented on the chassis.

Power Indicator Control – When read this register returns the current state of the Power Indicator; when written the Power Indicator is set to this state. If a Power Indicator is implemented on the card, when written, the Port will send the appropriate Power Indicator Message (determined by the decoding) to the device on the card.

6.7.3.4.   Power Controller Registers

Power Controller Present – This bit indicates if a Power Controller is implemented for this slot.

Power Controller Control – When read, this register returns the current state of the Power applied to the slot; when written, the Power Controller turns on or off power to slot.

Power Fault Detected – This bit is set when the Power Controller detects a power fault at this slot.

Power Fault Detected Enable – This bit when set enables the generation of the Hot-Plug interrupt or a wakeup event on a power fault event.

6.7.3.5.   Presence Detect Registers

Presence Detect State – This bit indicates the presence of a card in the slot. The bit reflects the Presence Detect status determined via in-band mechanism or via Present Detect pins as defined in the PCI Express Card Electromechanical Specification. This register is required to be implemented on all Switch devices and Root Ports. The presence detect field for Switch devices or Root Ports not connected to slots should be hardwired to 1. This register is required if a slot implemented.

Presence Detect Changed Event – This bit is set when a Presence Detect change is detected.

Presence Detect Changed Enable – This bit when set enables the generation of the Hot-Plug interrupt or a wakeup event on a presence detect changed event

The in-band presence detect mechanism requires that power be applied to a card for its presence to be detected. Consequently, form factors that require a power controller for Hot-Plug must implement a physical pin presence detect mechanism as defined in the *PCI Express Card Electromechanical Specification.*

IMPLEMENTATION NOTE

No Slot Power Controller

For slots that do not implement a power controller, software must ensure that system power planes are enabled to provide power to slots prior to reading presence detect state.

6.7.3.6. MRL Sensor Registers

MRL Sensor Present – This bit indicates if an MRL Sensor is implemented on the chassis.

MRL Sensor Changed – This bit is set when the value of the MRL Sensor state changed.

MRL Sensor Changed Enable – This bit when set enables the generation of the Hot-Plug interrupt or a wakeup event on a MRL Sensor changed event.

MRL Sensor State – This register reports the status of the MRL Sensor if it is implemented.

If switched signals are handled automatically by detecting an extraction event using the PRSNT#1 and PRSNT#2 pins, then MRL Sensor registers need not be implemented.

6.7.3.7. Command Completed Registers

Command Completed – This bit is set when the Hot-Plug Controller completes an issued command and is ready to accept the next command.

Command Completed Interrupt Enable – This bit when set enables the generation of Hot-Plug interrupt when a command is completed by the Hot-Plug control logic.

PCI Express Hot-Plug Controllers are required to implement the Command Completed registers regardless of the form factor.

The command completed register applies only to commands issued by software to control the Attention Indicator, Power Indicator or power controller. The software must wait confirmation of command completion (notification via Command Completed interrupt or polling Command Completed register) before issuing the next command. However, if the Command Completed register is not set 1 second after the command is issued, the host software is allowed to repeat the command or to issue the next command.

IMPLEMENTATION NOTE

When Does a Hot-Plug Controller Set the Command Completed Register?

A command completed notification is an indication to host software that the Hot-Plug Controller has processed the previous command and is ready to receive the next command; it provides no guarantee that the action corresponding to the command is complete.

Similarly, a read to the Slot Status register following a write will always return the value from the latest write, although the corresponding command action may not be complete.

6.7.3.8. Port Capabilities and Slot Information Registers

Slot Implemented – This bit when set indicates that the Link associated with this Downstream Port is connected to a slot, as oppose to being connected to an integrated device or being disabled.

Physical Slot Number – This hardware initialized field indicates the physical slot number attached to the Port. This field must be hardware initialized to a value that assigns a slot number that is globally unique within the chassis. These registers should be initialized to 0 for Ports connected to integrated devices on the system board or integrated within the same silicon as the Switch device or Root Port.

Hot-Plug Capable – This bit when set indicates this slot is capable of supporting Hot-Plug.

Surprise Removal Capable – This bit when set indicates that the device might be removed from the system without any prior notification.

6.7.3.9. Hot-Plug Interrupt Control Register

Hot-Plug Interrupt Enable – This bit when set enables generation of the Hot-Plug interrupt on enabled Hot-Plug events.

6.7.4. Hot-Plug Registers on Devices

These registers located in the Device Capabilities register are used by system software for discovery of Attention Indicator, Power Indicator, or Attention Button presence on card or module on which the device is implemented. These registers are required to be implemented on all Endpoint devices.

Attention Button Present – This bit when set indicates that an Attention Button is implemented on the add-in card or module.

Attention Indicator Present – This bit when set indicates that an Attention Indicator is implemented on the add-in card or module.

Power Indicator Present – This bit indicates when set indicates that a Power Indicator is implemented on the add-in card or module.

6.7.5.   Messages

The Messages defined here allow for cards to implement indicators and buttons on the card without having to connect signals directly to the Port. Detailed explanation of each Message is located in Chapter 2.

6.7.5.1.   Messages for the Attention Indicator

This series of Messages allows the Attention Indicator be implemented on the card as opposed to the chassis. These Messages are sent by the Downstream Port to the device and instruct the device to set its Attention Indicator to the indicated state. The following Messages are used:

ATTENTION_INDICATOR_ON

ATTENTION_INDICATOR_BLINK

ATTENTION_INDICATOR_OFF

All Endpoint devices are required to handle the Attention Indicator Messages even if the device does not implement the indicators.

6.7.5.2.   Messages for Power Indicator

This series of Messages allows the Power Indicator be implemented on the card as opposed to the chassis. These Messages are sent by the Downstream Port to the device and instruct the device to set its Power Indicator to the indicated state. The following Messages are used:

POWER_INDICATOR_ON

POWER_INDICATOR_BLINK

POWER_INDICATOR_OFF

All Endpoint devices are required to handle the Power Indicator Messages even if the device does not implement the indicators.

6.7.5.3.   Messages for Attention Button

ATTENTION_BUTTON_PRESSED – This Message allows the attention button to be implemented on the card and informs the Port that the attention button has been pressed. Upon receipt of this Message the Port terminates the Message and sets the Attention Button Pressed bit in the Hot-Plug Event register.

All down stream Ports of Switches and Root Ports are required to handle the ATTENTION_BUTTON_PRESSED Message even if the slot is not Hot-Plug capable.

6.7.6.   Hot-Plug Slot Register Requirements

The following registers are required for all slots that implement the Hot-Plug capability:

❑ Attention Button Pressed and Attention Button Pressed Enable registers

Attention Button Pressed Enabled (bit 0 of Slot Control register) and Attention Button Pressed (bit 0 of Slot Status register) are required to be implemented by all Hot-Plug capable slots to provide support for the ATTENTION_BUTTON_PRESSED Message.

❑ Attention Indicator Control Register

Attention Indicator Control register (bits 7:6 of the Slot Control register) is required to be implemented by all Hot-Plug capable slots to provide support for the Attention Indicator Messages.

❑ Power Indicator Control Register

Power Indicator Control register (bits 9:8 of the Slot Control register) is required to be implemented by all Hot-Plug capable slots to provide support for the Power Indicator Messages.

❑ Command Completed Registers

All registers associated with Command Completion (Section 6.7.3.7) must be implemented by all Hot-Plug capable slots

❑ Presence Detect Registers

All registers associated with Presence Detect (Section 6.7.3.5) must be implemented by all Hot-Plug capable slots

6.7.7.   PCI Express Hot-Plug Interrupt/Wakeup Signal Logic

A Port with Hot-Plug compatibilities supports generation of Hot-Plug interrupts on the following Hot-Plug events:

❑ Attention Button Pressed

❑ Power Fault Detected

❑ MRL Sensor Changed

❑ Presence Detect Changed

When the system is in a sleep state or if the Hot-Plug capable Port is in a device state D1, D2, or D3$_{hot}$, the above Hot-Plug Controller events generate a wakeup event (using PME mechanism) instead of a Hot-Plug interrupt. Note that the Hot-Plug Controller generates the wakeup on behalf of the hot-plugged device, and it is not necessary for that device to have auxiliary (or main) power.

A Hot-Plug capable Port also supports generation of Hot-Plug interrupt when the Hot-Plug control logic completes an issued command. However, a command completed event will not cause a wakeup event if the system is in a sleep state or if the Hot-Plug capable Port is in a device state D1, D2, or D3$_{hot}$.

Figure 6-9 shows the logical connection between the Hot-Plug event logic and the system interrupt/wakeup event generation logic.
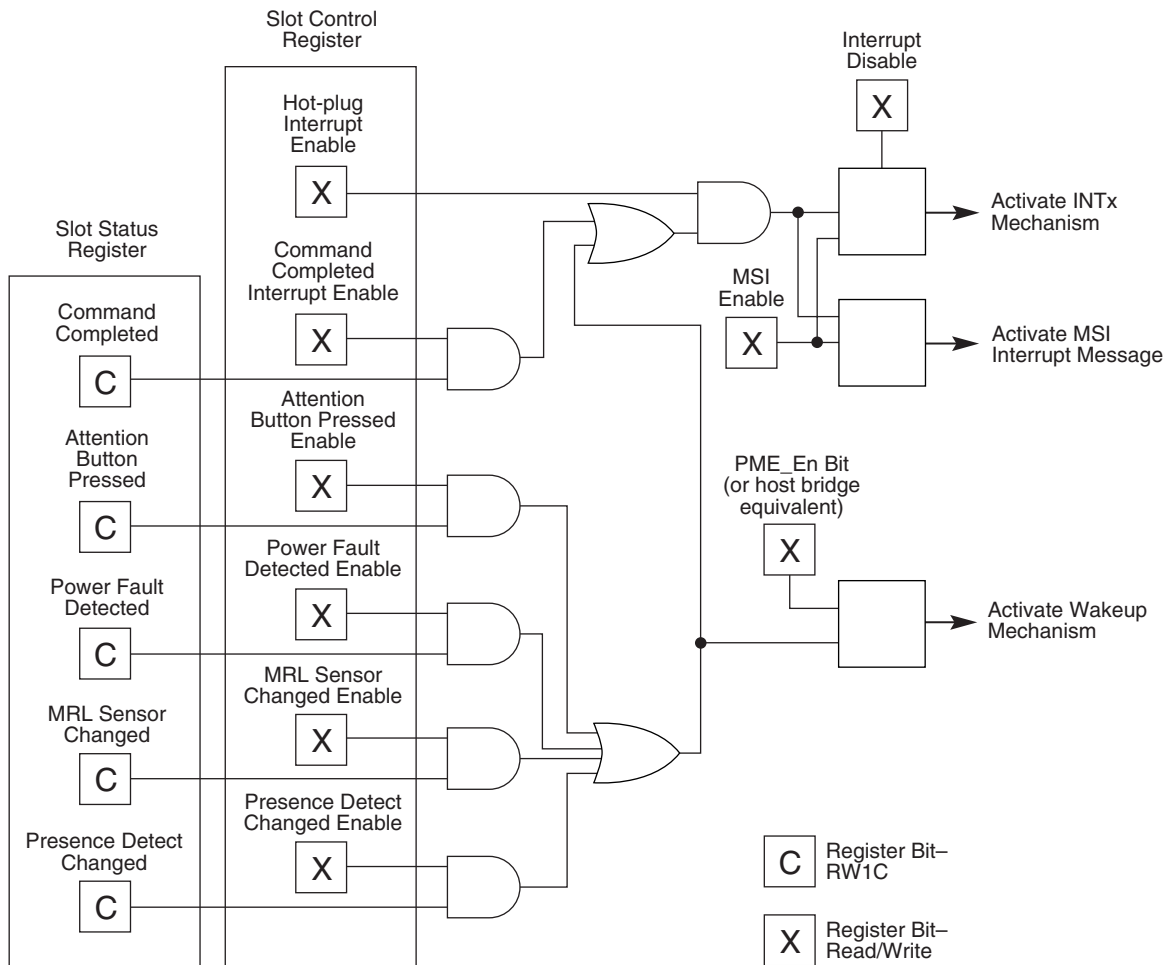
OM14494A

Figure 6-9: Hot-Plug Logic

6.7.8.   The Operating System Hot-Plug Method

Some systems that include Hot-Plug capable Root Ports and Switches that are released before
ACPI-compliant operating systems with native Hot-Plug support are available, can use ACPI
firmware for propagating Hot-Plug events. Firmware control of the Hot-Plug registers must be
disabled if an operating system with native support is used. Platforms that provide ACPI firmware
to propagate Hot-Plug events must also provide a control method to transfer control to the
operating system. This method is called Operating System Hot-Plug (OSHP) and is provided for
each Port that is Hot-Plug capable and being controlled by ACPI firmware.

Operating systems with native Hot-Plug support must execute the OSHP method, if present, for
each Hot-Plug capable Port before accessing the Hot-Plug registers and when returning from a
hibernated state. If a Port's OSHP method is executed multiple times, and the Switch to operating

304

system control has already been achieved, the method must return successfully without doing anything. After the OSHP method is executed, the firmware must not access the Port's Hot-Plug registers. If any signals such as the System Interrupt or PME# have been redirected for servicing by the firmware, they must be restored appropriately for operating system control.

5   The following is an example of a namespace entry for an SHPC that is managed by firmware.

```
Device(PPB1){

    ...

    Method(OSHP, 0) {
        // Disable firmware access to SHPC and restore
        // the normal System Interrupt and Wakeup mechanism.
    }
    ...
}
```

🗒 IMPLEMENTATION NOTE

Controlling Hot-Plug by Using ACPI

When using ACPI to control the Hot-Plug events, the following should be considered:

Firmware should redirect the System Interrupt to a GPE so that ACPI can service the interrupts
10   instead of the operating system. An appropriate _Exx GPE handler should be provided. When an operating system with native Hot-Plug support executes the OSHP method, the firmware restores the normal System Interrupt so the interrupts can be serviced by the operating system.

6.8.   Power Budgeting Capability

With the addition of a Hot-Plug capability for add-in cards, the need arises for the system to be
15   capable of properly allocating power to any new devices added to the system. This capability is a separate and distinct function from power management and a basic level of support is required to ensure proper operation of the system. The power budgeting concept puts in place the building blocks that allow devices to interact with systems to achieve these goals. There are many ways in which the system can implement the actual power budgeting capabilities, and as such, they are
20   beyond the scope of this specification.

Implementation of the power budgeting capability is optional for PCI Express devices that are implemented either in a form factor which does not require Hot-Plug support, or that are integrated on the system board. PCI Express form factor specifications may require support for power budgeting. The devices and/or add-in cards are required by PCI Express to remain under the
25   configuration power limit specified in the corresponding electromechanical specification until they have been configured and enabled by the system. The system should guarantee that power has been properly budgeted prior to enabling an add-in card.

6.8.1. System Power Budgeting Process Recommendations

It is recommended that system firmware provide the power budget management agent the following information:

5 ❑ Total system power budget (power supply information).

❑ Total power allocated by system firmware (system board devices).

❑ Total number of slots and the types of slots.

System firmware is responsible for allocating power for all devices on the system board that do not have power budgeting capabilities. The firmware may or may not include standard PCI Express
10 devices that are connected to the standard power rails. When the firmware allocates the power for a device then it must set the SYSTEM_ALLOC bit to 1b to indicate that it has been properly allocated. The power budget manager is responsible for allocating all PCI Express devices including system board devices that have the power budgeting capability and have not been marked allocated. The power budget manager is responsible for determining if hot-plugged devices can be budgeted
15 and enabled in the system.

There are alternate methods which may provide the same functionality, and it is not required that the power budgeting process be implemented in this manner.

6.9. Slot Power Limit Control

PCI Express provides a mechanism for software controlled limiting of the maximum power per slot
20 that PCI Express card/module (associated with that slot) can consume. The key elements of this mechanism are:

❑ Slot Power Limit Value and Scale fields of the Slot Capability register implemented in the Downstream Ports of a Root Complex and a Switch

❑ Slot Power Limit Value and Scale fields of the Device Capability register implemented in the
25 Upstream Ports of a Endpoint, Switch, and PCI Express-PCI Bridge

❑ Set_Slot_Power_Limit Message that conveys the content of the Slot Power Limit Value and Scale fields of the Slot Capability register of the Downstream Port (of a Root Complex or a Switch) to the corresponding Slot Power Limit Value and Scale fields of the Device Capability register in the Upstream Port of the component connected to the same Link

30 Power limits on the platform are typically controlled by the software (for example, platform firmware) that comprehends the specifics of the platform such as:

❑ Partitioning of the platform, including slots for I/O expansion using add-in cards/modules

❑ Power delivery capabilities

❑ Thermal capabilities

35 This software is responsible for correctly programming the Slot Power Limit Value and Scale fields of the Slot Capability registers of the Downstream Ports connected to I/O expansion slots. After the value has been written into the register within the Downstream Port, it is conveyed to the other

component connected to that Port using the Set_Slot_Power_Limit Message (see Section 2.2.8.5). The recipient of the Message must use the value in the Message data payload to limit usage of the power for the entire card/module, unless the card/module will never exceed the lowest value specified in the corresponding electromechanical specification. It is assumed that device driver software associated with card/module will be able (by reading the values of the Slot Power Limit Value and Scale fields of the Device Capability register) to configure hardware of the card/module to guarantee that the card/module will not exceed imposed limit. In the case where the platform imposes a limit that is below minimum needed for adequate operation, the device driver will be able to communicate this discrepancy to higher level configuration software. The software is required to program one of the maximum values specified in the appropriate Add-in Card/Module Power Dissipation Table based on the card/module type, form factor, and capability of the system.

The following rules cover the Slot Power Limit control mechanism:

For Cards/Modules:

❑ Until and unless a Set_Slot_Power_Limit Message is received indicating a Slot Power Limit value greater than the lowest value specified in the electromechanical specification for the card/module's form factor, the card/module must not consume more than the lowest value specified.

❑ A card/module must never consume more power than what was specified in the most recently received Set_Slot_Power_Limit Message.

❑ Endpoint, Switch and PCI Express-PCI Bridge components that are targeted for integration on a card/module where total consumed power is below lowest limit defined for the targeted form factor are permitted to ignore Set_Slot_Power_Limit Messages, and to return a value of 0 in the Slot Power Limit Value and Scale fields of the Device Capability register

❑ Such components still must be able to receive the Set_Slot_Power_Limit Message correctly but simply discard the Message

For Root Complex and Switches which source slots:

❑ A Downstream Port must not transmit a Set_Slot_Power_Limit Message which indicates a limit that is lower than the lowest value specified in the electromechanical specification for the slot's form factor.

✎ IMPLEMENTATION NOTE

Example Device Behavior Based on the Slot Power Limit Control Capability

The following power limit scenarios are examples of how a device must behave based on the Slot Power Limit control capability. The form factor limits are representations, and should not be taken as actual requirements.

Note: Form factor #1 has a maximum power requirement of 40 W and 25 W; form factor #2 has a maximum power requirement of 15 W.

Scenario 1: A Device Consuming 12 W

❑ If the device is plugged into a form factor #1 40 W slot, the Slot Power Limit control mechanism is followed, and the device operates normally.

❑ If the device is plugged into a form factor #1 25 W slot, the Slot Power Limit control mechanism is followed, and the device operates normally.

❑ If the device is plugged into a form factor #2 15 W slot, the Slot Power Limit control mechanism is followed, and the device operates normally.

In all cases, since the device operates normally within all the form factors, it can ignore any of the slot power limit Messages.

Scenario 2: A Device Consuming 18 W

❑ If the device is plugged into a form factor #1 40 W slot, the Slot Power Limit control mechanism is followed, and the device operates normally.

❑ If the device is plugged into a form factor #1 25 W slot, the Slot Power Limit control mechanism is followed, and the device operates normally.

❑ If the device is plugged into a form factor #2 15 W slot, the Slot Power Limit control mechanism is followed, and the device must scale down to 15 W or disable operation. A device that does not scale within any of the power limits for a given form factor will always be disabled in that form factor and should not be used.

In this case, if the device is only to be used in form factor #1, it can ignore any of the slot power limit Messages. To be useful in form factor #2, the device should be capable of scaling to the power limit of form factor #2.

Scenario 3: A Device Consuming 30 W

❑ If the device is plugged into a form factor #1 40 W slot, the Slot Power Limit control mechanism is followed, and the device operates normally.

❑ If the device is plugged into a form factor #1 25 W slot, the Slot Power Limit control mechanism is followed, and the device must scale down to 25 W or disable operation.

❑ If the device is plugged into a form factor #2 15 W slot, the Slot Power Limit control mechanism is followed, and the device must scale down to 15 W or disable operation. A device that does not scale within any of the power limits for a given form factor will always be disabled in that form factor and should not be used.

5   In this case, since the device consumes power above the lowest power limit for a slot, the device must be capable of scaling or disabling to prevent system failures. Operation of devices at power levels that exceed the capabilities of the slots in which they are plugged must be avoided.

IMPLEMENTATION NOTE

Slot Power Limit Control Registers

10   Typically Slot Power Limit registers within Downstream Ports of Root Complex or a Switch Device will be programmed by platform-specific software. Some implementations may use a hardware method for initializing the values of these registers and, therefore, not require software support.

Endpoint, Switch, and PCI Express-PCI Bridge components that are targeted for integration on the card/module where total consumed power is below lowest limit defined for that form factor are

15   allowed to ignore Set_Slot_Power_Limit Messages. Note that PCI Express components that take this implementation approach may not be compatible with potential future defined form factors. Such form factors may impose lower power limits that are below the minimum required by a new add-in card/module based on the existing component.

7

7.   Software Initialization and Configuration

The PCI Express Configuration model supports two configuration space access mechanisms:

❑   PCI compatible configuration mechanism

❑   PCI Express enhanced configuration mechanism

5   The PCI compatible mechanism supports 100% binary compatibility with PCI 2.3 or later aware operating systems and their corresponding bus enumeration and configuration software.

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

7.1.   Configuration Topology

10   To maintain compatibility with PCI software configuration mechanisms, all PCI Express elements have a PCI-compatible configuration space representation. Each PCI Express Link originates from a logical PCI-PCI Bridge and is mapped into configuration space as the secondary bus of this Bridge. The Root Port is a PCI-PCI Bridge structure that originates a PCI Express Link from a PCI Express Root Complex (see Figure 7-1).

15   A PCI Express Switch is represented by multiple PCI-PCI Bridge structures connecting PCI Express Links to an internal logical PCI bus (see Figure 7-2). The Switch Upstream Port is a PCI-PCI Bridge; the secondary bus of this Bridge represents the Switch's internal routing logic. Switch Downstream Ports are PCI-PCI Bridges bridging from the internal bus to buses representing the Downstream PCI Express Links from a PCI Express Switch. Only the PCI-PCI Bridges
20   representing the Switch Downstream Ports may appear on the internal bus. Endpoints, represented by Type 0 configuration space headers, may not appear on the internal bus.

A PCI Express Endpoint is mapped into configuration space as a single logical device with one or more logical functions.
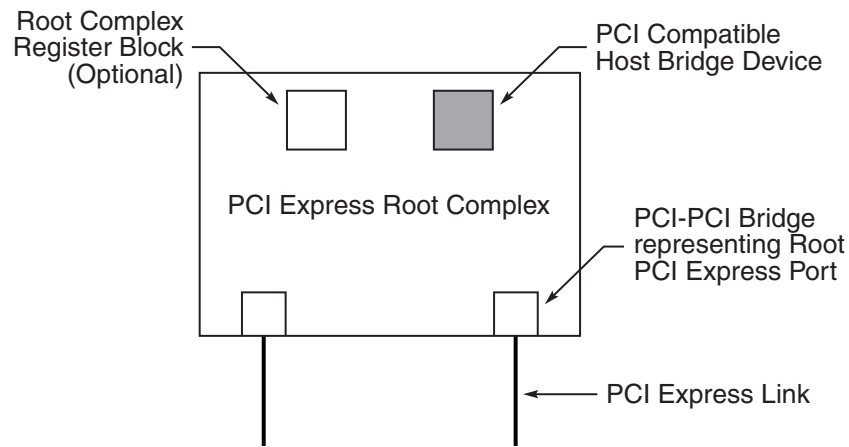
Figure 7-1: PCI Express Root Complex Device Mapping
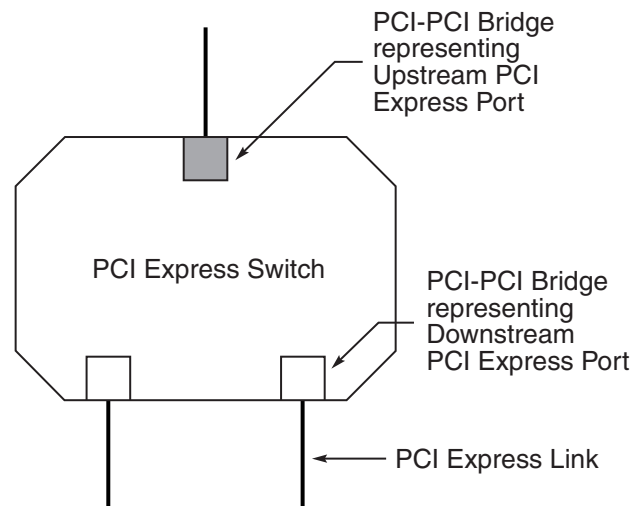
Figure 7-2: PCI Express Switch Device Mapping [60]

7.2.   PCI Express Configuration Mechanisms

PCI Express extends the configuration space to 4096 bytes per device function as compared to 256 bytes allowed by *PCI Local Bus Specification, Revision 2.3*. PCI Express configuration space is divided into a PCI 2.3 compatible region, which consists of the first 256 bytes of a logical device's configuration space, and an extended PCI Express configuration space region which consists of the remaining configuration space (see Figure 7-3). The PCI 2.3 compatible region can be accessed using either the mechanism defined in the PCI 2.3 specification or the enhanced PCI Express

---

[60] Future PCI Express Switches may be implemented as a single Switch device component (without the PCI-PCI Bridges) that is not limited by legacy compatibility requirements imposed by existing PCI software.

configuration access mechanism described later in this section. Accesses made using either access mechanism are equivalent. The extended PCI Express region can only be accessed by using the enhanced PCI Express configuration access mechanism.[61]
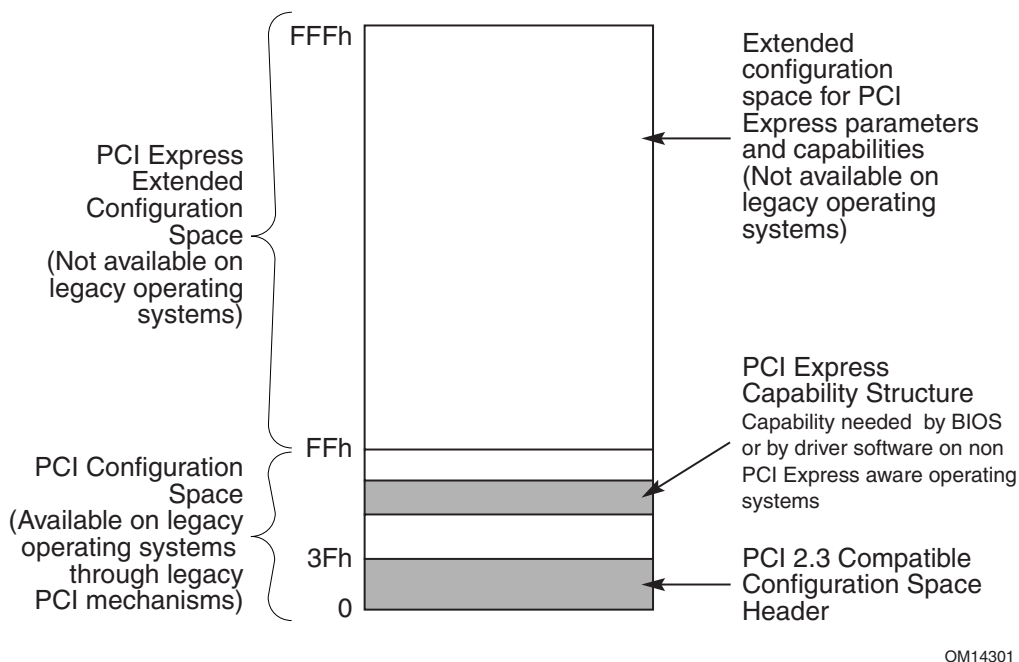

OM14301

Figure 7-3: PCI Express Configuration Space Layout

7.2.1.   PCI 2.3 Compatible Configuration Mechanism

5   The PCI 2.3 compatible PCI Express configuration mechanism supports the PCI configuration space programming model defined in the *PCI Local Bus Specification, Rev. 2.3*. By adhering to this model, systems incorporating PCI Express interfaces remain compliant with conventional PCI bus enumeration and configuration software.

In the same manner as PCI 2.3 devices, PCI Express devices are required to provide a configuration 10   register space for software-driven initialization and configuration. Except for the differences described in this chapter, the PCI Express configuration header space registers are organized to correspond with the format and behavior defined in the PCI 2.3 Specification (Section 6.1).

The PCI 2.3 compatible configuration access mechanism uses the same Request format as the enhanced PCI Express mechanism. For PCI compatible Configuration Requests, the Extended 15   Register Address field must be all zeros.

---

[61] The extended configuration access mechanism operates independently from the mechanism defined in PCI 2.3 for generation of configuration transactions; there is no implied ordering between the two.

7.2.2.   PCI Express Enhanced Configuration Mechanism

The enhanced PCI Express configuration access mechanism utilizes a flat memory-mapped address space to access device configuration registers. In this case, the memory address determines the configuration register accessed and the memory data returns the contents of the addressed register.
5   The mapping from memory address A[27:0] to PCI Express configuration space address is defined in Table 7-1. The base address A[63:28] is allocated in an implementation specific manner and reported by the system firmware to the operating system.

In some processor architectures, it is possible to generate memory space requests that cannot be expressed in a single Configuration Request, for example due to crossing a DW aligned boundary, or
10   because a locked access is used. A Root Complex implementation is not required to support the translation to Configuration Requests of such memory space requests.

Table 7-1: Enhanced Configuration Address Mapping

Memory Address[62]	PCI Express Configuration Space
A[27:20]	Bus Number
A[19:15]	Device Number
A[14:12]	Function Number
A[11:8]	Extended Register Number
A[7:2]	Register Number
A[1:0]	Along with size of the access, used to generate Byte Enables

📝 IMPLEMENTATION NOTE

Generating Configuration Requests

Because Root Complex implementations are not required to support the generation of
15   Configuration Requests from memory space accesses that cross DW boundaries, or that use locked semantics, software should take care not to cause the generation of such requests when using the memory-mapped configuration access mechanism unless it is known that the Root Complex implementation being used will support the translation.

---

[62] This address refers to the byte-level address from a software point of view.

7.2.2.1. Host Bridge Requirements

The PCI Express Host Bridge is required to translate the memory-mapped PCI Express configuration space accesses from the host processor to PCI Express configuration transactions. The use of Host Bridge PCI class code is reserved for backwards compatibility; host Bridge configuration space is opaque to standard PCI Express software and may be implemented in an implementation specific manner that is compatible with PCI Host Bridge Type 0 configuration space.

7.2.2.2. PCI Express Device Requirements

Devices must support an additional 4 bits for decoding configuration register access, i.e., they must decode the Extended Register Address[3:0] field of the Configuration Request header.

7.2.3. Root Complex Register Block

A Root Port may be associated with an optional 4096 byte block of memory mapped registers referred to as the Root Complex Register Block (RCRB). These registers are used in a manner similar to configuration space and can include PCI Express extended capabilities and other implementation specific registers that apply to the Root Complex. The structure of the RCRB is described in Section 7.9.2.

System firmware communicates the base address of the RCRB for each Root Port or internal device in the Root Complex to the operating system. Multiple Root Ports or internal devices may be associated with the same RCRB. The RCRB memory-mapped registers must not reside in the same address space as the memory-mapped configuration space.

A Root Complex implementation is not required to support memory space requests to an RCRB that cross DWORD aligned boundaries or that use locked semantics.

🖎 IMPLEMENTATION NOTE

Accessing Root Complex Register Block

Because Root Complex implementations are not required to support memory space requests to a RCRB that cross DW boundaries, or that use locked semantics, software should take care not to cause the generation of such requests when accessing a RCRB unless it is known that the Root Complex implementation being used will support the request.

7.3.   Configuration Transaction Rules

7.3.1.   Device Number

As in conventional PCI and PCI-X, all PCI Express components are restricted to implementing a single device number on their primary interface (Upstream Port), but may implement up to eight independent functions within that device number. Each internal function is selected based on decoded address information that is provided as part of the address portion of Configuration Request packets.

Switches and Root Complexes must associate only Device 0 with the device attached to the logical bus representing the Link from a Switch Downstream Port or a Root Port. Configuration Requests targeting the Bus Number associated with a Link specifying Device Number 0 are delivered to the device attached to the Link; Configuration Requests specifying all other Device Numbers (1-31) must be terminated by the Switch Downstream Port or the Root Port with an Unsupported Request Completion Status (equivalent to Master Abort in PCI). Devices must not assume that Device Number 0 is associated with their Upstream Port, but must capture their assigned Device Number as discussed in Section 2.2.6.2. Devices must respond to all Type 0 Configuration Read Requests, regardless of the Device Number specified in the Request.

Switches, and components wishing to incorporate more than eight functions at their upstream Port, may implement one or more "virtual switches," represented by multiple Type 1 (PCI-PCI Bridge) configuration space headers as illustrated in Figure 7-2. These virtual switches serve to allow fan-out beyond eight functions. Since Switch Downstream Ports may appear on any Device Number, in this case all address information fields (Bus, Device, and Function Numbers) must be completely decoded to access the correct register. Any configuration access targeting an unimplemented bus, device, or function must return a Completion with Unsupported Request Completion Status.

The following section provides details of the Configuration Space addressing mechanism.

7.3.2.   Configuration Transaction Addressing

PCI Express Configuration Requests use the following addressing fields:

❑ Bus Number – PCI Express maps logical PCI Bus Numbers onto PCI Express Links such that PCI 2.3 compatible configuration software views the configuration space of a PCI Express Hierarchy as a PCI hierarchy including multiple bus segments.

❑ Device Number – Device Number association is discussed in Section 7.3.1.

❑ Function Number – PCI Express also supports multi-function devices using the same discovery mechanism as PCI 2.3.

❑ Extended Register Number and Register Number – Specify the configuration space address of the register being accessed (concatenated such that the Extended Register Number forms the more significant bits).

7.3.3. Configuration Request Routing Rules

For PCI Express Endpoint devices, the following rules apply:

❑ If Configuration Request Type is 1,

- Follow the rules for handling Unsupported Requests

❑ If Configuration Request Type is 0,

- Determine if the Request addresses a valid local configuration space
   - ♦ If so, process the Request
   - ♦ If not, follow rules for handling Unsupported Requests

For Root Ports, Switches, and PCI Express-PCI Bridges, the following rules apply:

❑ Propagation of Configuration Requests from Downstream to Upstream as well as peer-to-peer are not supported

- Configuration Requests are initiated only by the Host Bridge

❑ If Configuration Request Type is 0,

- Determine if the Request addresses a valid local configuration space
   - ♦ If so, process the Request
   - ♦ If not, follow rules for handling Unsupported Requests

❑ If Configuration Request Type is 1, apply the following tests, in sequence, to the Bus Number field:

- If in the case of a PCI Express-PCI Bridge, equal to the bus number assigned to secondary PCI bus or, in the case of a Switch or Root Complex, equal to the bus number and decoded device numbers assigned to one of the Root (Root Complex) or Downstream Ports (Switch),
   - ♦ Transform the Request to Type 0 by changing the value in the Type[4:0] field of the Request (see Table 2-3) – all other fields of the Request remain unchanged
   - ♦ Forward the Request to that Downstream Port (or PCI bus, in the case of a PCI Express-PCI Bridge)

- If not equal to the bus number of any of Downstream Ports or secondary PCI bus, but in the range of bus numbers assigned to one of a Downstream Port or secondary PCI bus,
   - ♦ Forward the Request to that Downstream Port interface without modification

- Else (none of the above)
   - ♦ The Request is invalid - follow the rules for handling Unsupported Requests

❑ PCI Express-PCI Bridges must terminate as Unsupported Requests any Configuration Requests for which the Extended Register Address field is non-zero that are directed towards a PCI bus that does not support Extended Configuration Space.

Note: This type of access is a consequence of a programming error.

Additional rule specific to Root Complexes:

❑ Configuration Requests addressing bus numbers assigned to devices within the Root Complex are processed by the Root Complex

- The assignment of bus numbers to the logical devices within a Root Complex may be done
5   in an implementation specific way.

For all types of devices:

All other configuration space addressing fields are decoded according to the *PCI Local Bus Specification, Revision 2.3*.

7.3.4. PCI Special Cycles

10 PCI Special Cycles (see the *PCI Local Bus Specification, Rev. 2.3* for details) are not directly supported by PCI Express. PCI Special Cycles may be directed to PCI bus segments behind PCI Express-PCI Bridges using Type 1 Configuration Cycles as described in PCI 2.3.

7.4. Configuration Register Types

Configuration register fields are assigned one of the attributes described in Table 7-2. All PCI
15 Express components, with exception of the Root Complex and system-integrated devices, initialize register fields to specified default values. Root Complexes and system-integrated devices initialize register fields as required by the firmware for a particular system implementation.

Table 7-2: Register (and Register Bit-Field) Types

Register Attribute	Description
HwInit	Hardware Initialized: 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 Section 6.6).
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.

Register Attribute	Description
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 or PME Enable) is enabled. In these cases, registers are not initialized or modified by hot, warm, or cold reset (see Section 6.6).
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 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 Section 6.6).
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 Section 6.6).
RsvdP	Reserved and Preserved: Reserved for future RW implementations; software must preserve value read for writes to bits.
RsvdZ	Reserved and Zero: Reserved for future RW1C implementations; software must use 0 for writes to bits.

7.5.   PCI-Compatible Configuration Registers

The first 256 bytes of the PCI Express configuration space form the PCI 2.3 compatibility region. This region completely aliases the PCI 2.3 configuration space of the device/function. Legacy PCI devices may also be accessed via enhanced PCI Express configuration access mechanism without requiring any modifications to the device hardware or device driver software. This section establishes the mapping between PCI 2.3 and PCI Express for format and behavior of PCI 2.3 compatible registers.

All registers and fields not described in this section have the same definition as in PCI 2.3. Layout of the configuration space and format of individual configuration registers are depicted following the little-endian convention used in the *PCI Local Bus Specification, Rev 2.3*.

7.5.1.   Type 0/1 Common Configuration Space

Figure 7-4 details allocation for common register fields of PCI 2.3 Type 0 and Type 1 Configuration Space headers for PCI Express devices.
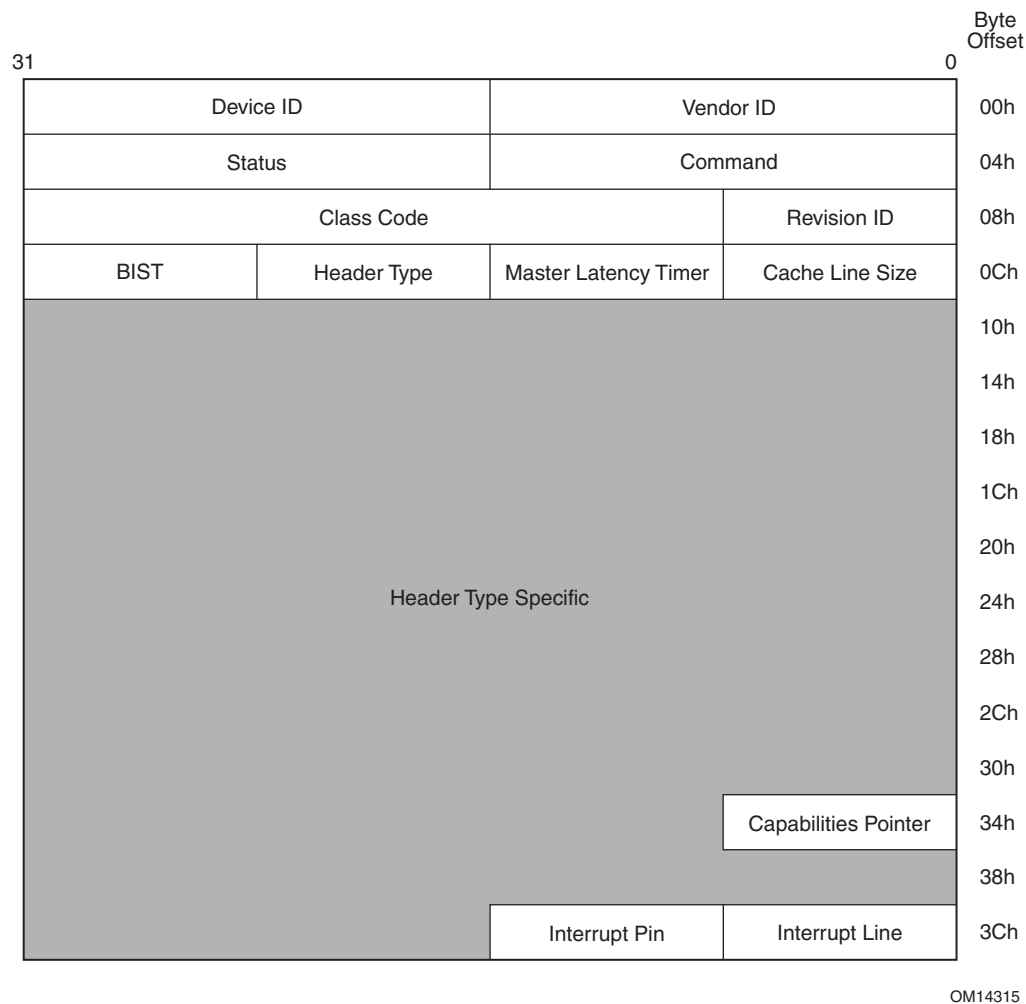

Figure 7-4: Common Configuration Space Header

These registers are defined for both Type 0 and Type 1 Configuration Space headers. The PCI Express-specific interpretation of these registers is defined in this section.

7.5.1.1. Command Register (Offset 04h)

Table 7-3 establishes the mapping between PCI 2.3 and PCI Express for PCI 2.3 configuration space Command register.

Table 7-3: Command Register

Bit Location	Register Description	Attributes
2	**Bus Master Enable** – Controls the ability of a PCI Express Endpoint to issue Memory and I/O Read/Write Requests, and the ability of a Root or Switch Port to forward Memory and I/O Read/Write Requests in the upstream direction *Endpoints:* Disabling this bit prevents a PCI Express agent from issuing any Memory or I/O Requests. Note that as MSI interrupt Messages are in-band memory writes, disabling the bus master enable bit disables MSI interrupt Messages as well. Requests other than Memory or I/O Requests are not controlled by this bit. Default value of this field is 0b. This bit is hardwired to 0b if a device does not generate Memory or I/O Requests. *Root and Switch Ports:* This bit controls forwarding of Memory or I/O Requests by a Switch or Root Port in the upstream direction. When this bit is not set, Memory and I/O Requests received at a Root Port or the downstream side of a Switch Port must be handled as Unsupported Requests (UR), and for Non-Posted Requests a Completion with UR completion status must be returned. This bit does not affect forwarding of Completions in either the upstream or downstream direction. The forwarding of Requests other than Memory or I/O Requests is not controlled by this bit. Default value of this field is 0b.	RW
3	**Special Cycle Enable** – Does not apply to PCI Express. Must be hardwired to 0.	RO
4	**Memory Write and Invalidate** – Does not apply to PCI Express. Must be hardwired to 0.	RO
5	**VGA Palette Snoop** – Does not apply to PCI Express. Must be hardwired to 0.	RO
6	**Parity Error Enable** – See Section 7.5.1.7. Default value of this field is 0.	RW
7	**IDSEL Stepping/Wait Cycle Control** – Does not apply to PCI Express. Must be hardwired to 0.	RO

Bit Location	Register Description	Attributes
8	**SERR Enable** – See Section 7.5.1.7. This bit, when set, enables reporting of Non-fatal and Fatal errors detected by the device to the Root Complex. Note that errors are reported if enabled either through this bit or through the PCI-Express specific bits in the Device Control register (see Section 7.8.4). Default value of this field is 0.	RW
9	**Fast Back-to-Back Transactions Enable** – Does not apply to PCI Express. Must be hardwired to 0.	RO
10	**Interrupt Disable** – Controls the ability of a PCI Express device to generate INTx interrupt Messages. When set, devices are prevented from generating INTx interrupt Messages. Any INTx emulation interrupts already asserted by the device must be deasserted when this bit is set. Note that INTx emulation interrupts forwarded by Root and Switch Ports from devices downstream of the Root or Switch Port are not affected by this bit. Default value of this field is 0.	RW

7.5.1.2. Status Register (Offset 06h)

Table 7-4 establishes the mapping between PCI 2.3 and PCI Express for PCI 2.3 configuration space Status register.

Table 7-4: Status Register

Bit Location	Register Description	Attributes
3	**Interrupt Status** – Indicates that an INTx interrupt Message is pending internally to the device. Note that INTx emulation interrupts forwarded by Root and Switch Ports from devices downstream of the Root or Switch Port are not reflected in this bit. Default value of this field is 0.	RO
4	**Capabilities List** – Indicates the presence of an extended capability list item. 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. Must be hardwired to 0.	RO
7	**Fast Back-to-Back Transactions Capable** – Does not apply to PCI Express. Must be hardwired to 0.	RO

Bit Location	Register Description	Attributes
8	**Master Data Parity Error** – See Section 7.5.1.7. This bit is set by Requestor (Primary Side for Type 1 Configuration Space header device) if its Parity Error Enable bit is set and either of the following two conditions occurs: • Requestor receives a Completion marked poisoned • Requestor poisons a write Request If the Parity Error Enable bit is cleared, this bit is never set. Default value of this field is 0.	RW1C
10:9	**DEVSEL Timing** – Does not apply to PCI Express. Must be hardwired to 0.	RO
11	**Signaled Target Abort** – See Section 7.5.1.7. This bit is set when a device (Primary Side for Type 1 Configuration Space header device for requests completed by the Type 1 header device itself) completes a Request using Completer Abort Completion Status. Default value of this field is 0.	RW1C
12	**Received Target Abort** – See Section 7.5.1.7. This bit is set when a Requestor (Primary Side for Type 1 Configuration Space header device for requests initiated by the Type 1 header device itself) receives a Completion with Completer Abort Completion Status. Default value of this field is 0.	RW1C
13	**Received Master Abort** – See Section 7.5.1.7. This bit is set when a Requestor (Primary Side for Type 1 header Configuration Space header device for requests initiated by the Type 1 header device itself) receives a Completion with Unsupported Request Completion Status. Default value of this field is 0.	RW1C
14	**Signaled System Error** – See Section 7.5.1.7. This bit is set when a device sends an ERR_FATAL or ERR_NONFATAL Message, and the SERR Enable bit in the Command register is 1. Default value of this field is 0.	RW1C
15	**Detected Parity Error** – See Section 7.5.1.7. This bit is set by a device (Primary Side for Type 1 Configuration Space header device) whenever it receives a Poisoned TLP, regardless of the state the Parity Error Enable bit. Default value of this field is 0.	RW1C

7.5.1.3. Cache Line Size Register (Offset 0Ch)

The cache line size register is set by the system firmware and the operating system to system cache line size. However, note that legacy PCI 2.3 software may not always be able to program this field correctly especially in case of Hot-Plug devices. This field is implemented by PCI Express devices as a read-write field for legacy compatibility purposes but has no impact on any PCI Express device functionality.

7.5.1.4. Master Latency Timer Register (Offset 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. This register must be hardwired to 0.

7.5.1.5. Interrupt Line Register (Offset 3Ch)

As in PCI 2.3, 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.

7.5.1.6. Interrupt Pin Register (Offset 3Dh)

The Interrupt Pin is a read-only register that identifies the legacy interrupt Message(s) the device (or device function) uses; refer to Section 6.1 for further details. Valid values are 1, 2, 3, and 4 that map to legacy interrupt Messages for INTA, INTB, INTC, and INTD respectively; a value of 0 indicates that the device uses no legacy interrupt Message(s).

7.5.1.7. Error Registers

The error control/status register bits in the Command and Status registers (see Section 7.5.1.1 and Section 7.5.1.2 respectively) control PCI compatible error reporting for both PCI and PCI Express devices. Mapping of PCI Express errors onto PCI errors is also discussed in Section 6.2.6.1. In addition to the PCI compatible error control and status, PCI Express error reporting may be controlled separately from PCI devices through the PCI Express Capability structure described in Section 7.8. The PCI compatible error control and status register fields do not have any effect on PCI Express error reporting enabled through the PCI Express Capability structure. PCI Express devices may also implement optional advanced error reporting as described in Section 7.10.

For PCI-Express Root Ports represented by a PCI 2.3 Type 1 Configuration Space header:

❑ The primary side error registers (control and status) apply to errors detected on the internal logic associated with the Root Complex.

❑ The secondary side error registers (control and status) apply to errors detected on the Link originating from the Root Port.

For PCI-Express Switch Upstream Ports represented by a PCI 2.3 Type 1 Configuration Space header:

❑ The primary side error registers (control and status) apply to errors detected on the Upstream Link of the Switch.

5 ❑ The secondary side error registers (control and status) apply to errors detected on the internal logic of the Switch.

For PCI-Express Switch Downstream Ports represented by a PCI 2.3 Type 1 Configuration Space header:

❑ The primary side error registers (control and status) apply to errors detected on the internal logic
10 of the Switch.

❑ The secondary side error registers (control and status) apply to errors detected on the Downstream Link originating from the Switch Port.

7.5.2.   Type 0 Configuration Space Header

Figure 7-5 details allocation for register fields of PCI 2.3 Type 0 Configuration Space header for PCI
15 Express devices.

Byte
Offset

Device ID		Vendor ID		00h
Status		Command		04h
Class Code			Revision ID	08h
BIST	Header Type	Master Latency Timer	Cache Line Size	0Ch
Base Address Registers				10h / 14h / 18h / 1Ch / 20h / 24h
Cardbus CIS Pointer				28h
Subsystem ID		Subsystem Vendor ID		2Ch
Expansion ROM Base Address				30h
Reserved			Capabilities Pointer	34h
Reserved				38h
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	3Ch

OM14316

Figure 7-5: Type 0 Configuration Space Header

Section 7.5.1 details the PCI Express-specific registers that are valid for all Configuration Space header types. The PCI Express-specific interpretation of registers specific to Type 0 PCI 2.3 Configuration Space header is defined in this section.

7.5.2.1.   Base Address Registers (Offset 10h - 24h)

A PCI Express Endpoint requesting memory resources through a BAR must set the BAR's Prefetchable bit unless the range contains locations with read side-effects or locations in which the device does not tolerate write merging. It is strongly encouraged that memory-mapped resources be designed as prefetchable whenever possible. PCI Express devices other than legacy Endpoints must support 64-bit addressing for any Base Address register that requests prefetchable memory resources. The minimum memory address range requested by a BAR is 128 bytes.

7.5.2.2. Min_Gnt/Max_Lat Registers (Offset 3Eh/3Fh)

These registers do not apply to PCI Express. They must be read-only and hardwired to 0.

7.5.3.   Type 1 Configuration Space Header

Figure 7-6 details allocation for register fields of PCI 2.3 Type 1 Configuration Space header for Switch and Root Complex virtual PCI Bridges.

31			0	Byte Offset
Device ID		Vendor ID		00h
Status		Command		04h
Class Code			Revision ID	08h
BIST	Header Type	Primary Latency Timer	Cache Line Size	0Ch
Base Address Register 0				10h
Base Address Register 1				14h
Secondary Latency Timer	Subordinate Bus Number	Secondary Bus Number	Primary Bus Number	18h
Secondary Status		I/O Limit	I/O Base	1Ch
Memory Limit		Memory Base		20h
Prefetchable Memory Limit		Prefetchable Memory Base		24h
Prefetchable Base Upper 32 Bits				28h
Prefetchable Limit Upper 32 Bits				2Ch
I/O Limit Upper 16 Bits		I/O Base Upper 16 Bits		30h
Reserved			Capability Pointer	34h
Expansion ROM Base Address				38h
Bridge Control		Interrupt Pin	Interrupt Line	3Ch

OM14317

Figure 7-6: Type 1 Configuration Space Header

Section 7.5.1 details the PCI Express-specific registers that are valid for all Configuration Space header types. The PCI Express-specific interpretation of registers specific to Type 1 PCI 2.3 Configuration Space header is defined in this section. Register interpretations described in this section apply to PCI-PCI Bridge structures representing Switch and Root Ports; other devices such as PCI-Express to PCI/PCI-X Bridges with Type 1 PCI 2.3 Configuration Space headers are not covered by this section.

7.5.3.1.   Base Address Registers (Offset 10h/14h)

A PCI Express Endpoint requesting memory resources through a BAR must set the BAR's Prefetchable bit unless the range contains locations with read side-effects or locations in which the device does not tolerate write merging. It is strongly encouraged that memory-mapped resources be designed as prefetchable whenever possible. PCI Express devices other than legacy Endpoints must support 64-bit addressing for any Base Address register that requests prefetchable memory resources. The minimum memory address range requested by a BAR is 128 bytes.

7.5.3.2.   Secondary Latency Timer (Offset 1Bh)

This register does not apply to PCI Express. It must be read-only and hardwired to 0.

7.5.3.3.   Secondary Status Register (Offset 1Eh)

Table 7-5 establishes the mapping between PCI 2.3 and PCI Express for PCI 2.3 configuration space Secondary Status register.

Table 7-5: Secondary Status Register

Bit Location	Register Description	Attributes
5	**66 MHz Capable** – Does not apply to PCI Express. Must be hardwired to 0.	RO
7	**Fast Back-to-Back Transactions Capable** – Does not apply to PCI Express. Must be hardwired to 0.	RO
8	**Master Data Parity Error** – See Section 7.5.1.7. This bit is set by the Secondary side Requestor if the Parity Error Response bit is set and either of the following two conditions occurs: ● Requestor receives Completion marked poisoned ● Requestor poisons a write Request If the Parity Error Response bit is cleared, this bit is never set. Default value of this field is 0.	RW1C
10:9	**DEVSEL Timing** – Does not apply to PCI Express. Must be hardwired to 0.	RO
11	**Signaled Target Abort** – See Section 7.5.1.7. This bit is set when the Secondary Side for Type 1 Configuration Space header device (for requests completed by the Type 1 header device itself) completes a Request using Completer Abort Completion Status. Default value of this field is 0.	RW1C

Bit Location	Register Description	Attributes
12	**Received Target Abort** – See Section 7.5.1.7.	RW1C
	This bit is set when the Secondary Side for Type 1 Configuration Space header device (for requests initiated by the Type 1 header device itself) receives a Completion with Completer Abort Completion Status.	
	Default value of this field is 0.	
13	**Received Master Abort** – See Section 7.5.1.7.	RW1C
	This bit is set when the Secondary Side for Type 1 Configuration Space header device (for requests initiated by the Type 1 header device itself) receives a Completion with Unsupported Request Completion Status.	
	Default value of this field is 0.	
14	**Received System Error** – See Section 7.5.1.7.	RW1C
	This bit is set when a device sends a ERR_FATAL or ERR_NONFATAL Message, and the SERR Enable bit in the Bridge Control register is 1b.	
	Default value of this field is 0.	
15	**Detected Parity Error** – See Section 7.5.1.7.	RW1C
	This bit is set by the Secondary Side for a Type 1 Configuration Space header device whenever it receives a Poisoned TLP, regardless of the state the Parity Error Response bit.	
	Default value of this field is 0.	

7.5.3.4.   Prefetchable Memory Base/Limit (Offset 24h)

The Prefetchable Memory Base and Prefetchable Memory Limit registers must indicate that 64-bit addresses are supported, as defined in PCI Bridge 1.1.

7.5.3.5. Bridge Control Register (Offset 3Eh)

Table 7-6 establishes the mapping between PCI 2.3 and PCI Express for the PCI 2.3 configuration space Bridge Control register.

<p align="center">Table 7-6: Bridge Control Register</p>

Bit Location	Register Description	Attributes
0	**Parity Error Response Enable** – See Section 7.5.1.7. This bit controls the response to Poisoned TLPs. Default value of this field is 0.	RW
1	**SERR Enable** – See Section 7.5.1.7. This bit controls forwarding of ERR_COR, ERR_NONFATAL and ERR_FATAL from secondary to primary. Default value of this field is 0.	RW
5	**Master Abort Mode** – Does not apply to PCI Express. Must be hardwired to 0.	RO
6	**Secondary Bus Reset** – Setting this bit triggers a hot reset on the corresponding PCI Express Port. Port configuration registers must not be affected, except as required to update Port status. Default value of this field is 0.	RW
7	**Fast Back-to-Back Transactions Enable** – Does not apply to PCI Express. Must be hardwired to 0.	RO
8	**Primary Discard Timer** – Does not apply to PCI Express. Must be hardwired to 0.	RO
9	**Secondary Discard Timer** – Does not apply to PCI Express. Must be hardwired to 0.	RO
10	**Discard Timer Status** – Does not apply to PCI Express. Must be hardwired to 0.	RO
11	**Discard Timer SERR Enable** – Does not apply to PCI Express. Must be hardwired to 0.	RO

7.6. PCI Power Management Capability Structure

This structure is required for all PCI Express devices. Figure 7-7 details allocation of the PCI PM Capability structure register fields in a PCI Express Context. PCI Express devices are required to support D0 and D3 device states (refer to Section 5.1.1); PCI-PCI Bridge structures representing PCI Express Ports as described in Section 7.1 are required to indicate PME Message passing capability due to the in-band nature of PME messaging for PCI Express.

The PME Status bit for the PCI-PCI Bridge structure representing PCI Express Ports, however, is only set when the PCI-PCI Bridge function is itself generating a PME. The PME Status bit is not set when the Bridge is propagating a PME Message but the PCI-PCI Bridge function itself is not internally generating a PME.

Figure 7-8 details allocation of register fields for Power Management Capabilities register; Table 7-7 establishes the mapping between PCI 2.3 and PCI Express for this register.
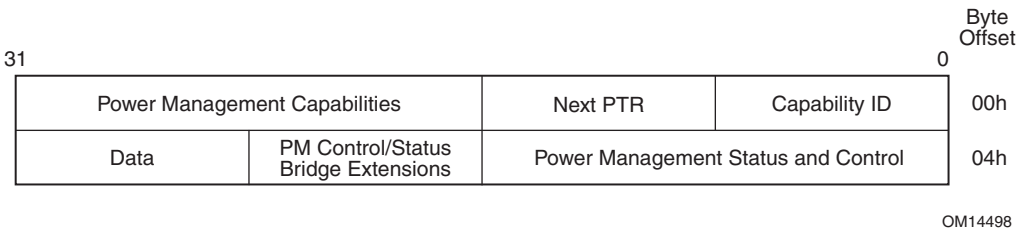
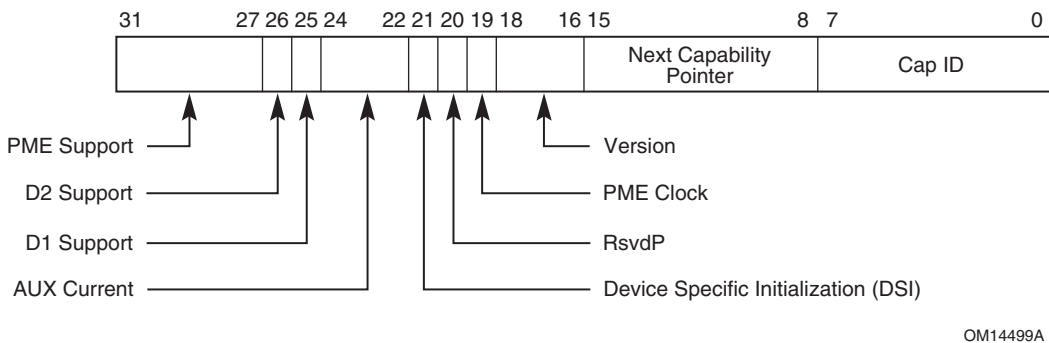**Figure 7-7: PCI Power Management Capability Structure**

Figure 7-8: Power Management Capabilities Register

Table 7-7: Power Management Capabilities Register

Bit Location	Register Description	Attributes
7:0	**Capability ID** – Must be set to 01h	RO
15:8	**Next Capability Pointer**	RO
18:16	**Version** – Set to 02h for this version of the specification.	RO
19	**PME Clock** – Does not apply to PCI Express. Must be hardwired to 0.	RO
21	**Device Specific Initialization**	RO
24:22	**AUX Current**	RO
25	**D1 Support**	RO
26	**D2 Support**	RO
31:27	**PME Support** – For a device, this 5-bit field indicates the power states in which the device may generate a PME. Bits 31, 30, and 27 must be set to 1b for PCI-PCI Bridge structures representing Ports on Root Complexes/Switches to indicate that the Bridge will forward PME Messages.	RO

Figure 7-9 details allocation of register fields for Power Management Status and Control register; Table 7-8 establishes the mapping between PCI 2.3 and PCI Express for this register.
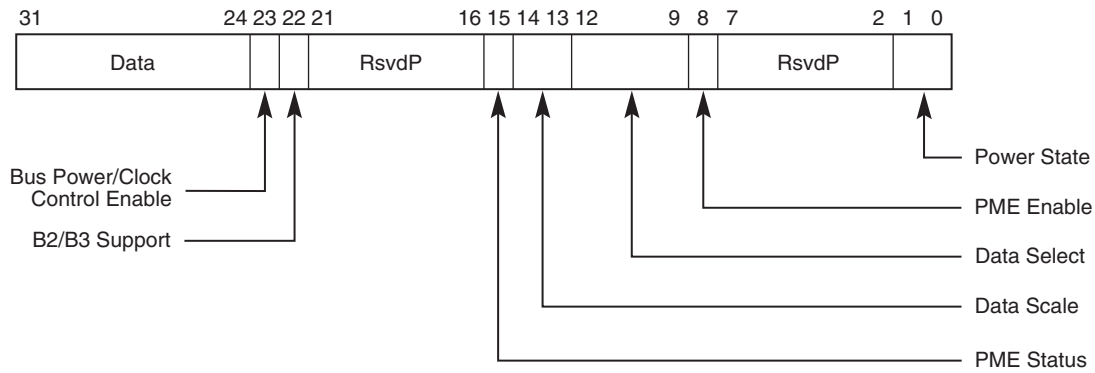
OM14500

Figure 7-9: Power Management Status/Control Register

Table 7-8: Power Management Status/Control Register

Bit Location	Register Description	Attributes
1:0	**Power State**	RW
8	**PME Enable** Note: Devices that consume AUX power must preserve the value of this sticky register when AUX power is available. In such devices, this register value is not modified by hot, warm, or cold reset.	RWS
12:9	**Data Select**	RW
14:13	**Data Scale**	RO
15	**PME Status** Note: Devices that consume AUX power must preserve the value of this sticky register when AUX power is available. In such devices, this register value is not modified by hot, warm, or cold reset.	RW1CS
22	**B2/B3 Support** – Does not apply to PCI Express. Must be hardwired to 0.	RO
23	**Bus Power/Clock Control Enable** – Does not apply to PCI Express. Must be hardwired to 0.	RO
31:24	**Data**	RO

7.7.   MSI Capability Structure

The MSI capability structure is required for all PCI Express devices that are capable of generating interrupts.   This capability is defined in the *PCI Local Bus Specification, Rev. 2.3*.
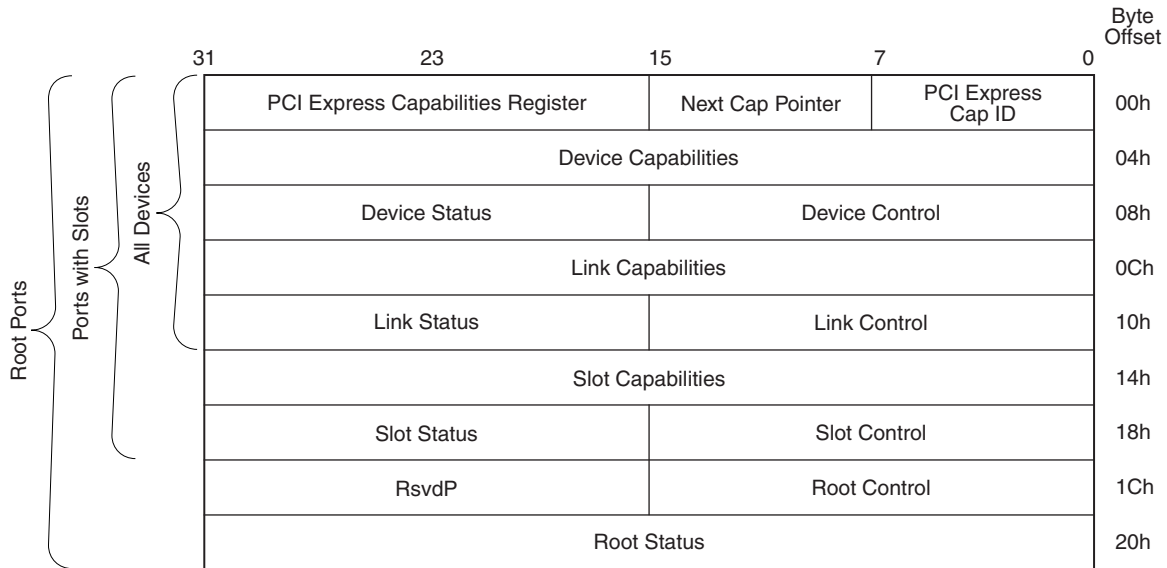
7.8.   PCI Express Capability Structure

5   PCI Express defines a capability structure in PCI 2.3 compatible configuration space (first 256 bytes) as shown in Figure 7-3 for identification of a PCI Express device and indicates support for new PCI Express features. The PCI Express Capability structure is required for PCI Express devices. The capability structure is a mechanism for enabling PCI software transparent features requiring support on legacy operating systems. In addition to identifying a PCI Express device, the PCI Express

10   Capability structure is used to provide access to PCI Express specific Control/Status registers and related Power Management enhancements.

Figure 7-10 details allocation of register fields in the PCI Express Capability structure. The PCI Express Capabilities, Device Capabilities, Device Status/Control, Link Capabilities, and Link Status/Control registers are required for all PCI Express devices. Endpoints are not required to

15   implement registers other than those listed above and terminate the capability structure.

Slot Capabilities and Slot Status/Control registers are required for Switch Downstream and Root Ports if a slot is implemented on the Port. Root Control/Status registers are required for Root Ports. Root Ports must implement the entire PCI Express Capability structure.
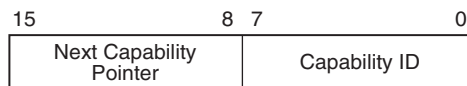
Figure 7-10: PCI Express Capability Structure

7.8.1. PCI Express Capability List Register (Offset 00h)

The PCI Express Capability List register enumerates the PCI Express Capability structure in the PCI 2.3 configuration space capability list. Figure 7-11 details allocation of register fields in the PCI Express Capability List register; Table 7-9 provides the respective bit definitions.

Figure 7-11: PCI Express Capability List Register

Table 7-9: PCI Express Capability List Register

Bit Location	Register Description	Attributes
7:0	**Capability ID** – Indicates the PCI Express Capability structure. This field must return a Capability ID of 10h indicating that this is a PCI Express Capability structure.	RO
15:8	**Next Capability Pointer** – The offset to the next PCI capability structure or 00h if no other items exist in the linked list of capabilities.	RO

7.8.2.   PCI Express Capabilities Register (Offset 02h)

The PCI Express Capabilities register identifies PCI Express device type and associated capabilities. Figure 7-12 details allocation of register fields in the PCI Express Capabilities register; Table 7-10 provides the respective bit definitions.
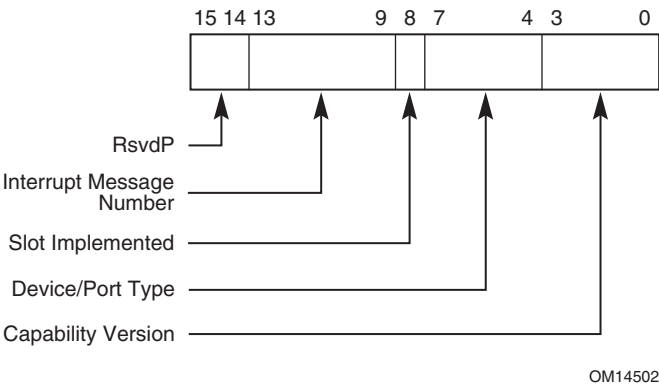

OM14502

Figure 7-12: PCI Express Capabilities Register

Table 7-10: PCI Express Capabilities Register

Bit Location	Register Description	Attributes
3:0	**Capability Version** – Indicates PCI-SIG defined PCI Express capability structure version number. Must be 1h for this specification.	RO

Bit Location	Register Description	Attributes

7:4	**Device/Port Type** – Indicates the type of PCI Express logical device.
	
	Defined encodings are:
	
	   0000b PCI Express Endpoint device
	
	   0001b Legacy PCI Express Endpoint device
	
	   0100b Root Port of PCI Express Root Complex*
	
	   0101b Upstream Port of PCI Express Switch*
	
	   0110b Downstream Port of PCI Express Switch*
	
	   0111b PCI Express-to-PCI/PCI-X Bridge*
	
	   1000b PCI/PCI-X to PCI Express Bridge*
	
	   All other encodings are reserved.
	
	*This value is only valid for devices/functions that implement a Type 01h PCI Configuration Space header.
	
	Native PCI Express Endpoint devices that do not require I/O resources for correct operation indicate a device Type of 0000b; such devices may request I/O resources (through BARs) for legacy boot support but system software is allowed to close requested I/O resources once appropriate services are made available to device specific software for access to device specific resources claimed through memory BARs.
	
	Legacy PCI Express Endpoint devices that require I/O resources claimed through BARs for run-time operations indicate a Device Type of 0001b.
	
	Extended configuration space capabilities, if implemented on legacy PCI Express Endpoint devices, may be ignored by software.
		RO

8	**Slot Implemented** – This bit when set indicates that the PCI Express Link associated with this Port is connected to a slot (as compared to being connected to an integrated component or being disabled). This field is valid for the following PCI Express device/Port Types: • Root Port of PCI Express Root Complex • Downstream Port of PCI Express Switch	HwInit
13:9	**Interrupt Message Number** – If this function is allocated more than one MSI interrupt number, this register is required to contain the offset between the base Message Data and the MSI Message that is generated when any of status bits in either the Slot Status register or the Root Port Status register of this capability structure are set. Hardware is required to update this field so that it is correct if the number of MSI Messages assigned to the device changes.	RO

7.8.3. Device Capabilities Register (Offset 04h)

The Device Capabilities register identifies PCI Express device specific capabilities. Figure 7-13 details allocation of register fields in the Device Capabilities register; Table 7-11 provides the respective bit definitions.
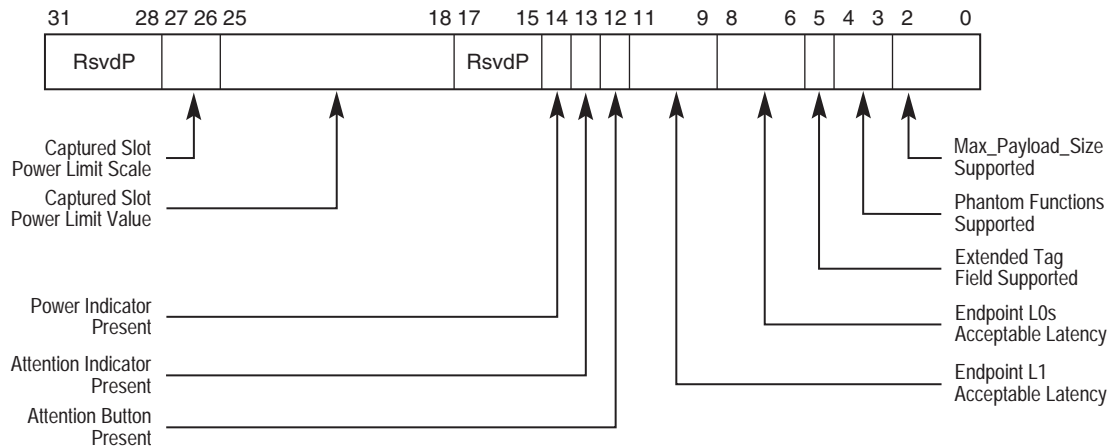
OM14503A

Figure 7-13: Device Capabilities Register

Table 7-11: Device Capabilities Register

Bit Location	Register Description	Attributes
2:0	**Max_Payload_Size Supported** – This field indicates the maximum payload size that the device can support for TLPs. Defined encodings are: 000b 128 bytes max payload size 001b 256 bytes max payload size 010b 512 bytes max payload size 011b 1024 bytes max payload size 100b 2048 bytes max payload size 101b 4096 bytes max payload size 110b Reserved 111b Reserved	RO

Bit Location	Register Description	Attributes
4:3	**Phantom Functions Supported** – This field indicates the support for use of unclaimed function numbers to extend the number of outstanding transactions allowed by logically combining unclaimed function numbers (called Phantom Functions) with the Tag identifier. See Section 2.2.6.2 for description of Tag Extensions. This field indicates the number of most significant bits of the function number portion of Requester ID that are logically combined with the Tag identifier. Defined encodings are: 00b No function number bits used for Phantom Functions; device may implement all function numbers. 01b First most significant bit of function number in Requestor ID used for Phantom Functions; device may implement functions 0-3. Functions 0, 1, 2, and 3 may claim functions 4, 5, 6, and 7 as Phantom Functions respectively. 10b First two most significant bits of function number in Requestor ID used for Phantom Functions; device may implement functions 0-1. Function 0 may claim functions 2, 4, and 6 as Phantom Functions, function 1 may claim functions 3, 5, and 7 as Phantom Functions. 11b All three bits of function number in Requestor ID used for Phantom Functions; device must be a single function 0 device that may claim all other functions as Phantom Functions. Note that Phantom Function support for the Device must be enabled by the corresponding control field in the Device Control register.	RO
5	**Extended Tag Field Supported** – This field indicates the maximum supported size of the Tag field as a Requester. Defined encodings are: 0b 5-bit Tag field supported 1b 8-bit Tag field supported Note that 8-bit Tag field support must be enabled by the corresponding control field in the Device Control register.	RO

Bit Location	Register Description	Attributes
8:6	**Endpoint L0s Acceptable Latency** – This field indicates the acceptable total latency that an Endpoint can withstand due to the transition from L0s state to the L0 state. It is essentially an indirect measure of the Endpoint's internal buffering. Power management software uses the reported L0s Acceptable Latency number to compare against the L0s exit latencies reported by all components comprising the data path from this Endpoint to the Root Complex Root Port to determine whether ASPM L0s entry can be used with no loss of performance. Defined encodings are: 000b Less than 64 ns 001b 64 ns to less than 128 ns 010b 128 ns to less than 256 ns 011b 256 ns to less than 512 ns 100b 512 ns to less than 1 µs 101b 1 µs to less than 2 µs 110b 2 µs-4 µs 111b More than 4 µs	RO
11:9	**Endpoint L1 Acceptable Latency** – This field indicates the acceptable latency that an Endpoint can withstand due to the transition from L1 state to the L0 state. It is essentially an indirect measure of the Endpoint's internal buffering. Power management software uses the reported L1 Acceptable Latency number to compare against the L1 Exit Latencies reported (see below) by all components comprising the data path from this Endpoint to the Root Complex Root Port to determine whether ASPM L1 entry can be used with no loss of performance. Defined encodings are: 000b Less than 1µs 001b 1 µs to less than 2 µs 010b 2 µs to less than 4 µs 011b 4 µs to less than 8 µs 100b 8 µs to less than 16 µs 101b 16 µs to less than 32 µs 110b 32 µs-64 µs 111b More than 64 µs	RO

Bit Location	Register Description	Attributes
12	**Attention Button Present** – This bit when set indicates that an Attention Button is implemented on the card or module. This bit is valid for the following PCI Express device Types: • PCI Express Endpoint device • Legacy PCI Express Endpoint device • Upstream Port of PCI Express Switch • PCI Express-to-PCI/PCI-X Bridge	RO
13	**Attention Indicator Present** – This bit, when set, indicates that an Attention Indicator is implemented on the card or module. This bit is valid for the following PCI Express device Types: • PCI Express Endpoint device • Legacy PCI Express Endpoint device • Upstream Port of PCI Express Switch • PCI Express-to-PCI/PCI-X Bridge	RO
14	**Power Indicator Present** – This bit, when set, indicates that a Power Indicator is implemented on the card or module. This bit is valid for the following PCI Express device Types: • PCI Express Endpoint device • Legacy PCI Express Endpoint device • Upstream Port of PCI Express Switch • PCI Express-to-PCI/PCI-X Bridge	RO
25:18	**Captured Slot Power Limit Value** (Upstream Ports only) – In combination with the Slot Power Limit Scale value, specifies the upper limit on power supplied by slot. Power limit (in Watts) calculated by multiplying the value in this field by the value in the Slot Power Limit Scale field. This value is set by the Set_Slot_Power_Limit Message or hardwired to 0000 0000b (see Section 6.9). The default value is 0000 0000b.	RO
27:26	**Captured Slot Power Limit Scale** (Upstream Ports only) – Specifies the scale used for the Slot Power Limit Value. Range of Values: 00b = 1.0x 01b = 0.1x 10b = 0.01x 11b = 0.001x This value is set by the Set_Slot_Power_Limit Message or hardwired to 00b (see Section 6.9). The default value is all 00b.	RO

7.8.4. Device Control Register (Offset 08h)

The Device Control register controls PCI Express device specific parameters. Figure 7-14 details allocation of register fields in the Device Control register; Table 7-12 provides the respective bit definitions.

OM14504A
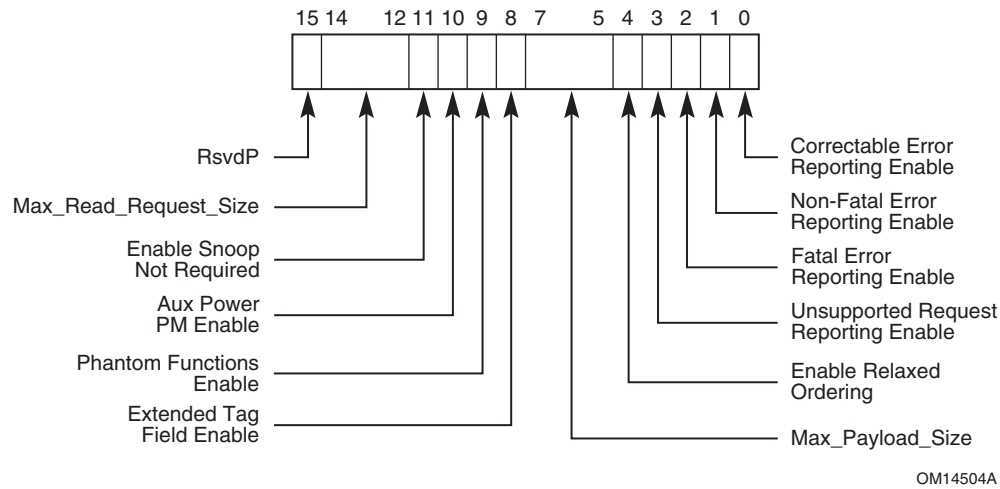
Figure 7-14: Device Control Register

Table 7-12: Device Control Register

Bit Location	Register Description	Attributes
0	**Correctable Error Reporting Enable** – This bit controls reporting of correctable errors. Refer to Section 6.2 for further details. For a multi-function device, this bit controls error reporting for each function from point-of-view of the respective function. For a Root Port, the reporting of correctable errors is internal to the root. No external ERR_COR Message is generated. Default value of this field is 0.	RW
1	**Non-Fatal Error Reporting Enable** – This bit controls reporting of Non-fatal errors. Refer to Section 6.2 for further details. For a multi-function device, this bit controls error reporting for each function from point-of-view of the respective function. For a Root Port, the reporting of Non-fatal errors is internal to the root. No external ERR_NONFATAL Message is generated. Default value of this field is 0.	RW

Bit Location	Register Description	Attributes
2	**Fatal Error Reporting Enable** – This bit controls reporting of Fatal errors. Refer to Section 6.2 for further details. For a multi-function device, this bit controls error reporting for each function from point-of-view of the respective function. For a Root Port, the reporting of Fatal errors is internal to the root. No external ERR_FATAL Message is generated. Default value of this field is 0.	RW
3	**Unsupported Request Reporting Enable** – This bit enables reporting of Unsupported Requests when set. Refer to Section 6.2 for further details. For a multi-function device, this bit controls error reporting for each function from point-of-view of the respective function. Note that the reporting of error Messages (ERR_COR, ERR_NONFATAL, ERR_FATAL) received by Root Port is controlled exclusively by Root Control register described in Section 7.8.12. Default value of this field is 0.	RW
4	**Enable Relaxed Ordering** – If this bit is set, the device is permitted to set the Relaxed Ordering bit in the Attributes field of transactions it initiates that do not require strong write ordering (see Sections 2.2.6.4 and 2.4). Default value of this bit is 1. This bit may be hardwired to 0 if a device never sets the Relaxed Ordering attribute in transactions it initiates as a requester.	RW
7:5	**Max_Payload_Size** – This field sets maximum TLP payload size for the device. As a Receiver, the device must handle TLPs as large as the set value; as Transmitter, the device must not generate TLPs exceeding the set value. Permissible values that can be programmed are indicated by the Max_Payload_Size Supported in the Device Capabilities register (refer to Section 7.8.3). Defined encodings for this field are: 000b 128 bytes max payload size 001b 256 bytes max payload size 010b 512 bytes max payload size 011b 1024 bytes max payload size 100b 2048 bytes max payload size 101b 4096 bytes max payload size 110b Reserved 111b Reserved Default value of this field is 000b.	RW

Bit Location	Register Description	Attributes
8	**Extended Tag Field Enable** – When set, this bit enables a device to use an 8-bit Tag field as a requester. If the bit is cleared, the device is restricted to a 5-bit Tag field. See Section 2.2.6.2 for description of Tag extensions. Default value of this field is 0. Devices that do not implement this capability hardwire this bit to 0.	RW
9	**Phantom Functions Enable** – When set, this bit enables a device to use unclaimed functions as Phantom Functions to extend the number of outstanding transaction identifiers. If the bit is cleared, the device is not allowed to use Phantom Functions. See Section 2.2.6.2 for a description of Tag extensions. Default value of this field is 0. Devices that do not implement this capability hardwire this bit to 0.	RW
10	**Auxiliary (AUX) Power PM Enable** – This bit when set enables a device to draw AUX power independent of PME AUX power. Devices that require AUX power on legacy operating systems should continue to indicate PME AUX power requirements. AUX power is allocated as requested in the AUX_Current field of the Power Management Capabilities register (PMC), independent of the PME_En bit in the Power Management Control/Status register (PMCSR) (see Chapter 5). For multi-function devices, a component is allowed to draw AUX power if at least one of the functions has this bit set. Note: Devices that consume AUX power must preserve the value of this sticky register when AUX power is available. In such devices, this register value is not modified by hot, warm, or cold reset.Devices that do not implement this capability hardwire this bit to 0.	RWS
11	**Enable No Snoop** – If this bit is set to 1, the device is permitted to set the No Snoop bit in the Requester Attributes of transactions it initiates that do not require hardware enforced cache coherency (see Section 2.2.6.5). Note that setting this bit to 1 should not cause a device to blindly set the No Snoop attribute on all transactions that it initiates. Even when this bit is set to 1, a device may only set the No Snoop attribute on a transaction when it can guarantee that the address of the transaction is not stored in any cache in the system. Default value of this bit is 1. This bit may be hardwired to 0 if a device never sets the No Snoop attribute in transactions it initiates.	RW

Bit Location	Register Description	Attributes
14:12	**Max_Read_Request_Size** – This field sets the maximum Read Request size for the Device as a Requester. The Device must not generate read requests with size exceeding the set value. Defined encodings for this field are:	RW
	000b 128 bytes max read request size	
	001b 256 bytes max read request size	
	010b 512 bytes max read request size	
	011b 1024 bytes max read request size	
	100b 2048 bytes max read request size	
	101b 4096 bytes max read request size	
	110b Reserved	
	111b Reserved	
	Devices that do not generate Read Request larger than 128 bytes are permitted to implement this field as Read Only (RO) with a value of 000b.	
	Default value of this field is 010b.	

📝 IMPLEMENTATION NOTE

Use of Max_Payload_Size

The Max_Payload_Size mechanism allows software to control the maximum payload in packets sent by Endpoints to balance latency versus bandwidth trade-offs, particularly for isochronous traffic.

⁵ If software chooses to program the Max_Payload_Size of various system elements to non-default values, it must take care to ensure that each packet does not exceed the Max_Payload_Size parameter of any system element along the packet's path. Otherwise, the packet will be rejected by the system element whose Max_Payload_Size parameter is too small.

Discussion of specific algorithms used to configure Max_Payload_Size so that this requirement is
¹⁰ met is beyond the scope of this specification, but software should base its algorithm upon factors such as the following:

❑ the Max_Payload_Size capability of each system element within a hierarchy

❑ awareness of when system elements are added or removed through Hot-Plug operations

❑ knowing which system elements send packets to each other, what type of traffic is carried, what
¹⁵   type of transactions are used, or if packet sizes are constrained by other mechanisms

For the case of firmware that configures system elements in preparation for running legacy operating system environments, the firmware may need to avoid programming a Max_Payload_Size above the default of 128 bytes, which is the minimum supported by Endpoints.

For example, if the operating system environment does not comprehend PCI Express, firmware
²⁰ probably should not program a non-default Max_Payload_Size for a hierarchy that supports Hot-

Plug operations. Otherwise, if no software is present to manage Max_Payload_Size settings when a new element is added, improper operation may result. Note that a newly added element may not even support a Max_Payload_Size setting as large as the rest of the hierarchy, in which case software may need to deny enabling the new element or reduce the Max_Payload_Size settings of other 5 elements.

🖐 IMPLEMENTATION NOTE

Use of Max_Read_Request_Size

The Max_Read_Request_Size mechanism allows improved control of bandwidth allocation in systems where quality of service (QoS) is important for the target applications. For example, an 10 arbitration scheme based on counting requests (and not the sizes of those requests) provides imprecise bandwidth allocation when some Requesters use much larger sizes than others. The Max_Read_Request_Size mechanism can be used to force more uniform allocation of bandwidth, by restricting the upper size of read requests.

7.8.5.   Device Status Register (Offset 0Ah)

15 The Device Status register provides information about PCI Express device specific parameters. Figure 7-15 details allocation of register fields in the Device Status register; Table 7-13 provides the respective bit definitions.
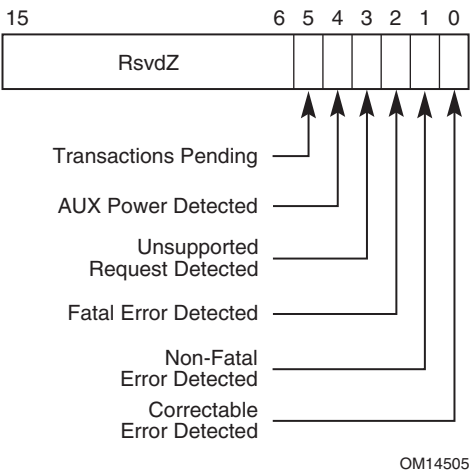
Figure 7-15: Device Status Register

Table 7-13: Device Status Register

Bit Location	Register Description	Attributes
0	**Correctable Error Detected** – This bit indicates status of correctable errors detected. Errors are logged in this register regardless of whether error reporting is enabled or not in the Device Control register. For a multi-function device, each function indicates status of errors as perceived by the respective function. For devices supporting Advanced Error Handling, errors are logged in this register regardless of the settings of the correctable error mask register. Default value of this field is 0.	RW1C
1	**Non-Fatal Error Detected** – This bit indicates status of Non-fatal errors detected. Errors are logged in this register regardless of whether error reporting is enabled or not in the Device Control register. For a multi-function device, each function indicates status of errors as perceived by the respective function. For devices supporting Advanced Error Handling, errors are logged in this register regardless of the settings of the correctable error mask register. Default value of this field is 0.	RW1C
2	**Fatal Error Detected** – This bit indicates status of Fatal errors detected. Errors are logged in this register regardless of whether error reporting is enabled or not in the Device Control register. For a multi-function device, each function indicates status of errors as perceived by the respective function. For devices supporting Advanced Error Handling, errors are logged in this register regardless of the settings of the correctable error mask register. Default value of this field is 0.	RW1C
3	**Unsupported Request Detected** – This bit indicates that the device received an Unsupported Request. Errors are logged in this register regardless of whether error reporting is enabled or not in the Device Control register. For a multi-function device, each function indicates status of errors as perceived by the respective function. Default value of this field is 0.	RW1C
4	**AUX Power Detected** – Devices that require AUX power report this bit as set if AUX power is detected by the device.	RO

Bit Location	Register Description	Attributes
5	**Transactions Pending** – *Endpoints:* This bit when set indicates that the device has issued Non-Posted Requests which have not been completed. A device reports this bit cleared only when all outstanding Non-Posted Requests have completed or have been terminated by the Completion Timeout mechanism. *Root and Switch Ports:* This bit when set indicates that a Port has issued Non-Posted Requests on its own behalf (using the Port's own Requester ID) which have not been completed. The Port reports this bit cleared only when all such outstanding Non-Posted Requests have completed or have been terminated by the Completion Timeout mechanism. Note that Root and Switch Ports implementing only the functionality required by this document do not issue Non-Posted Requests on their own behalf, and therefore are not subject to this case. Root and Switch Ports that do not issue Non-Posted Requests on their own behalf hardwire this bit to 0b.	RO

7.8.6.   Link Capabilities Register (Offset 0Ch)

The Link Capabilities register identifies PCI Express Link specific capabilities. Figure 7-16 details allocation of register fields in the Link Capabilities register; Table 7-14 provides the respective bit definitions.
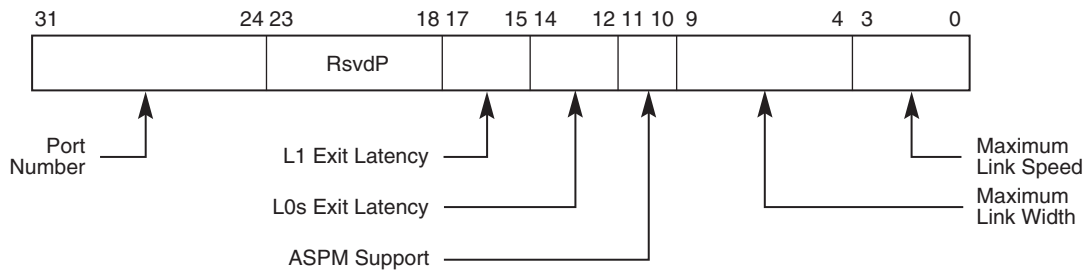

Figure 7-16: Link Capabilities Register

Table 7-14: Link Capabilities Register

Bit Location	Register Description	Attributes
3:0	**Maximum Link Speed** – This field indicates the maximum Link speed of the given PCI Express Link. Defined encodings are: 0001b 2.5 Gb/s Link All other encodings are reserved.	RO
9:4	**Maximum Link Width** – This field indicates the maximum width of the given PCI Express Link. Defined encodings are: 000000b Reserved 000001b x1 000010b x2 000100b x4 001000b x8 001100b x12 010000b x16 100000b x32	RO
11:10	**Active State Power Management (ASPM) Support** – This field indicates the level of ASPM supported on the given PCI Express Link. Defined encodings are: 00b Reserved 01b L0s Entry Supported 10b Reserved 11b L0s and L1 Supported	RO

Bit Location	Register Description	Attributes
14:12	**L0s Exit Latency** – This field indicates the L0s exit latency for the given PCI Express Link. The value reported indicates the length of time this Port requires to complete transition from L0s to L0. Defined encodings are: 000b Less than 64 ns 001b 64 ns to less than 128 ns 010b 128 ns to less than 256 ns 011b 256 ns to less than 512 ns 100b 512 ns to less than 1 µs 101b 1 µs to less than 2 µs 110b 2 µs-4 µs 111b More than 4 µs Note that exit latencies may be influenced by PCI Express reference clock configuration depending upon whether a component uses a common or separate reference clock.	RO
17:15	**L1 Exit Latency** – This field indicates the L1 exit latency for the given PCI Express Link. The value reported indicates the length of time this Port requires to complete transition from L1 to L0. Defined encodings are: 000b Less than 1µs 001b 1 µs to less than 2 µs 010b 2 µs to less than 4 µs 011b 4 µs to less than 8 µs 100b 8 µs to less than 16 µs 101b 16 µs to less than 32 µs 110b 32 µs-64 µs 111b More than 64 µs Note that exit latencies may be influenced by PCI Express reference clock configuration depending upon whether a component uses a common or separate reference clock.	RO
31:24	**Port Number** – This field indicates the PCI Express Port number for the given PCI Express Link.	HwInit

If L1 state is not supported for ASPM (as reported in the ASPM Support field), then the L1 Exit latency field is ignored.

7.8.7.   Link Control Register (Offset 10h)

The Link Control register controls PCI Express Link specific parameters. Figure 7-17 details allocation of register fields in the Link Control register; Table 7-15 provides the respective bit definitions.

OM14507A
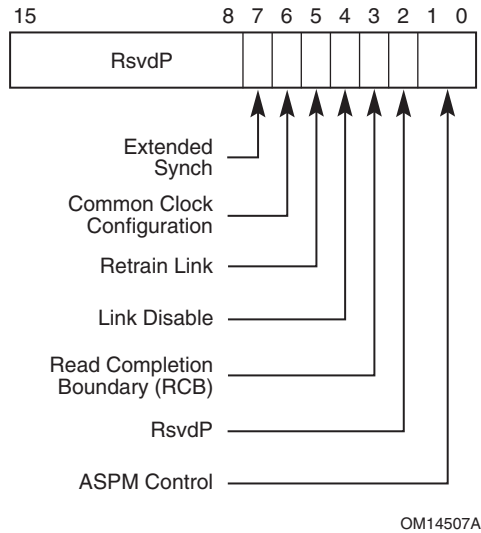
Figure 7-17: Link Control Register

Table 7-15: Link Control Register

Bit Location	Register Description	Attributes
1:0	**Active State Power Management (ASPM) Control** – This field controls the level of ASPM supported on the given PCI Express Link. Defined encodings are: 00b Disabled 01b L0s Entry Enabled 10b L1 Entry Enabled 11b L0s and L1 Entry Enabled Note: "L0s Entry Enabled" indicates the Transmitter entering L0s is supported. The Receiver must be capable of entering L0s even when the field is disabled (00b). Default value of this field is (00b) or (01b) depending on form factor.	RW

Bit Location	Register Description	Attributes
3	*Root Ports:*	*Root Ports:*
	Read Completion Boundary (RCB) – Indicates the RCB value for the Root Port. Refer to Section 2.3.1.1 for the definition of the parameter RCB.	RO
	Defined encodings are:	
	0b 64 byte	
	1b 128 byte	
	This field is hardwired for a Root Port and returns its RCB support capabilities.	
	Endpoints:	*Endpoints:*
	Read Completion Boundary (RCB) – May be set by configuration software to indicate the RCB value of the Root Port upstream from the Endpoint. Refer to Section 2.3.1.1 for the definition of the parameter RCB.	RW
	Defined encodings are:	
	0b 64 byte	
	1b 128 byte	
	Devices that do not implement this feature must hardwire the field to 0b.	
	Switch Ports:	*Switch Ports:*
	Not applicable – must hardwire the field to 0b.	RO
4	**Link Disable** – This bit disables the Link when set to 1b; this field is not applicable and reserved for Endpoint devices and Upstream Ports of Switches.	RW
	Writes to this bit are immediately reflected in the value read from the bit, regardless of actual Link state.	
	Default value of this field is 0b.	
5	**Retrain Link** – A write of 1b to this bit initiates Link retraining by directing the Physical Layer LTSSM to the Recovery state. Reads of this bit always return 0b.	RW
	This field is not applicable and is reserved for Endpoint devices and Upstream Ports of Switches.	
	This bit always returns 0b when read.	

Bit Location	Register Description	Attributes
6	**Common Clock Configuration** – This bit when set indicates that this component and the component at the opposite end of this Link are operating with a distributed common reference clock. A value of 0b indicates that this component and the component at the opposite end of this Link are operating with asynchronous reference clock. Components utilize this common clock configuration information to report the correct L0s and L1 Exit Latencies. After changing the value in this bit in both components on a Link, software must trigger the Link to retrain by writing a 1b to the Retrain Link bit. Default value of this field is 0b.	RW
7	**Extended Synch** – This bit when set forces the transmission of 4096 FTS ordered sets in the L0s state followed by a single SKP ordered set (see Section 4.2.4.3) prior to entering the L0 state, and the transmission of 1024 TS1 ordered sets in the L1 state prior to entering the Recovery state. This mode provides external devices (e.g., logic analyzers) monitoring the Link time to achieve bit and Symbol lock before the Link enters the L0 or Recovery states and resumes communication. Default value for this bit is 0b.	RW

IMPLEMENTATION NOTE

Use of the Slot Clock Configuration and Common Clock Configuration Bits

In order to accurately determine the common clocking configuration of components on opposite ends of a Link that crosses a connector, there are two pieces of information that are required. The following description defines these requirements.

The first necessary piece of information is whether the Port that connects to the system side of the connector uses a clock that has a common source and therefore constant phase relationship to the clock signal provided on the connector. This information is provided by the system side component through a hardware initialized bit (Slot Clock Configuration) in its Link Status register. Note that some electromechanical form factor specifications may require the Port that connects to the system side of the connector use a clock that has a common source to the clock signal provided on the connector.

The second necessary piece of information is whether the device on the option card uses the clock supplied on the connector or one generated locally on the card. It is the card design and layout that will determine whether or not the Endpoint component is connected to the clock source provided by the slot connector. An Endpoint going onto this card should have some hardware initialized method for the card design/designer to indicate the configuration used for this particular card design. This information is reported by the Endpoint device in bit 12 (Slot Clock Configuration) of

its Link Status register. Note that some electromechanical form factor specifications may require the Port that connects to the card side of the connector use the clock signal provided on the connector.

System firmware or software will read this value from the components on both ends of a physical Link. If both devices report the use of a common clock connection this firmware/software will program bit 6 (Common Clock Configuration) of the Link Control register to a one on both components connected to the Link. Each component uses this bit to determine the length of time required to re-synch its Receiver to the opposing component's Transmitter when exiting L0s.

This value is reported as a time value in bits 12-14 of the Link Capabilities register (offset 0Ch) and is sent to the opposing Transmitter as part if the initialization process as N_FTS. Components would be expected to require much longer synch times without common clocking and would therefore report a longer L0s exit latency in bits 12-14 of the Link Capabilities register and would send a larger number for N_FTS during training. This forces a requirement that whatever software changes this bit should force a Link retrain in order to get the correct N-FTS set for the Receivers at both ends of the Link.

7.8.8. Link Status Register (Offset 12h)

The Link Status register provides information about PCI Express Link specific parameters. Figure 7-18 details allocation of register fields in the Link Status register; Table 7-16 provides the respective bit definitions.
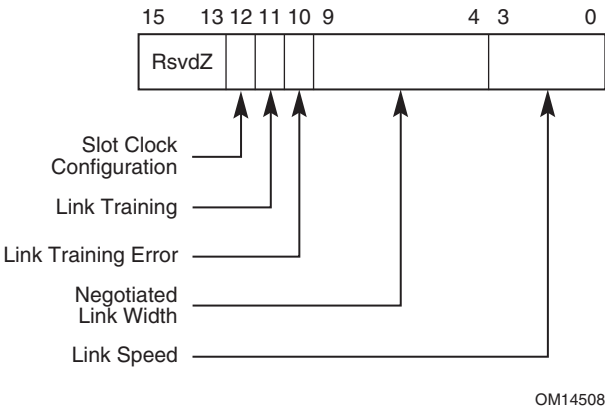

Figure 7-18: Link Status Register

Table 7-16: Link Status Register

Bit Location	Register Description	Attributes
3:0	**Link Speed** – This field indicates the negotiated Link speed of the given PCI Express Link. Defined encodings are: 0001b 2.5 Gb/s PCI Express Link All other encodings are reserved.	RO

9:4	**Negotiated Link Width** – This field indicates the negotiated width of the given PCI Express Link.
	
	Defined encodings are:
	
	  000001b X1
	
	  000010b X2
	
	  000100b X4
	
	  001000b X8
	
	  001100b X12
	
	  010000b X16
	
	  100000b X32
	
	All other encodings are reserved.
		RO

10	**Training Error** – This read-only bit indicates that a Link training error occurred. This field is not applicable and reserved for Endpoint devices and Upstream Ports of Switches. This bit is cleared by hardware upon successful training of the Link to the L0 Link state.	RO
11	**Link Training** – This read-only bit indicates that Link training is in progress (Physical Layer LTSSM in Configuration or Recovery state) or that 1b was written to the Retrain Link bit but Link training has not yet begun. Hardware clears this bit once Link training is complete. This field is not applicable and reserved for Endpoint devices and Upstream Ports of Switches.	RO
12	**Slot Clock Configuration** – This bit indicates that the component uses the same physical reference clock that the platform provides on the connector. If the device uses an independent clock irrespective of the presence of a reference on the connector, this bit must be clear.	HwInit

7.8.9. Slot Capabilities Register (Offset 14h)

The Slot Capabilities register identifies PCI Express slot specific capabilities. Figure 7-19 details allocation of register fields in the Slot Capabilities register; Table 7-17 provides the respective bit definitions.
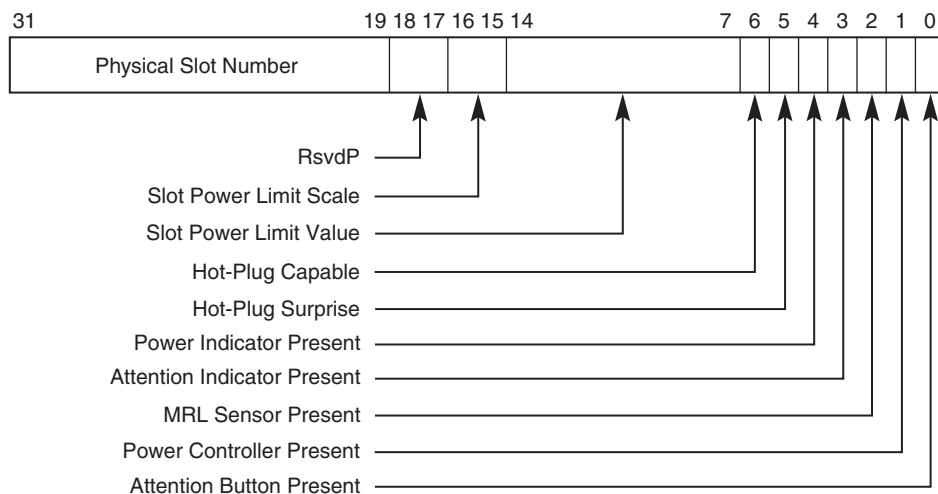
OM14509A

Figure 7-19: Slot Capabilities Register

Table 7-17: Slot Capabilities Register

Bit Location	Register Description	Attributes
0	**Attention Button Present** – This bit when set indicates that an Attention Button is implemented on the chassis for this slot.	HwInit
1	**Power Controller Present** – This bit when set indicates that a Power Controller is implemented for this slot.	HwInit
2	**MRL Sensor Present** – This bit when set indicates that an MRL Sensor is implemented on the chassis for this slot.	HwInit
3	**Attention Indicator Present** – This bit when set indicates that an Attention Indicator is implemented on the chassis for this slot.	HwInit
4	**Power Indicator Present** – This bit when set indicates that a Power Indicator is implemented on the chassis for this slot.	HwInit
5	**Hot-Plug Surprise** – This bit when set indicates that a device present in this slot might be removed from the system without any prior notification.	HwInit
6	**Hot-Plug Capable** – This bit when set indicates that this slot is capable of supporting Hot-Plug operations.	HwInit

Bit Location	Register Description	Attributes
14:7	**Slot Power Limit Value** – In combination with the Slot Power Limit Scale value, specifies the upper limit on power supplied by slot (see Section 6.9). Power limit (in Watts) calculated by multiplying the value in this field by the value in the Slot Power Limit Scale field. This register must be implemented if the Slot Implemented bit is set. Writes to this register also cause the Port to send the Set_Slot_Power_Limit Message. The default value prior to hardware/firmware initialization is 0000 0000b.	HwInit

16:15	**Slot Power Limit Scale** – Specifies the scale used for the Slot Power Limit Value (see Section 6.9).
	
	Range of Values:
	
	  00b = 1.0x
	
	  01b = 0.1x
	
	  10b = 0.01x
	
	  11b = 0.001x
	
	This register must be implemented if the Slot Implemented bit is set.
	
	Writes to this register also cause the Port to send the Set_Slot_Power_Limit Message.
	
	The default value prior to hardware/firmware initialization is 00b.
		HwInit

31:19	**Physical Slot Number** – This hardware initialized field indicates the physical slot number attached to this Port. This field must be hardware initialized to a value that assigns a slot number that is globally unique within the chassis. These registers should be initialized to 0 for Ports connected to devices that are either integrated on the system board or integrated within the same silicon as the Switch device or Root Port.	HwInit

7.8.10. Slot Control Register (Offset 18h)

The Slot Control register controls PCI Express Slot specific parameters. Figure 7-20 details allocation of register fields in the Slot Control register; Table 7-18 provides the respective bit definitions. Register fields for control parameters not implemented by the device have the RsvdP attribute.
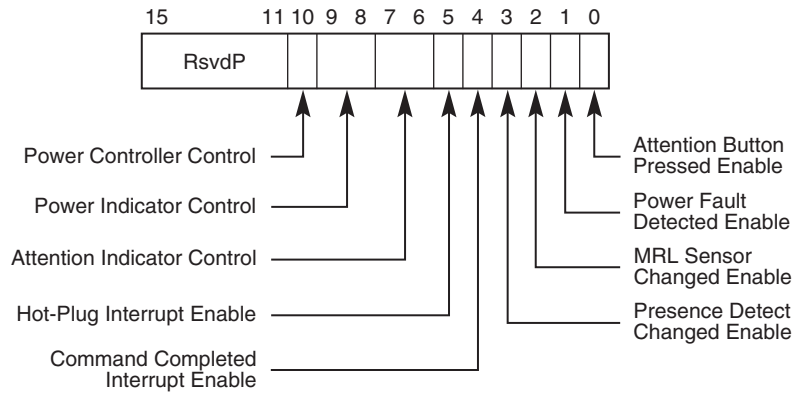
Figure 7-20: Slot Control Register

Table 7-18: Slot Control Register

Bit Location	Register Description	Attributes
0	**Attention Button Pressed Enable** – This bit when set enables the generation of Hot-Plug interrupt or wakeup event on an attention button pressed event. Default value of this field is 0.	RW
1	**Power Fault Detected Enable** – This bit when set enables the generation of Hot-Plug interrupt or wakeup event on a power fault event. Default value of this field is 0.	RW
2	**MRL Sensor Changed Enable** – This bit when set enables the generation of Hot-Plug interrupt or wakeup event on a MRL sensor changed event. Default value of this field is 0.	RW
3	**Presence Detect Changed Enable** – This bit when set enables the generation of Hot-Plug interrupt or wakeup event on a presence detect changed event. Default value of this field is 0.	RW
4	**Command Completed Interrupt Enable** – This bit when set enables the generation of Hot-Plug interrupt when a command is completed by the Hot-Plug Controller. Default value of this field is 0.	RW

Bit Location	Register Description	Attributes
5	**Hot-Plug Interrupt Enable** – This bit when set enables generation of Hot-Plug interrupt on enabled Hot-Plug events. Default value of this field is 0.	RW
7:6	**Attention Indicator Control** – Reads to this register return the current state of the Attention Indicator; writes to this register set the Attention Indicator. Defined encodings are: 00b Reserved 01b On 10b Blink 11b Off Writes to this register also cause the Port to send the appropriate ATTENTION_INDICATOR_* Messages.	RW
9:8	**Power Indicator Control** – Reads to this register return the current state of the Power Indicator; writes to this register set the Power Indicator. Defined encodings are: 00b Reserved 01b On 10b Blink 11b Off Writes to this register also cause the Port to send the appropriate POWER_INDICATOR_* Messages.	RW
10	**Power Controller Control** – When read this register returns the current state of the Power applied to the slot; when written sets the power state of the slot per the defined encodings. The defined encodings are: 0b Power On 1b Power Off	RW

7.8.11.   Slot Status Register (Offset 1Ah)

The Slot Status register provides information about PCI Express Slot specific parameters. Figure
7-21 details allocation of register fields in the Slot Status register; Table 7-19 provides the respective
bit definitions. Register fields for status bits not implemented by the device have the RsvdZ
attribute.
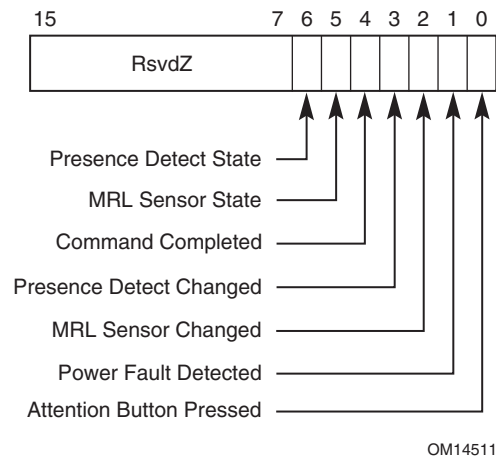
Figure 7-21: Slot Status Register

Table 7-19: Slot Status Register

Bit Location	Register Description	Attributes
0	**Attention Button Pressed** – This bit is set when the attention button is pressed.	RW1C
1	**Power Fault Detected** – This bit is set when the Power Controller detects a power fault at this slot.	RW1C
2	**MRL Sensor Changed** – This bit is set when a MRL Sensor state change is detected.	RW1C
3	**Presence Detect Changed** – This bit is set when a Presence Detect change is detected.	RW1C
4	**Command Completed** – This bit is set when the Hot-Plug Controller completes an issued command.	RW1C
5	**MRL Sensor State** – This register reports the status of the MRL sensor if it is implemented. Defined encodings are: 0b MRL Closed 1b MRL Open	RO

Bit Location	Register Description	Attributes
6	**Presence Detect State** – This bit indicates the presence of a card in the slot. The bit reflects the Presence Detect status determined via an in-band mechanism or via the Present Detect pins as defined in the *PCI Express Card Electromechanical Specification*. Defined encodings are: 0b Slot Empty 1b Card Present in slot This register is required to be implemented on all Switch devices and Root Ports. The presence detect field for Switch devices or Root Ports not connected to slots should be hardwired to 1. This register is required if a slot is implemented.	RO

7.8.12.   Root Control Register (Offset 1Ch)

The Root Control register controls PCI Express Root Complex specific parameters. Figure 7-22 details allocation of register fields in the Root Control register; Table 7-20 provides the respective bit definitions.
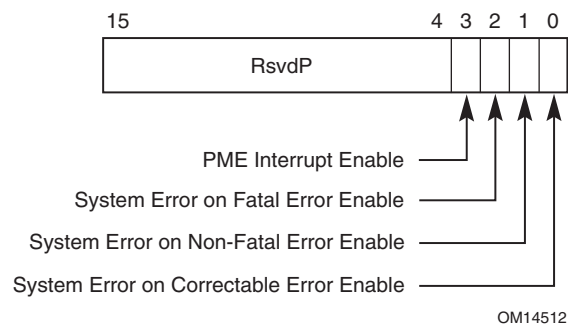
Figure 7-22: Root Control Register

Table 7-20: Root Control Register

Bit Location	Register Description	Attributes
0	**System Error on Correctable Error Enable** – If set this bit indicates that a System Error should be generated if a correctable error (ERR_COR) is reported by any of the devices in the hierarchy associated with this Root Port, or by the Root Port itself. The mechanism for signaling a System Error to the system is system specific. Default value of this field is 0.	RW
1	**System Error on Non-Fatal Error Enable** – If set this bit indicates that a System Error should be generated if a Non-fatal error (ERR_NONFATAL) is reported by any of the devices in the hierarchy associated with this Root Port, or by the Root Port itself. The mechanism for signaling a System Error to the system is system specific. Default value of this field is 0.	RW
2	**System Error on Fatal Error Enable** – If set this bit indicates that a System Error should be generated if a Fatal error (ERR_FATAL) is reported by any of the devices in the hierarchy associated with this Root Port, or by the Root Port itself. The mechanism for signaling a System Error to the system is system specific. Default value of this field is 0.	RW
3	**PME Interrupt Enable** – This bit when set enables interrupt generation upon receipt of a PME Message as reflected in the PME Status register bit (see Table 7-21). A PME interrupt is also generated if the PME Status register bit is set when this bit is set from a cleared state. Default value of this field is 0.	RW

7.8.13. Root Status Register (Offset 20h)

The Root Status register provides information about PCI Express device specific parameters. Figure 7-23 details allocation of register fields in the Root Status register; Table 7-21 provides the respective bit definitions.
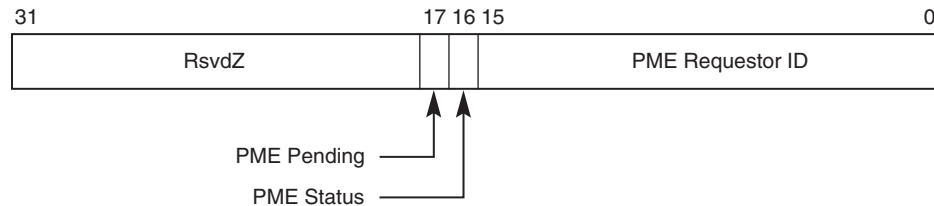
Figure 7-23: Root Status Register

Table 7-21: Root Status Register

Bit Location	Register Description	Attributes
15:0	**PME Requestor ID** – This field indicates the PCI requestor ID of the last PME requestor.	RO
16	**PME Status** – This bit indicates that PME was asserted by the requestor ID indicated in the PME Requestor ID field. Subsequent PMEs are kept pending until the status register is cleared by software by writing a 1.	RW1C
17	**PME Pending** – This read-only bit indicates that another PME is pending when the PME Status bit is set. When the PME Status bit is cleared by software; the PME is delivered by hardware by setting the PME Status bit again and updating the Requestor ID appropriately. The PME pending bit is cleared by hardware if no more PMEs are pending.	RO

7.9.   PCI Express Extended Capabilities

PCI Express Extended Capability registers are located in device configuration space at offsets 256 or greater as shown in Figure 7-24 or in the Root Complex Register Block (RCRB). These registers when located in the device configuration space are accessible using only the PCI Express extended configuration space flat memory-mapped access mechanism.

PCI Express Extended Capability structures are allocated using a linked list of optional or required PCI Express Extended Capabilities following a format resembling PCI capability structures. The first DWORD of the capability structure identifies the capability/version and points to the next capability as shown in Figure 7-24.

Each capability structure must be DWORD aligned.
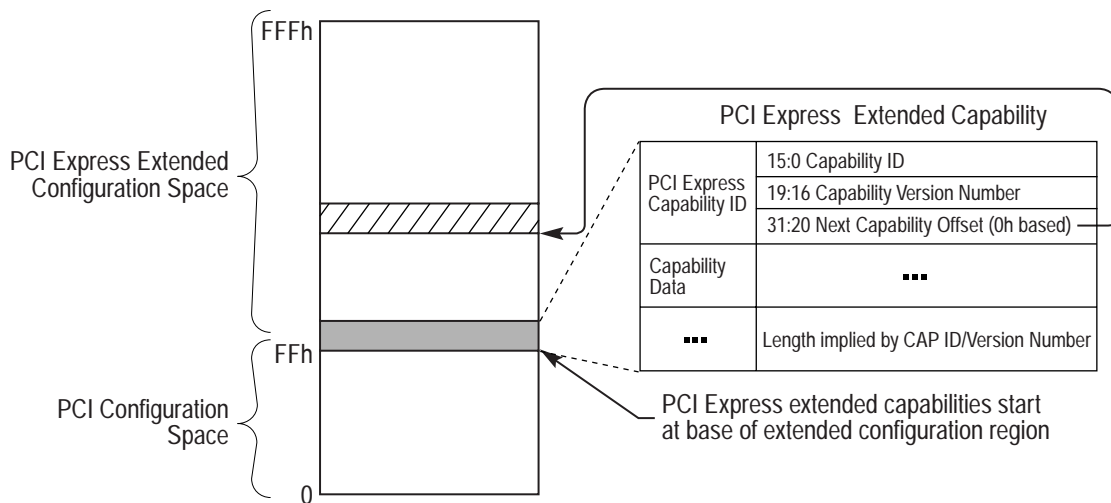
OM14302

Figure 7-24: PCI Express Extended Configuration Space Layout

7.9.1.   Extended Capabilities in Configuration Space

Extended Capabilities in device configuration space always begin at offset 100h with a PCI Express Enhanced Capability header (Section 7.9.3). Absence of any Extended Capabilities is required to be indicated by an Enhanced Capability header with a Capability ID of 0000h, a Capability Version of 0h, and a Next Capability Offset of 0h.

7.9.2.   Extended Capabilities in the Root Complex Register Block

Extended Capabilities in a Root Complex Register Block always begin at offset 0h with a PCI Express Enhanced Capability header (Section 7.9.3). Absence of any Extended Capabilities is required to be indicated by an Enhanced Capability header with a Capability ID of FFFFh and a Next Capability Offset of 0h.

7.9.3. PCI Express Enhanced Capability Header

All PCI Express extended capabilities must begin with a PCI Express Enhanced Capability header. Figure 7-25 details the allocation of register fields of a PCI Express Extended Capability header; Table 7-22 provides the respective bit definitions.
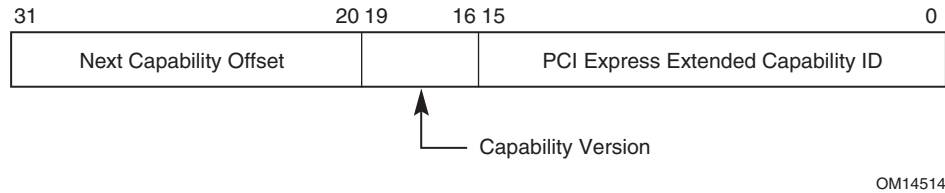
Figure 7-25: PCI Express Enhanced Capability Header

Table 7-22: PCI Express Enhanced Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the extended capability.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the capability structure present.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express capability structure or 000h if no other items exist in the linked list of capabilities.	RO
	For Extended Capabilities implemented in device configuration space, this offset is relative to the beginning of PCI compatible configuration space and thus must always be either 000h (for terminating list of capabilities) or greater than 0FFh.	
	The bottom two bits of this offset are reserved and must be implemented as 00b although software must mask them to allow for future uses of these bits.	

7.10. 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. The Advanced Error Reporting capability structure definition has additional interpretation for Root Ports; software must interpret the PCI Express device/Port Type field (Section 7.8.1) in the PCI Express Capability structure to determine the availability of additional registers for Root Ports.

Figure 7-26 shows the PCI Express Advanced Error Reporting Capability structure.

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 0.

	Byte Offset
31 0	
PCI Express Enhanced Capability Header	00h
Uncorrectable Error Status Register	04h
Uncorrectable Error Mask Register	08h
Uncorrectable Error Severity Register	0Ch
Correctable Error Status Register	10h
Correctable Error Mask Register	14h
Advanced Error Capabilities and Control Register	18h
	1Ch
Header Log Register	
Root Error Command	2Ch
Root Error Status	30h
Error Source Identification Register / Correctable Error Source Identification Register	34h

Root Ports Only

OM14319

Figure 7-26: PCI Express Advanced Error Reporting Extended Capability Structure

7.10.1. Advanced Error Reporting Enhanced Capability Header (Offset 00h)

Figure 7-27 details the allocation of register fields of a Advanced Error Reporting Enhanced Capability header; Table 7-23 provides the respective bit definitions.

5   See Section 7.9.3 for a description of the PCI Express Enhanced Capability header. The Extended Capability ID for the Advanced Error Reporting Capability is 0001h.
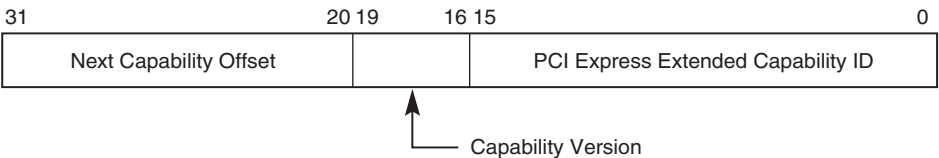
OM14515

Figure 7-27: Advanced Error Reporting Enhanced Capability Header

Table 7-23: Advanced Error Reporting Enhanced Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the extended capability. Extended Capability ID for the Advanced Error Reporting Capability is 0001h.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the capability structure present. Must be 1h for this version of the specification.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express capability structure or 000h if no other items exist in the linked list of capabilities. For Extended Capabilities implemented in device configuration space, this offset is relative to the beginning of PCI compatible configuration space and thus must always be either 000h (for terminating list of capabilities) or greater than 0FFh.	RO

7.10.2.   Uncorrectable Error Status Register (Offset 04h)

The Uncorrectable Error Status register reports error status of individual error sources on a PCI Express device. 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. Refer to Section 6.2 for further details. Register fields for bits not implemented by the device are hardwired to 0. Figure 7-28 details the allocation of register fields of the Uncorrectable Error Status register; Table 7-24 provides the respective bit definitions.
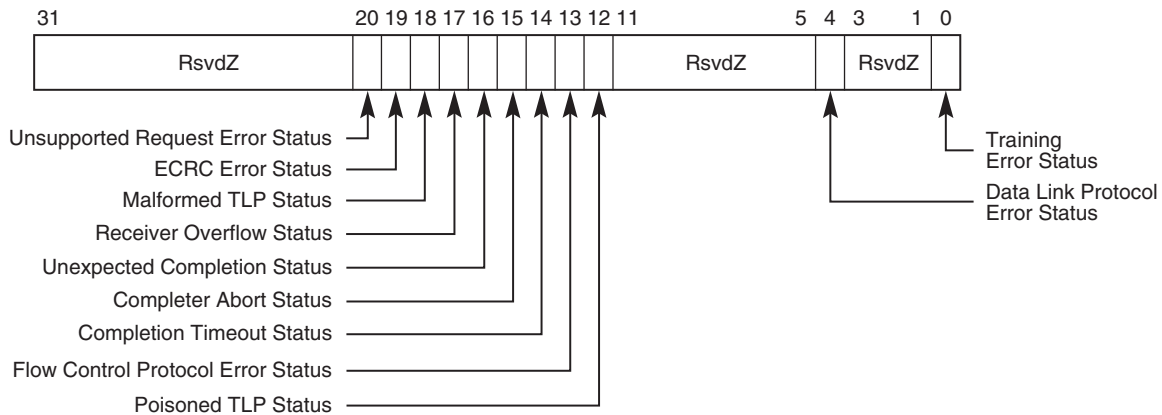
OM14516A

Figure 7-28: Uncorrectable Error Status Register

Table 7-24: Uncorrectable Error Status Register

Bit Location	Register Description	Attributes	Default Value
0	**Training Error Status** (Optional)	RW1CS	0
4	**Data Link Protocol Error Status**	RW1CS	0
12	**Poisoned TLP Status**	RW1CS	0
13	**Flow Control Protocol Error Status** (Optional)	RW1CS	0
14	**Completion Timeout Status**	RW1CS	0
15	**Completer Abort Status** (Optional)	RW1CS	0
16	**Unexpected Completion Status**	RW1CS	0
17	**Receiver Overflow Status** (Optional)	RW1CS	0
18	**Malformed TLP Status**	RW1CS	0
19	**ECRC Error Status** (Optional)	RW1CS	0
20	**Unsupported Request Error Status**	RW1CS	0

7.10.3. Uncorrectable Error Mask Register (Offset 08h)

The Uncorrectable Error Mask register controls reporting of individual errors by the device to the PCI Express Root Complex via a PCI Express error Message. A masked error (respective bit set to 1b in the mask register) is not logged in the Header Log register, does not update the First Error Pointer, and is not reported to the PCI Express Root Complex by an individual device. Refer to Section 6.2 for further details. There is a mask bit per error bit of the Uncorrectable Error Status register. Register fields for bits not implemented by the device are hardwired to 0. Figure 7-29 details the allocation of register fields of the Uncorrectable Error Mask register; Table 7-25 provides the respective bit definitions.

OM14517A

Figure 7-29: Uncorrectable Error Mask Register

Table 7-25: Uncorrectable Error Mask Register

Bit Location	Register Description	Attributes	Default Value
0	**Training Error Mask** (Optional)	RWS	0
4	**Data Link Protocol Error Mask**	RWS	0
12	**Poisoned TLP Mask**	RWS	0
13	**Flow Control Protocol Error Mask** (Optional)	RWS	0
14	**Completion Timeout Mask**	RWS	0
15	**Completer Abort Mask** (Optional)	RWS	0
16	**Unexpected Completion Mask**	RWS	0
17	**Receiver Overflow Mask** (Optional)	RWS	0
18	**Malformed TLP Mask**	RWS	0
19	**ECRC Error Mask** (Optional)	RWS	0
20	**Unsupported Request Error Mask**	RWS	0

7.10.4. Uncorrectable Error Severity Register (Offset 0Ch)

The Uncorrectable Error Severity register 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. Refer to Section 6.2 for further details. Register fields for bits not implemented by the device are hardwired to the specified default value. Figure 7-30 details the allocation of register fields of the Uncorrectable Error Severity register; Table 7-26 provides the respective bit definitions.
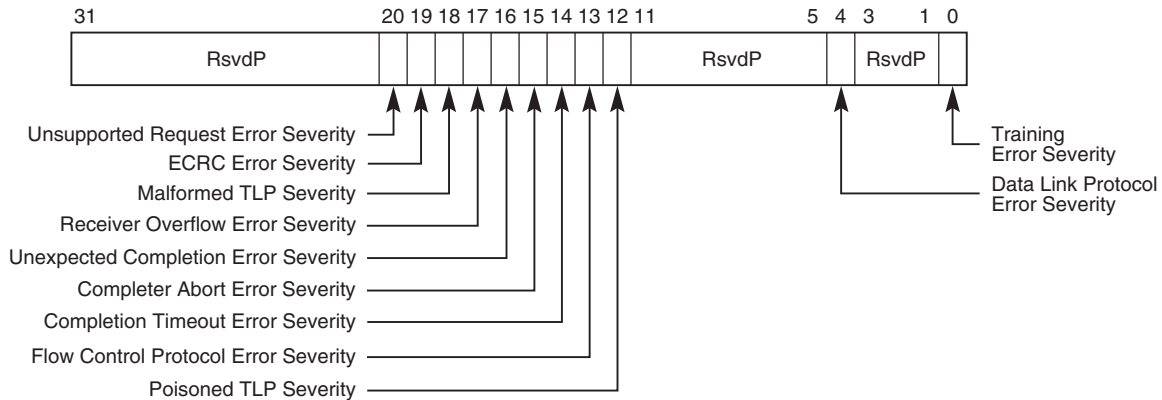
OM14518A

Figure 7-30: Uncorrectable Error Severity Register

Table 7-26: Uncorrectable Error Severity Register

Bit Location	Register Description	Attributes	Default Value
0	**Training Error Severity** (Optional)	RWS	1
4	**Data Link Protocol Error Severity**	RWS	1
12	**Poisoned TLP Severity**	RWS	0
13	**Flow Control Protocol Error Severity** (Optional)	RWS	1
14	**Completion Timeout Error Severity**	RWS	0
15	**Completer Abort Error Severity** (Optional)	RWS	0
16	**Unexpected Completion Error Severity**	RWS	0
17	**Receiver Overflow Error Severity** (Optional)	RWS	1
18	**Malformed TLP Severity**	RWS	1
19	**ECRC Error Severity** (Optional)	RWS	0
20	**Unsupported Request Error Severity**	RWS	0

7.10.5. Correctable Error Status Register (Offset 10h)

The Correctable Error Status register reports error status of individual correctable error sources on a PCI Express device. When an individual error status bit is set, it indicates that a particular error occurred; software may clear an error status by writing a 1 to the respective bit. Refer to Section 6.2 for further details. Figure 7-31 details the allocation of register fields of the Correctable Error Status register; Table 7-27 provides the respective bit definitions.
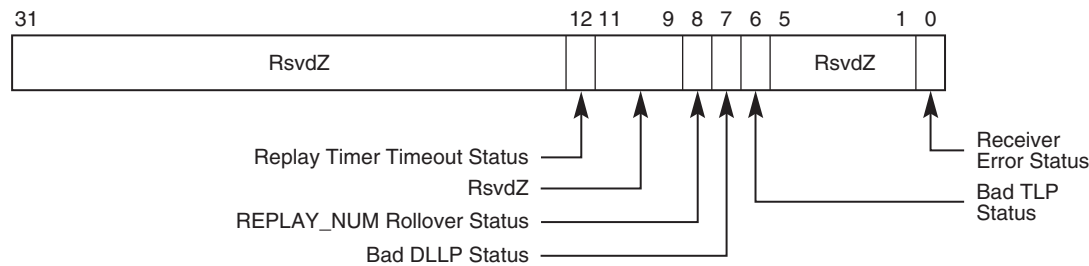
Figure 7-31: Correctable Error Status Register

Table 7-27: Correctable Error Status Register

Bit Location	Register Description	Attributes	Default Value
0	**Receiver Error Status** (Optional)	RW1CS	0
6	**Bad TLP Status**	RW1CS	0
7	**Bad DLLP Status**	RW1CS	0
8	**REPLAY_NUM Rollover Status**	RW1CS	0
12	**Replay Timer Timeout Status**	RW1CS	0

7.10.6. Correctable Error Mask Register (Offset 14h)

The Correctable Error Mask register controls reporting of individual correctable errors by device 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 an individual device. Refer to Section 6.2 for further details. There is a mask bit per error bit in the Correctable Error Status register. Figure 7-32 details the allocation of register fields of the Correctable Error Mask register; Table 7-28 provides the respective bit definitions.
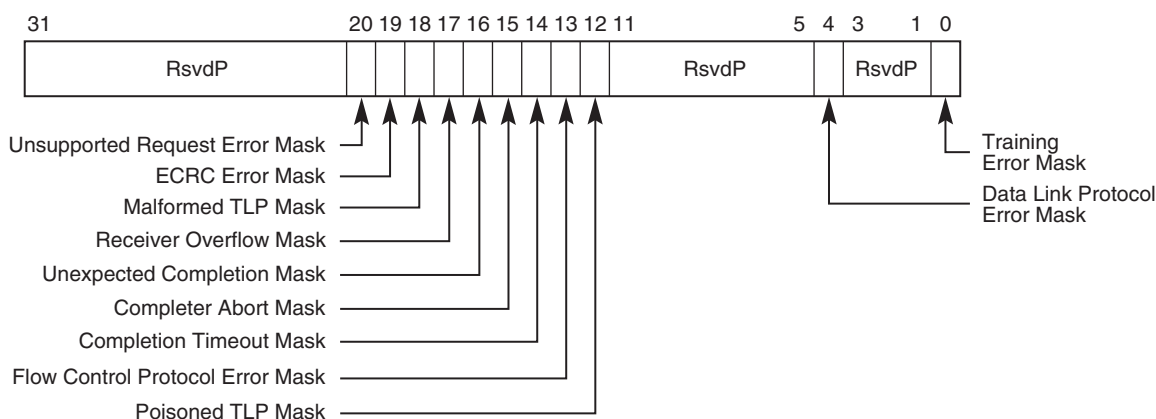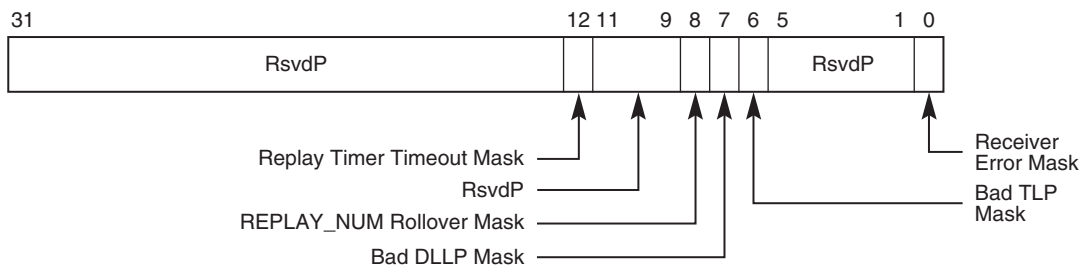
Figure 7-32: Correctable Error Mask Register

Table 7-28: Correctable Error Mask Register

Bit Location	Register Description	Attributes	Default Value
0	**Receiver Error Mask** (Optional)	RWS	0
6	**Bad TLP Mask**	RWS	0
7	**Bad DLLP Mask**	RWS	0
8	**REPLAY_NUM Rollover Mask**	RWS	0
12	**Replay Timer Timeout Mask**	RWS	0

7.10.7. Advanced Error Capabilities and Control Register (Offset 18h)

Figure 7-33 details allocation of register fields in the Advanced Error Capabilities and Control register; Table 7-29 provides the respective bit definitions. Handling of multiple errors is discussed in Section 6.2.4.2.

Figure 7-33: Advanced Error Capabilities and Control Register

Table 7-29: Advanced Error Capabilities and Control Register

Bit Location	Register Description	Attributes
4:0	**First Error Pointer** – The First Error Pointer is a read-only register that identifies the bit position of the first error reported in the Uncorrectable Error Status register. Refer to Section 6.2 for further details	ROS
5	**ECRC Generation Capable** – This bit indicates that the device is capable of generating ECRC (see Section 2.7).	RO
6	**ECRC Generation Enable** – This bit when set enables ECRC generation (see Section 2.7). Default value of this field is 0.	RWS
7	**ECRC Check Capable** – This bit indicates that the device is capable of checking ECRC (see Section 2.7).	RO
8	**ECRC Check Enable** – This bit when set enables ECRC checking (see Section 2.7). Default value of this field is 0.	RWS

7.10.8. Header Log Register (Offset 1Ch)

The Header Log register captures the Header for the TLP corresponding to a detected error; refer to Section 6.2 for further details. Section 6.2 also describes the conditions where the packet Header is logged. This register is 16 bytes and adheres to the format of the Headers defined throughout this specification.

The header is captured such that the fields of the header read by software in the same way the headers are presented in this document, when the register is read using DW accesses. Therefore, byte 0 of the header is located in byte 3 of the Header Log register, byte 1 of the header is in byte 2 of the Header Log register and so forth. For 12-byte headers, only bytes 0 through 11 of the Header Log register are used and values in bytes 12 through 15 are undefined.

Figure 7-34 details allocation of register fields in the Header Log register; Table 7-30 provides the respective bit definitions.

31	24	23	16	15	8	7	0
Header Log Register (1st DW)							
Header Byte 0		Header Byte 1		Header Byte 2		Header Byte 3	
Header Log Register (2nd DW)							
Header Byte 4		Header Byte 5		Header Byte 6		Header Byte 7	
Header Log Register (3rd DW)							
Header Byte 8		Header Byte 9		Header Byte 10		Header Byte 11	
Header Log Register (4th DW)							
Header Byte 12		Header Byte 13		Header Byte 14		Header Byte 15	
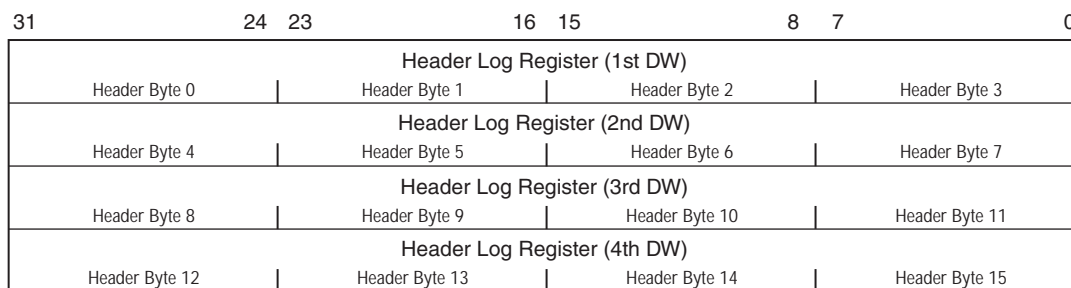
OM14549A

Figure 7-34: Header Log Register

Table 7-30: Header Log Register

Bit Location	Register Description	Attributes	Default Value
127:0	Header of TLP associated with error	ROS	0

7.10.9. Root Error Command Register (Offset 2Ch)

The Root Error Command register allows further control of Root Complex response to Correctable, Non-Fatal, and Fatal error Messages than the basic Root Complex capability to generate system errors in response to error Messages. Bit fields (see Figure 7-35) enable or disable generation of interrupts (claimed by the Root Port) in addition to system error Messages according to the definitions in Table 7-31.

Fatal Error Reporting Enable
Non-Fatal Error Reporting Enable
Correctable Error Reporting Enable

OM14523

Figure 7-35: Root Error Command Register

Table 7-31: Root Error Command Register

Bit Location	Register Description	Attributes	Default Value
0	**Correctable Error Reporting Enable** – When set this bit enables the generation of an interrupt when a correctable error is reported by any of the devices in the hierarchy associated with this Root Port. Refer to Section 6.2 for further details.	RW	0
1	**Non-Fatal Error Reporting Enable** – When set this bit enables the generation of an interrupt when a Non-fatal error is reported by any of the devices in the hierarchy associated with this Root Port. Refer to Section 6.2 for further details.	RW	0
2	**Fatal Error Reporting Enable** – When set this bit enables the generation of an interrupt when a Fatal error is reported by any of the devices in the hierarchy associated with this Root Port. Refer to Section 6.2 for further details.	RW	0

System error generation in response to PCI Express error Messages may be turned off by system software using the PCI Express Capability structure described in Section 7.8 when advanced error reporting via interrupts is enabled. Refer to Section 6.2 for further details.

7.10.10. Root Error Status Register (Offset 30h)

5   The Root Error Status register reports status of error Messages (ERR_COR), ERR_NONFATAL, and ERR_FATAL) received by the Root Complex, and of errors detected by the Root Port itself (which are treated conceptually as if the Root Port had sent an error Message to itself). ERR_NONFATAL and ERR_FATAL Messages are grouped together as uncorrectable. Each correctable and uncorrectable (Non-fatal and Fatal) error source has a first error bit and a next error
10   bit associated with it respectively. When an error is received by a Root Complex, the respective first error bit is set and the Requestor ID is logged in the Error Source Identification register. A set individual error status bit indicates that a particular error category occurred; software may clear an error status by writing a 1 to the respective bit. If software does not clear the first reported error before another error Message is received of the same category (correctable or uncorrectable),, the
15   corresponding next error status bit will be set but the Requestor ID of the subsequent error Message is discarded. The next error status bits may be cleared by software by writing a 1 to the respective bit as well. Refer to Section 6.2 for further details. This register is updated regardless of the settings of the Root Control register and the Root Error Command register. Figure 7-36 details allocation of register fields in the Root Error Status register; Table 7-32 provides the respective bit definitions.
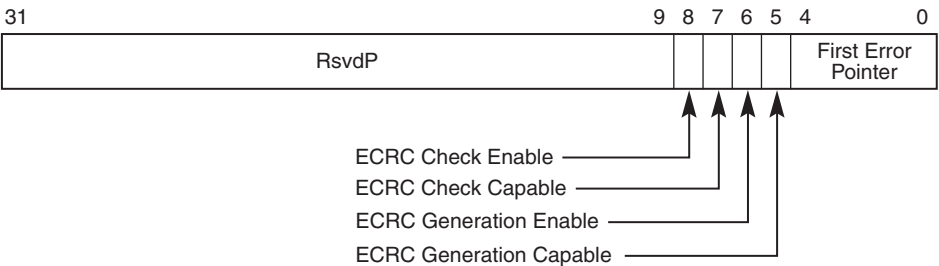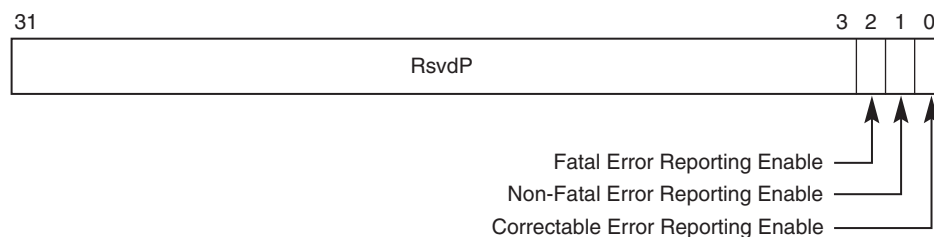
OM14524A
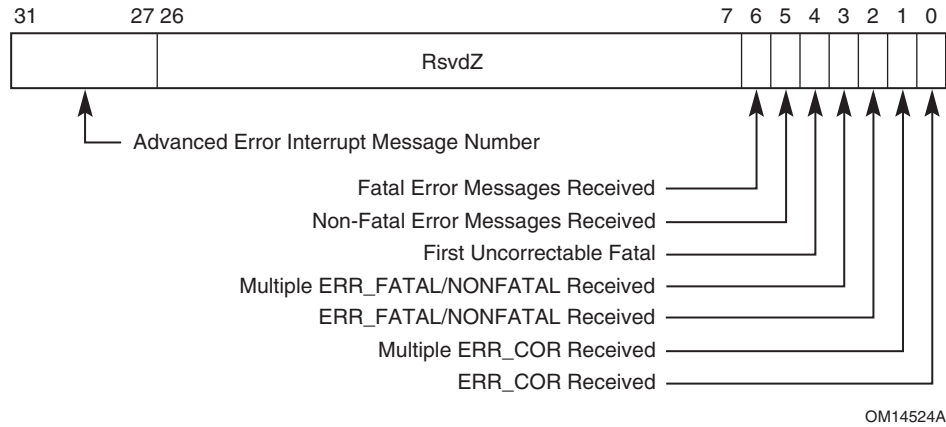
Figure 7-36: Root Error Status Register

Table 7-32: Root Error Status Register

Bit Location	Register Description	Attributes
0	**ERR_COR Received** – Set when a correctable error Message is received and this bit is not already set. Default value of this field is 0.	RW1CS
1	**Multiple ERR_COR Received** – Set when a correctable error Message is received and ERR_COR Received is already set. Default value of this field is 0.	RW1CS
2	**ERR_FATAL/NONFATAL Received** – Set when either a Fatal or a Non-fatal error Message is received and this bit is not already set. Default value of this field is 0.	RW1CS
3	**Multiple ERR_FATAL/NONFATAL Received** – Set when either a Fatal or a Non-fatal error is received and **ERR_FATAL/NONFATAL Received** is already set. Default value of this field is 0.	RW1CS
4	**First Uncorrectable Fatal** – Set to 1b when the first Uncorrectable error Message received is for a Fatal error. Default value of this field is 0.	RW1CS
5	**Non-Fatal Error Messages Received** – Set to 1b when one or more Non-Fatal Uncorrectable error Messages have been received. Default value of this field is 0.	RW1CS

Bit Location	Register Description	Attributes
6	**Fatal Error Messages Received** – Set to 1b when one or more Fatal Uncorrectable error Messages have been received. Default value of this field is 0.	RW1CS
31:27	**Advanced Error Interrupt Message Number** – If this function is allocated more than one MSI interrupt number, this register is required to contain the offset between the base Message Data and the MSI Message that is generated when any of the status bits of this capability are set. Hardware is required to update this field so that it is correct if the number of MSI Messages assigned to the device changes.	RO

7.10.11. Error Source Identification Register (Offset 34h)

The Error Source Identification register identifies the source (Requestor ID) of first correctable and uncorrectable (Non-fatal/Fatal) errors reported in the Root Error Status register. Refer to Section 6.2 for further details. This register is updated regardless of the settings of the Root Control register and the Root Error Command register. Figure 7-37 details allocation of register fields in the Error Source Identification register; Table 7-33 provides the respective bit definitions.
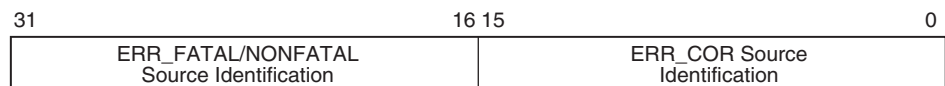
OM14525A

Figure 7-37: Error Source Identification Register

Table 7-33: Error Source Identification Register

Bit Location	Register Description	Attributes
15:0	**ERR_COR Source Identification** – Loaded with the Requestor ID indicated in the received ERR_COR Message when the ERR_COR Received register is not already set. Default value of this field is 0.	ROS
31:16	**ERR_FATAL/NONFATAL Source Identification** – Loaded with the Requestor ID indicated in the received ERR_FATAL or ERR_NONFATAL Message when the ERR_FATAL/NONFATAL Received register is not already set. Default value of this field is 0.	ROS

7.11. Virtual Channel Capability

The PCI Express Virtual Channel capability is an optional extended capability that is required to be implemented by PCI Express Ports of devices that support PCI Express functionality beyond the general purpose I/O traffic, i.e., the default Traffic Class 0 (TC0) over the default Virtual Channel 0
5   (VC0). This may apply to devices with only one VC that support TC filtering or to devices that support multiple VCs. Note that a PCI Express device that supports only TC0 over VC0 does not require VC extended capability and associated registers. Figure 7-38 provides a high level view of the PCI Express Virtual Channel Capability structure for all devices. This structure controls Virtual Channel assignment for PCI Express Links and may be present in Endpoint devices, Switch Ports
10   (Upstream and Downstream), Root Ports and RCRB. Some registers/fields in the PCI Express Virtual Channel Capability structure may have different interpretation for Endpoint devices, Switch Ports, Root Ports and RCRB. Software must interpret the PCI Express device/Port Type field (Section 7.8.1) in the PCI Express Capability structure to determine the availability and meaning of these registers/fields.

15   The PCI Express Virtual Channel Capability structure can be present in the Extended Configuration Space of all devices or in RCRB with the restriction that it is only present in the Extended Configuration Space of Function 0 for multi-function devices at their Upstream Ports.
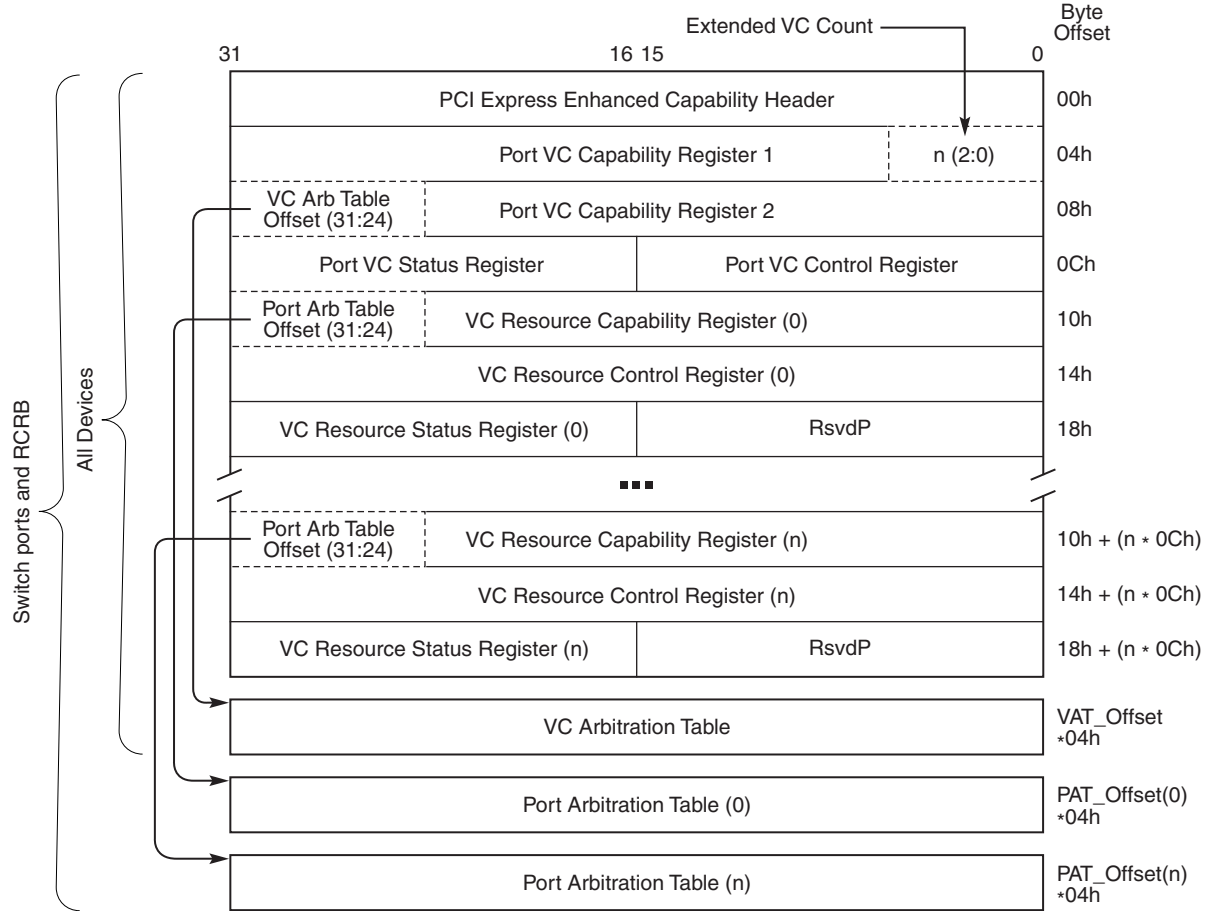
Figure 7-38: PCI Express Virtual Channel Capability Structure

The following sections describe the registers/fields of the PCI Express Virtual Channel Capability structure.

7.11.1. Virtual Channel Enhanced Capability Header

See Section 7.9.3 for a description of the PCI Express Enhanced Capability header. The Extended Capability ID for the Virtual Channel Capability is 0002h. Figure 7-39 details allocation of register fields in the Virtual Channel Enhanced Capability header; Table 7-34 provides the respective bit definitions.
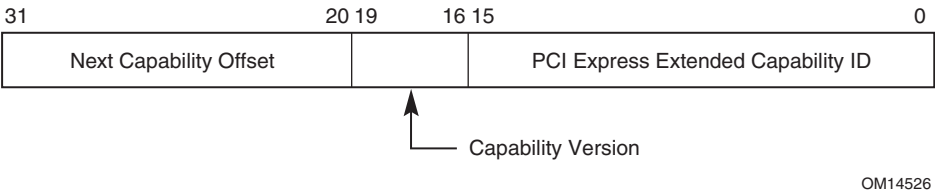
Figure 7-39: Virtual Channel Enhanced Capability Header

Table 7-34: Virtual Channel Enhanced Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the extended capability. Extended Capability ID for the Virtual Channel Capability is 0002h.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the capability structure present. Must be 1h for this version of the specification.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express capability structure or 000h if no other items exist in the linked list of capabilities. For Extended Capabilities implemented in device configuration space, this offset is relative to the beginning of PCI compatible configuration space and thus must always be either 000h (for terminating list of capabilities) or greater than 0FFh.	RO

7.11.2. Port VC Capability Register 1

The Port VC Capability register 1 describes the configuration of the Virtual Channels associated with a PCI Express Port. Figure 7-40 details allocation of register fields in the Port VC Capability register 1; Table 7-35 provides the respective bit definitions.
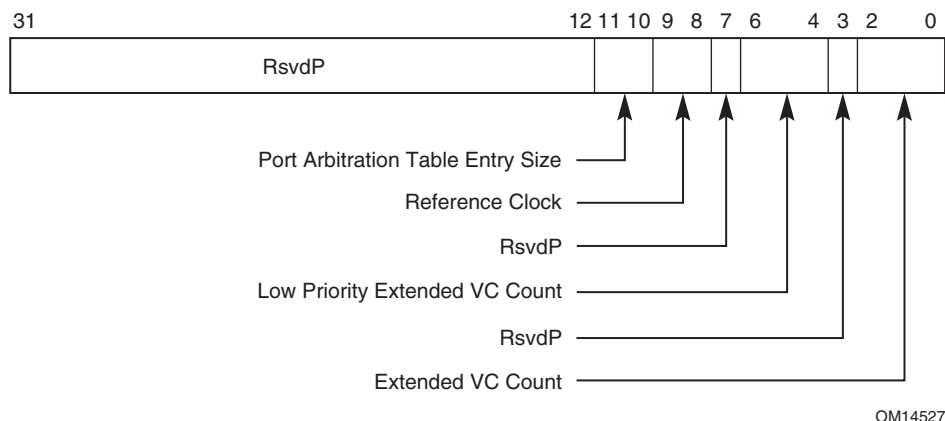
Figure 7-40: Port VC Capability Register 1

Table 7-35: Port VC Capability Register 1

Bit Location	Register Description	Attributes
2:0	**Extended VC Count** – Indicates the number of (extended) Virtual Channels in addition to the default VC supported by the device. This field is valid for all devices. The minimum value of this field is 0 (for devices that only support the default VC). The maximum value is 7.	RO
6:4	**Low Priority Extended VC Count** – Indicates the number of (extended) Virtual Channels in addition to the default VC belonging to the low-priority VC (LPVC) group that has the lowest priority with respect to other VC resources in a strict-priority VC Arbitration. This field is valid for all devices. The minimum value of this field is 0 and the maximum value is Extended VC Count.	RO

9:8	**Reference Clock** – Indicates the reference clock for Virtual Channels that support time-based WRR Port Arbitration. This field is valid for RCRB and for Switch Ports; it is not valid for Root Ports and Endpoint devices and must be set to 0.
	
	 Defined encodings are:
	
	   00b 100 ns reference clock
	
	   01b – 11b Reserved
		RO

Bit Location	Register Description	Attributes

11:10	**Port Arbitration Table Entry Size** – Indicates the size (in bits) of Port Arbitration table entry in the device. This field is valid only for RCRB and any Switch Port. It is not valid and must be set to 0 for Endpoint devices and Root Ports.
	
	Defined encodings are:
	
	   00b The size of Port Arbitration table entry is 1 bit
	
	   01b The size of Port Arbitration table entry is 2 bits
	
	   10b The size of Port Arbitration table entry is 4 bits
	
	   11b The size of Port Arbitration table entry is 8 bits
		RO

7.11.3. Port VC Capability Register 2

The Port VC Capability register 2 provides further information about the configuration of the Virtual Channels associated with a PCI Express Port. Figure 7-41 details allocation of register fields in the Port VC Capability register 2; Table 7-36 provides the respective bit definitions.


OM14532

Figure 7-41: Port VC Capability Register 2

Table 7-36: Port VC Capability Register 2

Bit Location	Register Description	Attributes

7:0	**VC Arbitration Capability** – Indicates the types of VC Arbitration supported by the device for the LPVC group. This field is valid for all devices that report a Low Priority Extended VC Count greater than 0.
	
	Each bit location within this field corresponds to a VC Arbitration capability defined below. When more than one bit in this field is set, it indicates that the Port can be configured to provide different VC arbitration services.
	
	Defined bit positions are:
	
	   Bit 0 Hardware fixed arbitration scheme, e.g., Round Robin
	
	   Bit 1 Weighted Round Robin (WRR) arbitration with 32 phases
	
	   Bit 2 WRR arbitration with 64 phases
	
	   Bit 3 WRR arbitration with 128 phases
	
	   Bits 4-7 Reserved
		RO

Bit Location	Register Description	Attributes
31:24	**VC Arbitration Table Offset** – Indicates the location of the VC Arbitration Table. This field is valid for all devices. This field contains the zero-based offset of the table in DQWORDS (16 bytes) from the base address of the Virtual Channel Capability structure. A value of 0 indicates that the table is not present.	RO

7.11.4.   Port VC Control Register

Figure 7-42 details allocation of register fields in the Port VC Control register; Table 7-37 provides the respective bit definitions.
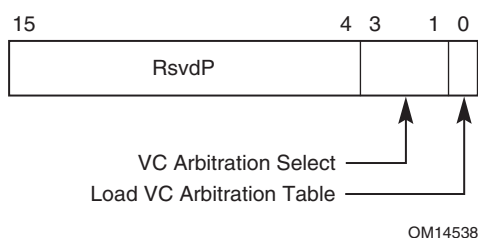

Figure 7-42: Port VC Control Register

Table 7-37: Port VC Control Register

Bit Location	Register Description	Attributes
0	**Load VC Arbitration Table** – Used for software to update the VC Arbitration Table. This field is valid for all devices when the selected VC Arbitration uses the VC Arbitration Table. Software sets this bit to request hardware to apply new values programmed into VC Arbitration Table; clearing this bit has no effect. Software checks the VC Arbitration Table Status field to confirm that new values stored in the VC Arbitration Table are latched by the VC arbitration logic. This bit always returns 0 when read.	RW
3:1	**VC Arbitration Select** – Used for software to configure the VC arbitration by selecting one of the supported VC Arbitration schemes indicated by the VC Arbitration Capability field in the Port VC Capability register 2. This field is valid for all devices. The value of this field is the number corresponding to one of the asserted bits in the VC Arbitration Capability field. This field cannot be modified when more than one VC in the LPVC group is enabled.	RW

7.11.5. Port VC Status Register

The Port VC Status register provides status of the configuration of Virtual Channels associated with a Port. Figure 7-43 details allocation of register fields in the Port VC Status register; Table 7-38 provides the respective bit definitions.
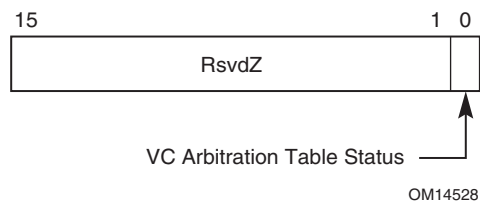
Figure 7-43: Port VC Status Register

Table 7-38: Port VC Status Register

Bit Location	Register Description	Attributes
0	**VC Arbitration Table Status** – Indicates the coherency status of the VC Arbitration Table. This field is valid for all devices when the selected VC uses the VC Arbitration Table. This bit is set by hardware when any entry of the VC Arbitration Table is written by software. This bit is cleared by hardware when hardware finishes loading values stored in the VC Arbitration Table after software sets the Load VC Arbitration Table field in the Port VC Control register. Default value of this field is 0.	RO

7.11.6. VC Resource Capability Register

The VC Resource Capability register describes the capabilities and configuration of a particular Virtual Channel resource. Figure 7-44 details allocation of register fields in the VC Resource Capability register; Table 7-39 provides the respective bit definitions.
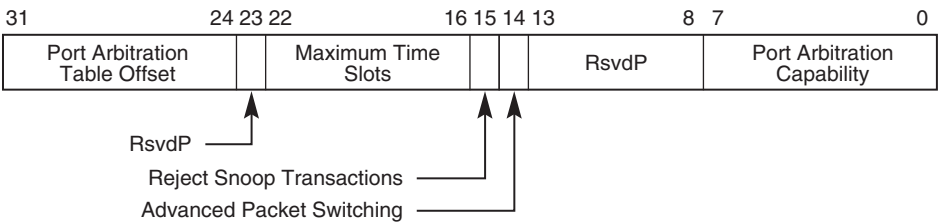
Figure 7-44: VC Resource Capability Register

Table 7-39: VC Resource Capability Register

Bit Location	Register Description	Attributes
7:0	**Port Arbitration Capability** – Indicates types of Port Arbitration supported by the VC resource. This field is valid for all Switch Ports and RCRB, but not for PCI Express Endpoint devices or Root Ports. Each bit location within this field corresponds to a Port Arbitration capability defined below. When more than one bit in this field is set, it indicates that the VC resource can be configured to provide different arbitration services. Software selects among these capabilities by writing to the Port Arbitration Select field (see below). Defined bit positions are: Bit 0 — Non-configurable hardware-fixed arbitration scheme, e.g., Round Robin (RR) Bit 1 — Weighted Round Robin (WRR) arbitration with 32 phases Bit 2 — WRR arbitration with 64 phases Bit 3 — WRR arbitration with 128 phases Bit 4 — Time-based WRR with 128 phases Bit 5 — WRR arbitration with 256 phases Bits 6-7 — Reserved	RO
14	**Advanced Packet Switching** – Indicates the VC resource only supports transactions optimized for Advanced Packet Switching (AS). This field is valid for all PCI Express Ports and RCRB. When this field is set to 0, it indicates that the VC resource is capable of supporting all transactions defined by this specification (including AS transport packets). When this field is set to 1, it indicates that the VC resource only supports transactions optimized for Advanced Packet Switching, and, therefore, must not be used for non-AS packet traffic.	RO
15	**Reject Snoop Transactions** – When set to zero, transactions with or without the No Snoop bit set within the TLP Header are allowed on this VC. When set to 1, any transaction without the No Snoop bit set within the TLP Header will be rejected as an Unsupported Request. This field is valid for Root Ports and RCRB; it is not valid for PCI Express Endpoint devices or Switch Ports.	HwInit

Bit Location	Register Description	Attributes
22:16	**Maximum Time Slots** – Indicates the maximum number of time slots (minus one) that the VC resource is capable of supporting when it is configured for time-based WRR Port Arbitration. For example, a value 0 in this field indicates the supported maximum number of time slots is 1 and a value of 127 indicates the supported maximum number of time slot is 128. This field is valid for all Switch Ports, Root Ports and RCRB, but not for PCI Express Endpoint devices. In addition, this field is valid only when Port Arbitration Capability indicates that the VC resource supports time-based WRR Port Arbitration.	HwIniit
31:24	**Port Arbitration Table Offset** – Indicates the location of the Port Arbitration Table associated with the VC resource. This field is valid for all Switch Ports and RCRB, but not for PCI Express Endpoint devices or Root Ports. This field contains the zero-based offset of the table in DQWORDS (16 bytes) from the base address of the Virtual Channel Capability structure. A value of 0 indicates that the table is not present.	RO

7.11.7. VC Resource Control Register

Figure 7-45 details allocation of register fields in the VC Resource Control register; Table 7-40 provides the respective bit definitions.


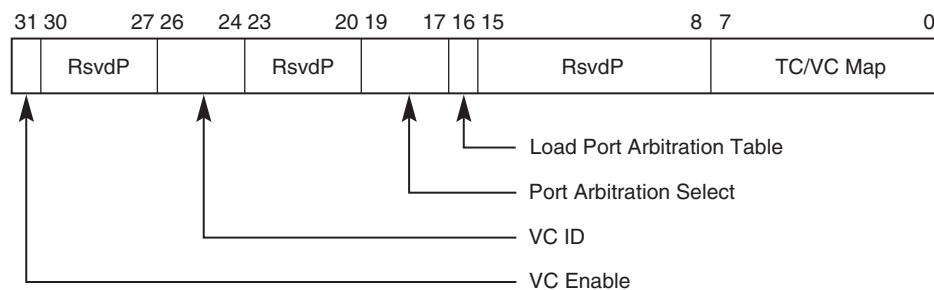**Figure 7-45: VC Resource Control Register**

Table 7-40: VC Resource Control Register

Bit Location	Register Description	Attributes
7:0	**TC/VC Map** – This field indicates the TCs that are mapped to the VC resource. This field is valid for all devices. Bit locations within this field correspond to TC values. For example, when bit 7 is set in this field, TC7 is mapped to this VC resource. When more than one bit in this field is set, it indicates that multiple TCs are mapped to the VC resource. In order to remove one or more TCs from the TC/VC Map of an enabled VC, software must ensure that no new or outstanding transactions with the TC labels are targeted at the given Link. Default value of this field is FFh for the first VC resource and is 00h for other VC resources. Note: Bit 0 of this field is read-only. It must be set to 1 for the default VC 0 and set to 0 for all other enabled VCs.	RW (see the note for exceptions)
16	**Load Port Arbitration Table** – This bit, when set, updates the Port Arbitration logic from the Port Arbitration Table for the VC resource. This field is valid for all Switch Ports and RCRB, but not for PCI Express Endpoint devices or Root Ports. In addition, this field is only valid when the Port Arbitration Table is used by the selected Port Arbitration scheme (that is indicated by a set bit in the Port Arbitration Capability field selected by Port Arbitration Select). Software sets this bit to signal hardware to update Port Arbitration logic with new values stored in Port Arbitration Table; clearing this bit has no effect. Software uses the Port Arbitration Table Status bit to confirm whether the new values of Port Arbitration Table are completely latched by the arbitration logic. This bit always returns 0 when read. Default value of this field is 0.	RW
19:17	**Port Arbitration Select** – This field configures the VC resource to provide a particular Port Arbitration service. This field is valid for RCRB and Switch Ports, but not for PCI Express Endpoint devices or Root Ports. The permissible value of this field is a number corresponding to one of the asserted bits in the Port Arbitration Capability field of the VC resource.	RW
26:24	**VC ID** – This field assigns a VC ID to the VC resource (see note for exceptions). This field is valid for all devices. This field cannot be modified when the VC is already enabled. Note: For the first VC resource (default VC), this field is a read-only field that must be set to 0 ('hard-wired').	RW

Bit Location	Register Description	Attributes
31	**VC Enable** – This field, when set, enables a Virtual Channel (see note 1 for exceptions). The Virtual Channel is disabled when this field is cleared. This field is valid for all devices. Software must use the VC Negotiation Pending bit to check whether the VC negotiation is complete. When VC Negotiation Pending bit is cleared, a 1 read from this VC Enable bit indicates that the VC is enabled (Flow Control initialization is completed for the PCI Express Port); a 0 read from this bit indicates that the Virtual Channel is currently disabled. Default value of this field is 1 for the first VC resource and is 0 for other VC resource(s). Notes: 1. This bit is hardwired to 1 for the default VC (VC0), i.e., writing to this field has no effect for VC0. 2. To enable a Virtual Channel, the VC Enable bits for that Virtual Channel must be set in both components on a Link. 3. To disable a Virtual Channel, the VC Enable bits for that Virtual Channel must be cleared in both components on a Link. 4. Software must ensure that no traffic is using a Virtual Channel at the time it is disabled. 5. Software must fully disable a Virtual Channel in both components on a Link before re-enabling the Virtual Channel.	RW

7.11.8. VC Resource Status Register

Figure 7-46 details allocation of register fields in the VC Resource Status register; Table 7-41 provides the respective bit definitions.
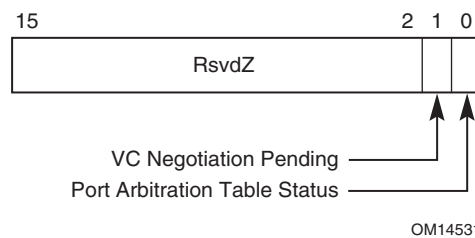

Figure 7-46: VC Resource Status Register

Table 7-41: VC Resource Status Register

Bit Location	Register Description	Attributes
0	**Port Arbitration Table Status** – This bit indicates the coherency status of the Port Arbitration Table associated with the VC resource. This field is valid for RCRB and Switch Ports, but not for PCI Express Endpoint devices or Root Ports. In addition, this field is valid only when the Port Arbitration Table is used by the selected Port Arbitration for the VC resource. This bit is set by hardware when any entry of the Port Arbitration Table is written to by software. This bit is cleared by hardware when hardware finishes loading values stored in the Port Arbitration Table after software sets the Load Port Arbitration Table field. Default value of this field is 0.	RO
1	**VC Negotiation Pending** –This bit indicates whether the Virtual Channel negotiation (initialization or disabling) is in pending state. This field is valid for all devices. When this bit is set by hardware, it indicates that the VC resource is still in the process of negotiation. This bit is cleared by hardware after the VC negotiation is complete. For a non-default Virtual Channel, software may use this bit when enabling or disabling the VC. For the default VC, this bit indicates the status of the process of Flow Control initialization. Before using a Virtual Channel, software must check whether the VC Negotiation Pending fields for that Virtual Channel are cleared in both components on a Link.	RO

7.11.9. VC Arbitration Table

The VC Arbitration Table is a read-write register array that is used to store the arbitration table for VC Arbitration. This field is valid for all devices when the selected VC Arbitration uses a WRR table. If it exists, the VC Arbitration Table is located by the VC Arbitration Table Offset field.

5   The VC Arbitration Table is a register array with fixed-size entries of 4 bits. Figure 7-47 depicts the table structure of an example VC Arbitration Table with 32-phases. Each 4-bit table entry corresponds to a phase within a WRR arbitration period. The definition of table entry is depicted in Table 7-42. The lower three bits (bit 0 to bit 2) contain the VC ID value, indicating that the corresponding phase within the WRR arbitration period is assigned to the Virtual Channel indicated
10   by the VC ID (must be a valid VC ID that corresponds to an enabled VC).

The highest bit (bit 3) of the table entry is reserved. The length of the table depends on the selected VC Arbitration as shown in Table 7-43.

When the VC Arbitration Table is used by the default VC Arbitration method, the default values of the table entries must be all zero to ensure forward progress for the default VC (with VC ID of 0).
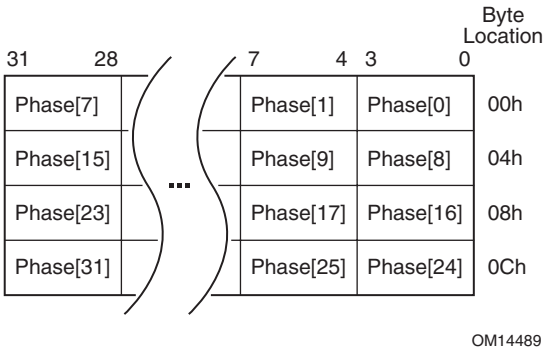
Figure 7-47: Example VC Arbitration Table with 32 Phases

Table 7-42: Definition of the 4-bit Entries in the VC Arbitration Table

Bit Location	Description	Attributes
2:0	VC ID	RW
3	RsvdP	RW

Table 7-43: Length of the VC Arbitration Table

VC Arbitration Select	VC Arbitration Table Length (in # of Entries)
001b	32
010b	64
011b	128

7.11.10. Port Arbitration Table

The Port Arbitration Table register is a read-write register array that is used to store the WRR or time-based WRR arbitration table for Port Arbitration for the VC resource. This register array is valid for all Switch Ports and RCRB, but not for Endpoint devices or Root Ports. It is only present when one or more asserted bits in the Port Arbitration Capability field indicate that the device supports a Port Arbitration scheme that uses a programmable arbitration table. Furthermore, it is only valid when one of the above mentioned bits in the Port Arbitration Capability field is selected by the Port Arbitration Select field.

The Port Arbitration Table represents one Port arbitration period. Figure 7-48 shows the structure of an example Port Arbitration Table with 128 phases and 2-bit table entries. Each table entry containing a Port Number corresponds to a phase within a Port arbitration period. For example, a table with 2-bit entries can be used by a Switch component with up to four Ports. A Port Number written to a table entry indicates that the phase within the Port Arbitration period is assigned to the selected PCI Express Port (the Port Number must be a valid one).

❑ When the WRR Port Arbitration is used for a VC of any Egress Port, at each arbitration phase, the Port Arbiter serves one transaction from the Ingress Port indicated by the Port Number of the current phase. When finished, it immediately advances to the next phase. A phase is skipped, i.e., the Port Arbiter simply moves to the next phase without delay, if the Ingress Port indicated by the phase does not contain any transaction for the VC (note that a phase cannot contain the Egress Port's Port Number).

❑ When the Time-based WRR Port Arbitration is used for a VC of any given Port, at each arbitration phase aligning to a virtual timeslot, the Port Arbiter serves one transaction from the Ingress Port indicated by the Port Number of the current phase. It advances to the next phase at the next virtual timeslot. A phase indicates an "idle" timeslot, i.e., the Port Arbiter does not serve any transaction during the phase, if

- the phase contains the Egress Port's Port Number, or

- the Ingress Port indicated by the phase does not contain any transaction for the VC.

The Port Arbitration Table Entry Size field in the Port VC Capability register determines the table entry size. The length of the table is determined by the Port Arbitration Select field as shown in Table 7-44.

When the Port Arbitration Table is used by the default Port Arbitration for the default VC, the default values for the table entries must contain at least one entry for each of the other PCI Express Ports of the device to ensure forward progress for the default VC for each Port. The table may contain RR or RR-like fair Port Arbitration for the default VC.
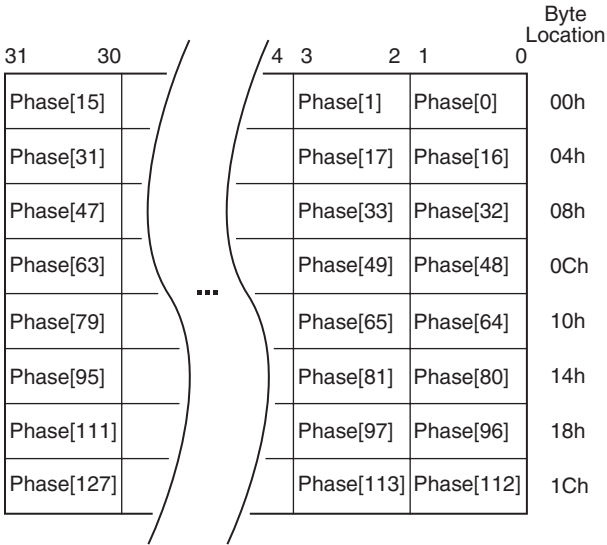
OM14490

Figure 7-48: Example Port Arbitration Table with 128 Phases and 2-bit Table Entries

Table 7-44: Length of Port Arbitration Table

Port Arbitration Select	Port Arbitration Table Length (in number of Entries)
001b	32
010b	64
011b	128
100b	128
101b	256

7.12. Device Serial Number Capability

The PCI Express Device Serial Number capability is an optional extended capability that may be implemented by any PCI Express device. The Device Serial Number is a read-only 64-bit value that is unique for a given PCI Express device. Figure 7-49 details allocation of register fields in the PCI Express Capability structure.

All multi-function devices that implement this capability must implement it for function 0; other functions that implement this capability must return the same Device Serial Number value as that reported by function 0.

A PCI Express multi-device component such as a PCI Express Switch that implements this capability must return the same Device Serial Number for each device.
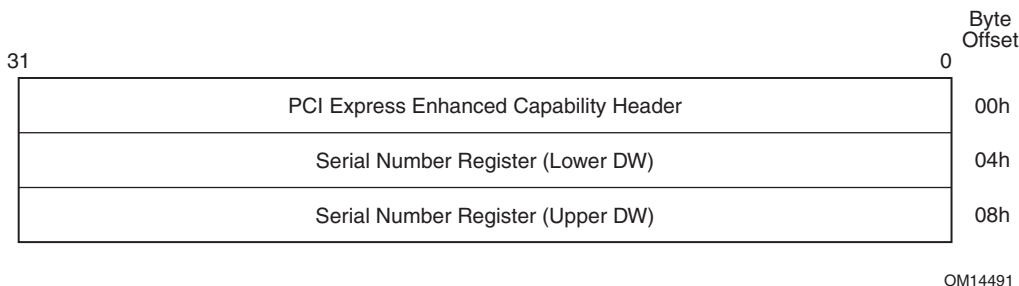
Figure 7-49: PCI Express Device Serial Number Capability Structure

7.12.1. Device Serial Number Enhanced Capability Header (Offset 00h)

Figure 7-50 details allocation of register fields in the Device Serial Number Enhanced Capability header; Table 7-45 provides the respective bit definitions. See Section 7.9.3 for a description of the PCI Express Enhanced Capability header. The Extended Capability ID for the Device Serial Number Capability is 0003h.
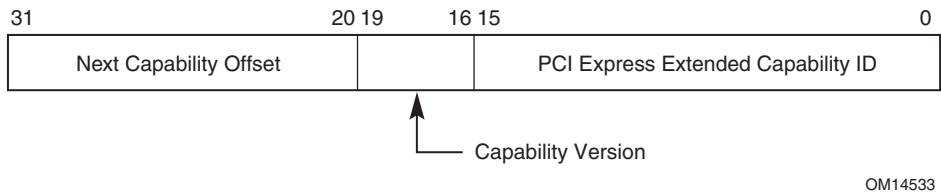
Figure 7-50: Device Serial Number Enhanced Capability Header

Table 7-45: Device Serial Number Enhanced Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the extended capability. Extended Capability ID for the Device Serial Number Capability is 0003h.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the capability structure present. Must be 1h for this version of the specification.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express capability structure or 000h if no other items exist in the linked list of capabilities. For Extended Capabilities implemented in device configuration space, this offset is relative to the beginning of PCI compatible configuration space and thus must always be either 000h (for terminating list of capabilities) or greater than 0FFh.	RO

7.12.2. Serial Number Register (Offset 04h)

The Serial Number register is a 64-bit field that contains the IEEE defined 64-bit extended unique identifier (EUI-64™). Figure 7-51 details allocation of register fields in the Serial Number register; Table 7-46 provides the respective bit definitions.

```
31                                                              0
 ┌────────────────────────────────────────────────────────────┐
 │         PCI Express Device Serial Number (1st DW)           │
 ├────────────────────────────────────────────────────────────┤
 │         PCI Express Device Serial Number (2nd DW)           │
 └────────────────────────────────────────────────────────────┘
63                                                             32
```
OM14534A

Figure 7-51: Serial Number Register

Table 7-46: Serial Number Register

Bit Location	Register Description	Attributes
63:0	**PCI Express Device Serial Number** – This field contains the IEEE defined 64-bit extended unique identifier (EUI-64™). This identifier includes a 24-bit company id value assigned by IEEE registration authority and a 40-bit extension identifier assigned by the manufacturer.	RO

7.13. Power Budgeting Capability

The PCI Express Power Budgeting Capability allows the system to properly allocate power to devices that are added to the system at runtime. Through this capability, a device can report the power it consumes on a variety of power rails, in a variety of device power management states, in a
5   variety of operating conditions. The system uses this information to ensure that the system is capable of providing the proper power and cooling levels to the device. Failure to properly indicate device power consumption may risk device or system failure.

Implementation of the power budgeting capability is optional for PCI Express devices that are implemented either in a form factor which does not require Hot-Plug support, or that are integrated
10 on the system board. PCI Express form factor specifications may require support for power budgeting. Figure 7-52 details allocation of register fields in the Power Budgeting Capability structure.
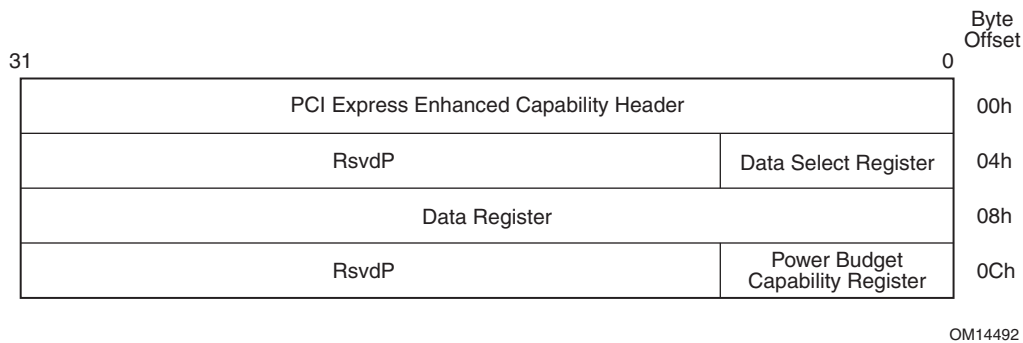
Figure 7-52: PCI Express Power Budgeting Capability Structure

7.13.1. Power Budgeting Enhanced Capability Header (Offset 00h)

15 Figure 7-53 details allocation of register fields in the Power Budgeting Enhanced Capability header; Table 7-47 provides the respective bit definitions. See Section 7.9.3 for a description of the PCI Express Enhanced Capability header. The Extended Capability ID for the Power Budgeting Capability is 0004h.
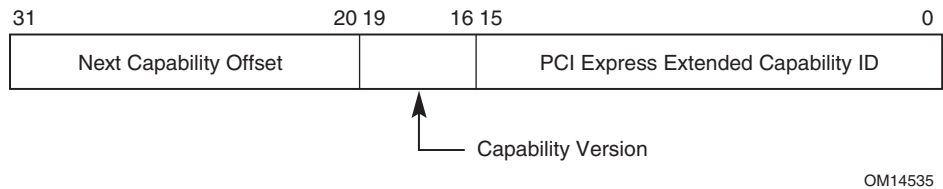
Figure 7-53: Power Budgeting Enhanced Capability Header

Table 7-47: Power Budgeting Enhanced Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the extended capability. Extended Capability ID for the Power Budgeting Capability is 0004h.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the capability structure present. Must be 1h for this version of the specification.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express capability structure or 000h if no other items exist in the linked list of capabilities. For Extended Capabilities implemented in device configuration space, this offset is relative to the beginning of PCI compatible configuration space and thus must always be either 000h (for terminating list of capabilities) or greater than 0FFh.	RO

7.13.2. Data Select Register (Offset 04h)

This read-write register indexes the Power Budgeting Data reported through the Data register and selects the DWORD of Power Budgeting Data that should appear in the Data register. Index values for this register start at 0 to select the first DWORD of Power Budgeting Data; subsequent DWORDs of Power Budgeting Data are selected by increasing index values.

7.13.3. Data Register (Offset 08h)

This read-only register returns the DWORD of Power Budgeting Data selected by the Data Select register. Each DWORD of the Power Budgeting Data describes the power usage of the device in a particular operati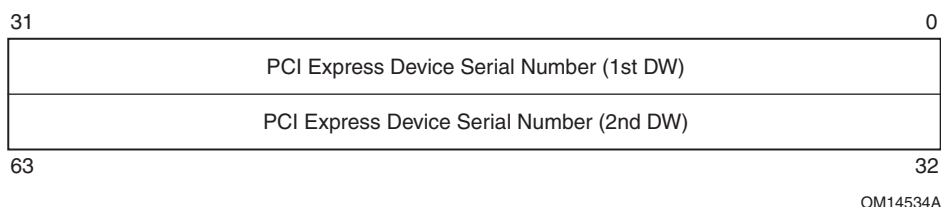ng condition. Power Budgeting Data for different operating conditions is not required to be returned in any particular order, as long as incrementing the Data Select register causes information for a different operating condition to be returned. If the Data Select register contains a value greater than or equal to the number of operating conditions for which the device provides power information, this register should return all zeros. Figure 7-54 details allocation of register fields in the Power Budgeting Data register; Table 7-48 provides the respective bit definitions.

The Base Power and Data Scale fields describe the power usage of the device; the Power Rail, Type, PM State, and PM Sub State fields describe the conditions under which the device has this power usage.
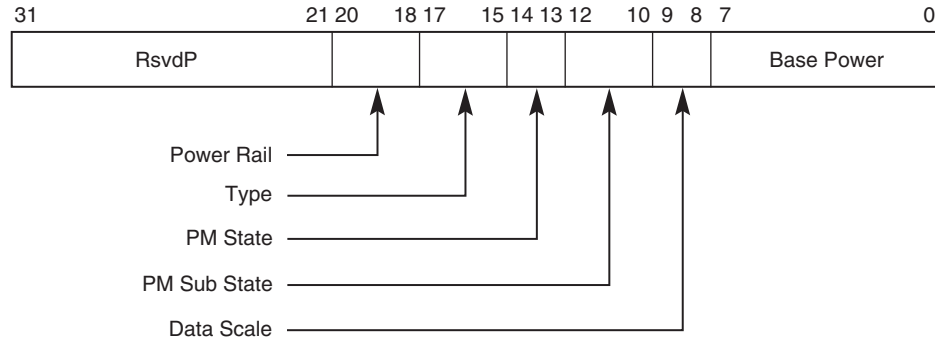
OM14536

Figure 7-54: Power Budgeting Data Register

Table 7-48: Power Budgeting Data Register

Bit Location	Register Description	Attributes
7:0	**Base Power** – Specifies in watts the base power value in the given operating condition. This value must be multiplied by the data scale to produce the actual power consumption value.	RO
9:8	**Data Scale** – Specifies the scale to apply to the Base Power value. The power consumption of the device is determined by multiplying the contents of the Base Power register field with the value corresponding to the encoding returned by this field. Defined encodings are: 00b 1.0x 01b 0.1x 10b 0.01x 11b 0.001x	RO
12:10	**PM Sub State** – Specifies the power management sub state of the operating condition being described. Defined encodings are: 000b Default Sub State 001b – 111b Device Specific Sub State	RO

Bit Location	Register Description	Attributes
14:13	**PM State** – Specifies the power management state of the operating condition being described. Defined encodings are: 00b D0 01b D1 10b D2 11b D3 A device returns 11b in this field and Aux or PME Aux in the Type register to specify the D3-Cold PM State. An encoding of 11b along with any other Type register value specifies the D3-Hot state.	RO
17:15	**Type** – Specifies the type of the operating condition being described. Defined encodings are: 000b PME Aux 001b Auxiliary 010b Idle 011b Sustained 111b Maximum All other encodings are reserved.	RO
20:18	**Power Rail** – Specifies the power rail of the operating condition being described. Defined encodings are: 000b Power (12V) 001b Power (3.3V) 010b Power (1.8V) 111b Thermal All other encodings are reserved.	RO

A device that implements the Power Budgeting Capability is required to provide data values for the D0 Max and D0 Sustained PM State/Type combinations for every power rail from which it consumes power; data for the D0 Max Thermal and D0 Sustained Thermal combinations must also be provided if these values are different from the values reported for D0 Max and D0 Sustained on the power rails.

Devices that support auxiliary power or PME from auxiliary power must provide data for the appropriate power type (Aux or PME Aux).

7.13.4.   Power Budget Capability Register (Offset 0Ch)

This register indicates the power budgeting capabilities of a device. Figure 7-55 details allocation of register fields in the Power Budget Capability register; Table 7-49 provides the respective bit definitions.
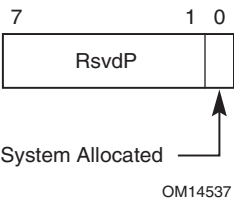
Figure 7-55: Power Budget Capability Register

Table 7-49: Power Budget Capability Register

Bit Location	Register Description	Attributes
0	**System Allocated** – This bit when set indicates that the power budget for the device is included within the system power budget. Reported Power Budgeting Data for this device should be ignored by software for power budgeting decisions if this bit is set.	HwInit

A

A. Isochronous Applications

A.1. Introduction

The design goal of isochronous mechanisms in PCI Express is to ensure that isochronous traffic receives its allocated bandwidth over a relevant time period while also preventing starvation of other non-isochronous traffic.

Furthermore, there may exist data traffic that requires a level of service falling in between what is required for bulk data traffic and isochronous data traffic. This type of traffic can be supported through the use of Port arbitration within switches, the use of TC labels [1:7], and optional additional VC resources. Policies for assignment of TC labels and VC resources that are not isochronous-focused are outside the scope of the PCI Express specification.

Two paradigms of PCI Express communication are supported by the PCI Express isochronous mechanisms: Endpoint-to-Root-Complex communication model and peer-to-peer (Endpoint-to-Endpoint) communication model. In the Endpoint-to-Root-Complex communication model, the primary isochronous traffic is memory read and write requests to the Root Complex and read completions from the Root Complex. Figure A-1 shows an example of a simple system with both communication models. In the figure, devices A, B, called Requesters, are PCI Express Endpoint devices capable of issuing isochronous request transactions, while device C and Root Complex, called Completers, are capable of being the targets of isochronous request transactions. An Endpoint-to-Root-Complex communication is established between device A and the Root Complex, and a peer-to-peer communication is established between device B and device C. In the rest of this section, Requester and Completer will be used to make reference to PCI Express elements involved in transactions. The specific aspects of each communication model will be called out explicitly.
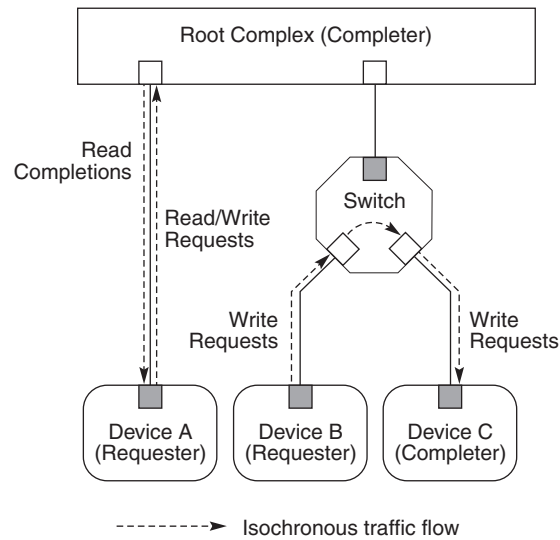
OM14288

Figure A-1: An Example Showing Endpoint-to-Root-Complex and Peer-to-Peer Communication Models

Guaranteed bandwidth and deterministic latency require end-to-end configuration of fabric resources. If isochronous traffic is intermixed with non-isochronous traffic, it may not be possible to provide any guarantees/determinism as required by the application usage model. It is recommended that system software configure and assign fabric resources such that traffic intermix either does not occur or is such that the application usage model guarantees can be met. This can be accomplished by assigning dedicated VC resources and corresponding TC labels to the isochronous traffic flow(s) on a given path within the fabric.

Note that there may be one or more isochronous traffic flows per VC/TC label and it is up to system software to insure that the aggregation of these flows does not exceed the requisite bandwidth and latency requirements.

It is also possible for a fabric to support multiple isochronous traffic flows separated across multiple VC (a given flow cannot span multiple VC/TC labels).

In general, as long as the device can meet the isochronous bandwidth and latency requirements, there is nothing to preclude a single VC device from supporting isochronous traffic if multiple TC labels are supported to delineate such traffic from non-isochronous traffic within the fabric.

A.2.   Isochronous Contract and Contract Parameters

In order to support isochronous data transfer with guaranteed bandwidth and deterministic latency, an isochronous contract must be established between a Requester/Completer pair and the PCI

5   Express fabric. This contract must enforce both resource reservation and traffic regulation. Without such a contract, two basic problems, over-subscription and congestion, may occur as illustrated in Figure A-2. When interconnect bandwidth resources are over-subscribed, the increased latency may cause failure of isochronous service and starvation of non-isochronous services. Traffic congestion occurs when flow control credits are not returned potential due to a higher than

10   expected/provisioned packet injection rate. This may cause excessive service latencies for both isochronous traffic and non-isochronous traffic.
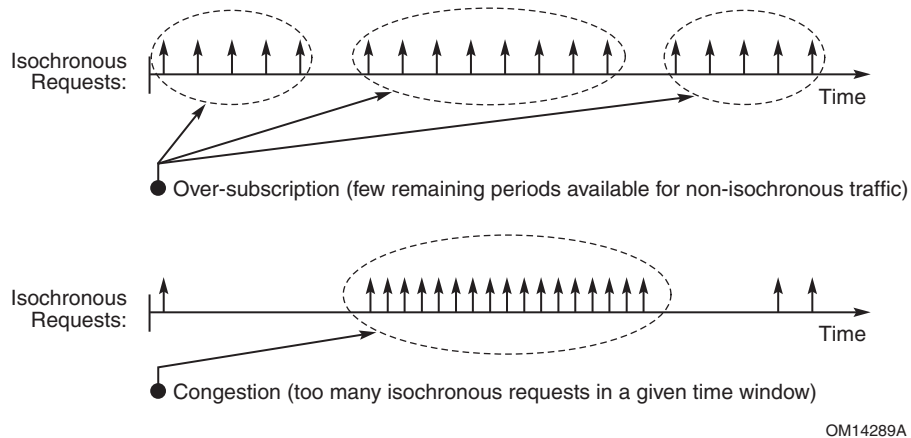
Figure A-2: Two Basic Bandwidth Resourcing Problems: Over-Subscription and Congestion

The isochronous transfer mechanism in this specification addresses these problems with traffic regulation including admission control and service discipline. Under a software managed admission control, a Requester must not issue isochronous transactions unless the required isochronous

15   bandwidth and resource have been allocated. Specifically, the isochronous bandwidth is given by the following formula:

$$BW = \frac{N \cdot Y}{T}$$

The formula defines allocated bandwidth (BW) as a function of specified number (N) of transactions of a specified payload size (Y) within a specified time period (T). Another important

20   parameter in the isochronous contract is latency. Based on the contract, isochronous transactions are completed within a specified latency (L). Once a Requester/Completer pair is admitted for isochronous communication, the bandwidth and latency are guaranteed to the Requester (a PCI Express Endpoint device) by the Completer (Root Complex for Endpoint-to-Root-Complex communication and another PCI Express Endpoint device for peer-to-peer communication) and by

25   the PCI Express fabric components (Switches).

Specific service disciplines must be implemented by isochronous-capable PCI Express components. The service disciplines are imposed to PCI Express Switches and Completers in such a manner that the service of isochronous requests is subject to a specific service interval (t). This mechanism is used to provide the method of controlling when an isochronous packet injected by a Requester is serviced. Consequently, isochronous traffic is policed in such manner that only packets that can be injected into the fabric in compliance with the isochronous contract are allowed to make immediate progress and start being serviced by the PCI Express fabric. A non-compliant Requester that tries to inject more isochronous transactions than what was being allowed by the contract is prevented from doing so by the flow-control mechanism thereby allowing compliant Requesters to correctly operate independent of non-compliant Requesters.

In the Endpoint-to-Root-Complex model, since the aggregated isochronous traffic is eventually limited by the host memory subsystem's bandwidth capabilities, isochronous read requests, and write requests (and Messages) are budgeted together. A Requester may divide the isochronous bandwidth between read requests and write requests as appropriate.

A.2.1. Isochronous Time Period and Isochronous Virtual Timeslot

The PCI Express isochronous time period (T) is uniformly divided into units of virtual timeslots (t). To provide precise isochronous bandwidth distribution only one isochronous request packet is allowed per virtual timeslot. The virtual timeslot supported by a PCI Express component is reported through the Reference Clock field in the PCI Express Virtual Channel Capability Structure defined in Section 7.11. When Reference Clock = 00b, duration of a virtual timeslot t is 100 ns. Duration of isochronous time period T depends on the number of phases of the supported time-based WRR Port arbitration table size. When the time-based WRR Port Arbitration Table size equals to 128, there are 128 virtual timeslots (t) in an isochronous time period, i.e. T = 12.8 μs.

Note that isochronous period T as well as virtual timeslots t do not need to be aligned and synchronized among different PCI Express isochronous devices, i.e., the notion of $\{T, t\}$ is local to each individual isochronous device.

A.2.2.   Isochronous Payload Size

The payload size (Y) for isochronous transactions must not exceed Max_Payload_Size (see Section 7.8.4). After configuration, the Max_Payload_Size is set and fixed for each path that supports isochronous service with a value required to meet isochronous latency. The fixed Max_Payload_Size value is used for isochronous bandwidth budgeting regardless of the actual size of data payload associated with isochronous transactions. For isochronous bandwidth budgeting, we have

$$Y = \text{Max\_Payload\_Size}$$

A transaction with partial write is treated as a normally accounted transaction. A Completer must account for partial writes as part of bandwidth assignment (for worst case servicing time).

A.2.3.   Isochronous Bandwidth Allocation

Given T, t and Y, the maximum virtual timeslots within a time period is

$$N_{max} = \frac{T}{t}$$

and the maximum specifiable isochronous bandwidth is

$$BW_{max} = \frac{Y}{t}$$

The granularity with which isochronous bandwidth can be allocated is defined as:

$$BW_{granularity} = \frac{Y}{T}$$

Given T and t at 12.8 µs and 100 ns, respectively, N_{max} is 128. As shown in Table A-1, BW_{max} and $BW_{granularity}$ are functions of the isochronous payload size Y.

Table A-1: Isochronous Bandwidth Ranges and Granularities

Y (bytes)	128	256	512	1024
BW_{max} (MB/s)	1280	2560	5120	10240
$BW_{granularity}$ (MB/s)	10	20	40	80

Similar to bandwidth budgeting, isochronous service disciplines including arbitration schemes are based on counting requests (not the sizes of those requests. Therefore, assigning isochronous bandwidth BW_{link} to a PCI Express Link is equivalent to assigning N_{link} virtual timeslots per isochronous period, where N_{link} is given by

$$N_{link} = \frac{BW_{link}}{BW_{granularity}}$$

A Switch Port serving as an Egress Port (or an RCRB serving as a "virtual" Egress Port) for an isochronous traffic, the N_{max} virtual timeslots within T are represented by the time-based WRR Port Arbitration Table in the PCI Express Virtual Channel Capability Structure detailed in Section 7.11.

The table consists of N_{max} entries. An entry in the table represents one virtual timeslot in the isochronous time period. When a table entry is given a value of PN, it means that the timeslot is assigned to an Ingress Port (in respect to the isochronous traffic targeting the Egress Port) designated by a Port Number of PN. Therefore, N_{link} virtual timeslots are assigned to the Ingress Port when there are N_{link} entries in the table with value of PN. The Egress Port may admit one isochronous request transaction from the Ingress Port for further service only when the table entry reached by the Egress Port's isochronous time ticker (that increments by 1 every t time and wraps around when reaching T) is set to PN. Even if there are outstanding isochronous requests ready in the Ingress Port, they will not be served until next round of time-based WRR arbitration. In this manner, the time-based Port Arbitration Table serves for both isochronous bandwidth assignment and isochronous traffic regulation.

For a PCI Express Endpoint device serving as a Requester or a Completer, isochronous bandwidth allocation is accomplished through negotiation between system software and device driver, which is outside of the scope of this specification.

A.2.4. Isochronous Transaction Latency

Transaction latency is composed of the latency through the PCI Express fabric and the latency contributed by the Completer. Isochronous transaction latency is defined for each transaction and measured in units of virtual timeslot t.

- The *read latency* is defined as the round-trip latency. This is the delay from the time when the device submits a memory read request packet to its Transaction Layer (Transmit side) to the time when the corresponding read completion arrives at the device's Transaction Layer (Receive side).

- The *write latency* is defined as the delay from the time when the Requester posts a memory write request to its PCI Express Transaction Layer (Transmit side) to the time when the data write becomes globally visible within the memory subsystem of the Completer. A write to memory reaches the point of global visibility when all agents accessing that memory address get the updated data.

When the upper bound and the lower bound of isochronous transaction latency are provided, the size of isochronous data buffers in a Requester can be determined. For most of common platforms, the minimum isochronous transaction latency is much smaller than the maximum. As a conservative measure, the minimum isochronous transaction latency is assumed to be zero; only guidelines on measuring the maximum isochronous transaction latency are provided here.

For a Requester, the maximum isochronous (read or write) transaction latency (L) can be accounted as the following:

$$L = L_{Fabric} + L_{Completer} ,$$

where L_{Fabric} is the maximum latency of the PCI Express fabric and $L_{Completer}$ is the maximum latency of the Completer.

L_{Fabric} which applies to both read and write transactions, depends on the topology, latency across each PCI Express Link, and the arbitration point in the path between the Requester to the Completer. The latency on a PCI Express Link depends on pipeline delays, width and operational

frequency of the Link, transmission of electrical signals across the medium, wake up latency from low power states, and delays caused by Data Link Layer Retry.

A restriction on the PCI Express topology may be imposed for each targeted platform in order to provide a practically meaningful guideline for L_{Fabric}. The values of L_{Fabric} should be reasonable and serve as practical upper limits under normal operating conditions.

The value of $L_{Completer}$ depends on the memory technology, memory configuration, and the arbitration policies in the Completer that comprehend PCI Express isochronous traffic. The target value for $L_{Completer}$ should provide enough headroom to allow for implementation tradeoffs.

Definitions of read and write transaction latencies for a Completer are different:

❑ Read transaction latency for the Completer is defined as the delay from the time a memory read transaction is available at the Receiver end of a PCI Express Port in the Completer to the time the corresponding read completion transaction is posted to the transmission end of the PCI Express Port.

❑ Write transaction latency is defined as the delay from the time a memory write transaction is available at the Receiver end of a PCI Express Port in the Completer to the time that the transmitted data is globally visible.

All of the isochronous transaction latencies defined above are based on the assumption that the Requester injects isochronous transactions uniformly. According to an isochronous contract of $\{N, T, t\}$, the uniform traffic injection is defined such that up to N transactions are evenly distributed over the isochronous period T based on a ticker granularity of virtual timeslot t. For a Requester with non-uniform isochronous transaction injection, the Requester is responsible of accounting for any additional delay due to the deviation of its injection pattern from a uniform injection pattern.

A.2.5. An Example Illustrating Isochronous Parameters

Figure A-3 illustrates the key isochronous parameters using a simplified example with $T = 20t$ and $L = 22t$. A Requester has reserved isochronous bandwidth of four transactions per T. The device shares the allocated isochronous bandwidths for both read requests and write requests. As shown, during one isochronous time period, the Requester issues two read requests and two write requests. All requests are completed within the designated transaction latency L. Also shown in the figure, there is no time dependency between the service time of write requests and the arrival time of read completions.
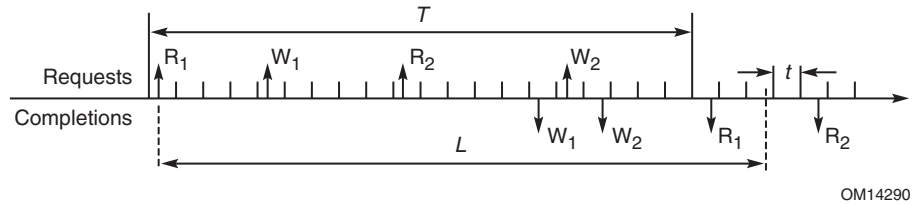
Figure A-3: A Simplified Example Illustrating PCI Express Isochronous Parameters

A.3. Isochronous Transaction Rules

Isochronous transactions follow the same rules as described in Chapter 2. In order to assist the Completer to meet latency requirements, the following additional rules further illustrate and clarify the proper behavior of isochronous transactions:

5   ❑ The value in the Length field of requests must never exceed Max_Payload_Size.

A.4. Transaction Ordering

In general, isochronous transactions follow the ordering rules described in Section 2.4. The following ordering rule further illustrates and clarifies the proper behavior of isochronous transactions:

10   ❑ There are no ordering guarantees between any isochronous and non-isochronous transactions because the traffic has been segregated into distinct VC resources.

❑ Isochronous write requests are serviced on any PCI Express Link in strictly the same order as isochronous write requests are posted.

❑ Switches must allow isochronous posted requests to pass isochronous read completions.

15 A.5. Isochronous Data Coherency

Cache coherency for isochronous transactions is an operating system software and Root Complex hardware issue. PCI Express provides the necessary mechanism to control Root Complex behavior in terms of enforcing hardware cache coherency on a per transaction basis.

For platforms where snoop latency in a Root Complex is either unbounded or can be excessively
20 large, in order to meet tight maximum isochronous transaction latency $L_{Completer}$, or more precisely $L_{Root\_Complex}$, all isochronous transactions should have the No Snoop Attribute bit set.

Root Complex must report the Root Complex's capability to the system software by setting the Reject Snoop Transactions field in the VC Resource Capability register (for any VC resource capable of supporting isochronous traffic) in RCRB. Based on whether or not a Root Complex is capable of
25 providing hardware enforced cache coherency for isochronous traffic while still meeting isochronous latency target, system software can then inform the device driver of Endpoint devices to set or unset the No Snoop Attribute bit for isochronous transactions.

Note that cache coherency considerations for isochronous traffic do not apply to peer-to-peer communication.

A.6. Flow Control

Completers and PCI Express fabric components should implement proper sizing of buffers such that under normal operating conditions, no backpressure due to flow control should be applied to isochronous traffic injected uniformly by a Requester. For Requesters that are compliant to the isochronous contract, but have bursty injection behavior, Switches and Completers may apply flow control backpressure as long as the admitted isochronous traffic is uniform and compliant to the isochronous contract. Under abnormal conditions when isochronous traffic jitter becomes significant or when isochronous traffic is oversubscribed due to excessive Data Link Layer Retry, flow control provides a natural mechanism to ensure functional correctness.

A.7. Considerations for Bandwidth Allocation

A.7.1. Isochronous Bandwidth of PCI Express Links

Isochronous bandwidth budgeting for PCI Express Links can be derived based on Link parameters such as isochronous payload size and the speed and width of the Link.

Isochronous bandwidth allocation for a PCI Express Link should be limited to certain percentage of the maximum effective Link bandwidth in order to leave sufficient bandwidth for non-isochronous traffic and to account for temporary Link bandwidth reduction due to retries. Link utilization is counted based on the actual cycles consumed on the physical PCI Express Link. The maximum number of virtual slots allowed per Link (N_{link}) depends on the isochronous packet payload size and the speed and width of the Link.

As isochronous bandwidth allocation on a PCI Express Link is based on number of request N_{link} per isochronous period. There is no distinction between read requests and write requests in budgeting isochronous bandwidth on a PCI Express Link.

A.7.2. Isochronous Bandwidth of Endpoint Devices

For peer-to-peer communication, the device driver is responsible for reporting to system software if the device is capable of being a Completer for isochronous transactions. In addition, the driver must report the device's isochronous bandwidth capability. The specifics of the report mechanism are outside the scope of this specification.

A.7.3. Isochronous Bandwidth of Switches

Allocation of isochronous bandwidth for a Switch must consider the capacity and utilization of PCI Express Links associated with the Ingress Port and the Egress Port of the Switch that connect the Requester and the Completer, respectively. The lowest common denominator of the two determines if a requested isochronous bandwidth can be supported.

A.7.4. Isochronous Bandwidth of Root Complex

Isochronous bandwidth of Root Complex is reported to the software through RCRB Structure. Specifically, the Maximum Time Slots field of the VC Resource Capability register in VC Capability

Structure indicates the total isochronous bandwidth shared by the Root Ports associated with the RCRB. Details of the platform budgeting for available isochronous bandwidth within a Root Complex are outside of the scope of this specification.

A.8.   Considerations for PCI Express Components

A.8.1.   A PCI Express Endpoint Device as a Requester

Before a PCI Express Endpoint device as a Requester can start issuing isochronous request transactions, the following configuration steps must be performed by software:

❑ Configuration of at least one VC resource capable of supporting isochronous communication and assignment of at least one TC label.

❑ Enablement of this VC resource.

When the Requester uniformly injects isochronous requests, the Receive Port, either a Switch Port or a Root Port, should issue Flow Control credit back promptly such that no backpressure should be applied to the associated VC. This type of Requester may size its buffer based on the PCI Express fabric latency L_{Fabric} plus the Completer's latency $L_{Completer}$.

When isochronous transactions are injected non-uniformly, either some transactions experience longer PCI Express fabric delay or the Requester gets back-pressured on the associated VC. This type of Requester must size its buffer to account for the deviation of its injection pattern from uniformity.

A.8.2.   A PCI Express Endpoint Device as a Completer

A PCI Express Endpoint device may serve as a Completer for isochronous peer-to-peer communication. Before a PCI Express Endpoint device starts serving isochronous transactions, system software must identify/configure a VC resource capable of supporting isochronous traffic and assigned a corresponding TC label.

An Endpoint Completer must observe the maximum isochronous transaction latency ($L_{Completer}$). An Endpoint Completer does not have to regulate isochronous request traffic if attached to a switch since Switches implement traffic regulation. However, an Endpoint Completer must size its internal buffer such that no backpressure should be applied to the corresponding VC.

A.8.3.   Switches

A Switch may have multiple ports capable of supporting isochronous transactions. Before a Switch starts serving isochronous transactions for a Port, the software must perform the following configuration steps:

5 ❑ Configuration/enablement of at least one VC resource capable of supporting isochronous communication.

❑ Configuration of the Port as an Ingress Port:

- Configuration (or reconfiguration if the associated VC of the Egress Port is already enabled) of the time-based WRR Port Arbitration Table of the targeting Egress Port to include N_{link}
10 entries set to the Ingress Port's Port Number. Here N_{link} is the isochronous allocation for the Ingress Port.

- Enabling the targeting Egress Port to load newly programmed Port Arbitration Table.

❑ Configuration of the Port as an Egress Port:

- Configuration of each VC's Port Arbitration Table with number of entries set according to
15 the assigned isochronous bandwidth for all Ingress Ports.

- Select proper VC Arbitration, e.g., as strict-priority based VC Arbitration.

- If required, configuration of the Port's VC Arbitration Table with large weights assigned accordingly to each associated VC.

Each VC associated with isochronous traffic may be served as the highest priority in arbitrating for
20 the shared PCI Express Link resource at an Egress Port. This is comprehended by a Switch's internal arbitration scheme.

In addition, a Switch Port may use "just in time" scheduling mechanism to reduce VC arbitration latency. Instead of pipelining non-isochronous Transport Layer packets to the Data Link Layer of the Egress Port in a manner that Data Link Layer transmit buffer becomes saturated, the Switch
25 Port may hold off scheduling of a new non-isochronous packet to the Data Link Layer as long as it is possible without incurring unnecessary Link idle time.

When a VC configured to support isochronous traffic is enabled for a Switch Port (ingress) that is connected to a Requester, the Switch must enforce proper traffic regulation to ensure that isochronous traffic from the Port conforms to this specification. With such enforcement, normal
30 isochronous transactions from compliant Requesters will not be impacted by ill behavior of any incompliant Requester.

The above isochronous traffic regulation mechanism only applies to request transactions but not to completion transactions. When Endpoint-to-Root-Complex and peer-to-peer communications co-exist in a Switch, an Egress Port may mix isochronous write requests and read completions in the
35 same direction. In the case of contention, the Egress Port must allow write requests to pass read completions to ensure the Switch meet latency requirement for isochronous requests.

A.8.4. Root Complex

A Root Complex may have multiple Root Ports capable of supporting isochronous transactions. Before a Root Complex starts serving isochronous transactions for a Root Port, the Port must be configured by software to enable VC to support isochronous traffic using the following configuration steps:

❑ Configuration of at least one VC resource capable of supporting isochronous communication and assignment of at least one TC label.

❑ Configuration of the Root Port as an Ingress Port:

• Configuration (or reconfiguration if the associated VC in RCRB is already enabled) of the time-based WRR Port Arbitration Table of the targeting RCRB to include N_{link} entries set to the Ingress Port's Port Number. Here N_{link} is the isochronous allocation for the Port.

• Enabling the targeting RCRB to load newly programmed Port Arbitration Table.

❑ Configuration of the Root Port as an Egress Port:

• If supported, configuration of the Root Port's VC Arbitration Table with large weights assigned to the associated VC.

A Root Complex must observe the maximum isochronous transaction latency ($L_{Completer}$ or more precisely $L_{Root\_Complex}$) that applies to all the Root Ports in the Root Complex. How a Root Complex schedules memory cycles for PCI Express isochronous transactions and other memory transactions is outside of the scope of this specification as long as $L_{Root\_Complex}$ is met for PCI Express isochronous transactions.

When a VC is enabled to support isochronous traffic for a Root Port, the Root Complex must enforce proper traffic regulation to ensure that isochronous traffic from the Root Port confirms to this specification. With such enforcement, normal isochronous transactions from compliant Requesters will not be impacted by ill behavior of any incompliant Requesters. Isochronous traffic regulation is implemented using the time-based Port Arbitration Table in RCRB.

Root Complex may perform the following operations for invalid isochronous transactions:

❑ Return partial completions for read requests with the value in the Length field exceeding Max_Payload_Size.

B
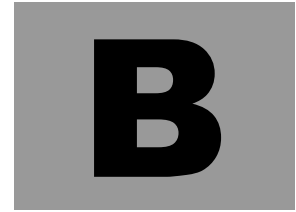
B. Symbol Encoding

Table B-1 shows the byte-to-Symbol encodings for data characters. Table B-2 shows the Symbol encodings for the Special Symbols used for TLP/DLLP Framing and for interface management. RD- and RD+ refer to the Running Disparity of the Symbol sequence on a per-Lane basis.

Table B-1: 8b/10b Data Symbol Codes

Data Byte Name	Data Byte Value (hex)	Bits HGF EDCBA (binary)	Current RD-abcdei fghj(binary)	Current RD+abcdei fghj (binary)
D0.0	00	000 00000	100111 0100	011000 1011
D1.0	01	000 00001	011101 0100	100010 1011
D2.0	02	000 00010	101101 0100	010010 1011
D3.0	03	000 00011	110001 1011	110001 0100
D4.0	04	000 00100	110101 0100	001010 1011
D5.0	05	000 00101	101001 1011	101001 0100
D6.0	06	000 00110	011001 1011	011001 0100
D7.0	07	000 00111	111000 1011	000111 0100
D8.0	08	000 01000	111001 0100	000110 1011
D9.0	09	000 01001	100101 1011	100101 0100
D10.0	0A	000 01010	010101 1011	010101 0100
D11.0	0B	000 01011	110100 1011	110100 0100
D12.0	0C	000 01100	001101 1011	001101 0100
D13.0	0D	000 01101	101100 1011	101100 0100
D14.0	0E	000 01110	011100 1011	011100 0100
D15.0	0F	000 01111	010111 0100	101000 1011
D16.0	10	000 10000	011011 0100	100100 1011
D17.0	11	000 10001	100011 1011	100011 0100
D18.0	12	000 10010	010011 1011	010011 0100
D19.0	13	000 10011	110010 1011	110010 0100
D20.0	14	000 10100	001011 1011	001011 0100
D21.0	15	000 10101	101010 1011	101010 0100
D22.0	16	000 10110	011010 1011	011010 0100

Data Byte Name	Data Byte Value (hex)	Bits HGF EDCBA (binary)	Current RD- abcdei fghj(binary)	Current RD+ abcdei fghj (binary)
D23.0	17	000 10111	111010 0100	000101 1011
D24.0	18	000 11000	110011 0100	001100 1011
D25.0	19	000 11001	100110 1011	100110 0100
D26.0	1A	000 11010	010110 1011	010110 0100
D27.0	1B	000 11011	110110 0100	001001 1011
D28.0	1C	000 11100	001110 1011	001110 0100
D29.0	1D	000 11101	101110 0100	010001 1011
D30.0	1E	000 11110	011110 0100	100001 1011
D31.0	1F	000 11111	101011 0100	010100 1011
D0.1	20	001 00000	100111 1001	011000 1001
D1.1	21	001 00001	011101 1001	100010 1001
D2.1	22	001 00010	101101 1001	010010 1001
D3.1	23	001 00011	110001 1001	110001 1001
D4.1	24	001 00100	110101 1001	001010 1001
D5.1	25	001 00101	101001 1001	101001 1001
D6.1	26	001 00110	011001 1001	011001 1001
D7.1	27	001 00111	111000 1001	000111 1001
D8.1	28	001 01000	111001 1001	000110 1001
D9.1	29	001 01001	100101 1001	100101 1001
D10.1	2A	001 01010	010101 1001	010101 1001
D11.1	2B	001 01011	110100 1001	110100 1001
D12.1	2C	001 01100	001101 1001	001101 1001
D13.1	2D	001 01101	101100 1001	101100 1001
D14.1	2E	001 01110	011100 1001	011100 1001
D15.1	2F	001 01111	010111 1001	101000 1001
D16.1	30	001 10000	011011 1001	100100 1001
D17.1	31	001 10001	100011 1001	100011 1001
D18.1	32	001 10010	010011 1001	010011 1001
D19.1	33	001 10011	110010 1001	110010 1001
D20.1	34	001 10100	001011 1001	001011 1001
D21.1	35	001 10101	101010 1001	101010 1001
D22.1	36	001 10110	011010 1001	011010 1001
D23.1	37	001 10111	111010 1001	000101 1001

Data Byte Name	Data Byte Value (hex)	Bits HGF EDCBA (binary)	Current RD- abcdei fghj(binary)	Current RD+ abcdei fghj (binary)
D24.1	38	001 11000	110011 1001	001100 1001
D25.1	39	001 11001	100110 1001	100110 1001
D26.1	3A	001 11010	010110 1001	010110 1001
D27.1	3B	001 11011	110110 1001	001001 1001
D28.1	3C	001 11100	001110 1001	001110 1001
D29.1	3D	001 11101	101110 1001	010001 1001
D30.1	3E	001 11110	011110 1001	100001 1001
D31.1	3F	001 11111	101011 1001	010100 1001
D0.2	40	010 00000	100111 0101	011000 0101
D1.2	41	010 00001	011101 0101	100010 0101
D2.2	42	010 00010	101101 0101	010010 0101
D3.2	43	010 00011	110001 0101	110001 0101
D4.2	44	010 00100	110101 0101	001010 0101
D5.2	45	010 00101	101001 0101	101001 0101
D6.2	46	010 00110	011001 0101	011001 0101
D7.2	47	010 00111	111000 0101	000111 0101
D8.2	48	010 01000	111001 0101	000110 0101
D9.2	49	010 01001	100101 0101	100101 0101
D10.2	4A	010 01010	010101 0101	010101 0101
D11.2	4B	010 01011	110100 0101	110100 0101
D12.2	4C	010 01100	001101 0101	001101 0101
D13.2	4D	010 01101	101100 0101	101100 0101
D14.2	4E	010 01110	011100 0101	011100 0101
D15.2	4F	010 01111	010111 0101	101000 0101
D16.2	50	010 10000	011011 0101	100100 0101
D17.2	51	010 10001	100011 0101	100011 0101
D18.2	52	010 10010	010011 0101	010011 0101
D19.2	53	010 10011	110010 0101	110010 0101
D20.2	54	010 10100	001011 0101	001011 0101
D21.2	55	010 10101	101010 0101	101010 0101
D22.2	56	010 10110	011010 0101	011010 0101
D23.2	57	010 10111	111010 0101	000101 0101
D24.2	58	010 11000	110011 0101	001100 0101

Data Byte Name	Data Byte Value (hex)	Bits HGF EDCBA (binary)	Current RD- abcdei fghj(binary)	Current RD+ abcdei fghj (binary)
D25.2	59	010 11001	100110 0101	100110 0101
D26.2	5A	010 11010	010110 0101	010110 0101
D27.2	5B	010 11011	110110 0101	001001 0101
D28.2	5C	010 11100	001110 0101	001110 0101
D29.2	5D	010 11101	101110 0101	010001 0101
D30.2	5E	010 11110	011110 0101	100001 0101
D31.2	5F	010 11111	101011 0101	010100 0101
D0.3	60	011 00000	100111 0011	011000 1100
D1.3	61	011 00001	011101 0011	100010 1100
D2.3	62	011 00010	101101 0011	010010 1100
D3.3	63	011 00011	110001 1100	110001 0011
D4.3	64	011 00100	110101 0011	001010 1100
D5.3	65	011 00101	101001 1100	101001 0011
D6.3	66	011 00110	011001 1100	011001 0011
D7.3	67	011 00111	111000 1100	000111 0011
D8.3	68	011 01000	111001 0011	000110 1100
D9.3	69	011 01001	100101 1100	100101 0011
D10.3	6A	011 01010	010101 1100	010101 0011
D11.3	6B	011 01011	110100 1100	110100 0011
D12.3	6C	011 01100	001101 1100	001101 0011
D13.3	6D	011 01101	101100 1100	101100 0011
D14.3	6E	011 01110	011100 1100	011100 0011
D15.3	6F	011 01111	010111 0011	101000 1100
D16.3	70	011 10000	011011 0011	100100 1100
D17.3	71	011 10001	100011 1100	100011 0011
D18.3	72	011 10010	010011 1100	010011 0011
D19.3	73	011 10011	110010 1100	110010 0011
D20.3	74	011 10100	001011 1100	001011 0011
D21.3	75	011 10101	101010 1100	101010 0011
D22.3	76	011 10110	011010 1100	011010 0011
D23.3	77	011 10111	111010 0011	000101 1100
D24.3	78	011 11000	110011 0011	001100 1100
D25.3	79	011 11001	100110 1100	100110 0011

Data Byte Name	Data Byte Value (hex)	Bits HGF EDCBA (binary)	Current RD- abcdei fghj(binary)	Current RD+ abcdei fghj (binary)
D26.3	7A	011 11010	010110 1100	010110 0011
D27.3	7B	011 11011	110110 0011	001001 1100
D28.3	7C	011 11100	001110 1100	001110 0011
D29.3	7D	011 11101	101110 0011	010001 1100
D30.3	7E	011 11110	011110 0011	100001 1100
D31.3	7F	011 11111	101011 0011	010100 1100
D0.4	80	100 00000	100111 0010	011000 1101
D1.4	81	100 00001	011101 0010	100010 1101
D2.4	82	100 00010	101101 0010	010010 1101
D3.4	83	100 00011	110001 1101	110001 0010
D4.4	84	100 00100	110101 0010	001010 1101
D5.4	85	100 00101	101001 1101	101001 0010
D6.4	86	100 00110	011001 1101	011001 0010
D7.4	87	100 00111	111000 1101	000111 0010
D8.4	88	100 01000	111001 0010	000110 1101
D9.4	89	100 01001	100101 1101	100101 0010
D10.4	8A	100 01010	010101 1101	010101 0010
D11.4	8B	100 01011	110100 1101	110100 0010
D12.4	8C	100 01100	001101 1101	001101 0010
D13.4	8D	100 01101	101100 1101	101100 0010
D14.4	8E	100 01110	011100 1101	011100 0010
D15.4	8F	100 01111	010111 0010	101000 1101
D16.4	90	100 10000	011011 0010	100100 1101
D17.4	91	100 10001	100011 1101	100011 0010
D18.4	92	100 10010	010011 1101	010011 0010
D19.4	93	100 10011	110010 1101	110010 0010
D20.4	94	100 10100	001011 1101	001011 0010
D21.4	95	100 10101	101010 1101	101010 0010
D22.4	96	100 10110	011010 1101	011010 0010
D23.4	97	100 10111	111010 0010	000101 1101
D24.4	98	100 11000	110011 0010	001100 1101
D25.4	99	100 11001	100110 1101	100110 0010
D26.4	9A	100 11010	010110 1101	010110 0010

Data Byte Name	Data Byte Value (hex)	Bits HGF EDCBA (binary)	Current RD- abcdei fghj(binary)	Current RD+ abcdei fghj (binary)
D27.4	9B	100 11011	110110 0010	001001 1101
D28.4	9C	100 11100	001110 1101	001110 0010
D29.4	9D	100 11101	101110 0010	010001 1101
D30.4	9E	100 11110	011110 0010	100001 1101
D31.4	9F	100 11111	101011 0010	010100 1101
D0.5	A0	101 00000	100111 1010	011000 1010
D1.5	A1	101 00001	011101 1010	100010 1010
D2.5	A2	101 00010	101101 1010	010010 1010
D3.5	A3	101 00011	110001 1010	110001 1010
D4.5	A4	101 00100	110101 1010	001010 1010
D5.5	A5	101 00101	101001 1010	101001 1010
D6.5	A6	101 00110	011001 1010	011001 1010
D7.5	A7	101 00111	111000 1010	000111 1010
D8.5	A8	101 01000	111001 1010	000110 1010
D9.5	A9	101 01001	100101 1010	100101 1010
D10.5	AA	101 01010	010101 1010	010101 1010
D11.5	AB	101 01011	110100 1010	110100 1010
D12.5	AC	101 01100	001101 1010	001101 1010
D13.5	AD	101 01101	101100 1010	101100 1010
D14.5	AE	101 01110	011100 1010	011100 1010
D15.5	AF	101 01111	010111 1010	101000 1010
D16.5	B0	101 10000	011011 1010	100100 1010
D17.5	B1	101 10001	100011 1010	100011 1010
D18.5	B2	101 10010	010011 1010	010011 1010
D19.5	B3	101 10011	110010 1010	110010 1010
D20.5	B4	101 10100	001011 1010	001011 1010
D21.5	B5	101 10101	101010 1010	101010 1010
D22.5	B6	101 10110	011010 1010	011010 1010
D23.5	B7	101 10111	111010 1010	000101 1010
D24.5	B8	101 11000	110011 1010	001100 1010
D25.5	B9	101 11001	100110 1010	100110 1010
D26.5	BA	101 11010	010110 1010	010110 1010
D27.5	BB	101 11011	110110 1010	001001 1010

Data Byte Name	Data Byte Value (hex)	Bits HGF EDCBA (binary)	Current RD- abcdei fghj(binary)	Current RD+ abcdei fghj (binary)
D28.5	BC	101 11100	001110 1010	001110 1010
D29.5	BD	101 11101	101110 1010	010001 1010
D30.5	BE	101 11110	011110 1010	100001 1010
D31.5	BF	101 11111	101011 1010	010100 1010
D0.6	C0	110 00000	100111 0110	011000 0110
D1.6	C1	110 00001	011101 0110	100010 0110
D2.6	C2	110 00010	101101 0110	010010 0110
D3.6	C3	110 00011	110001 0110	110001 0110
D4.6	C4	110 00100	110101 0110	001010 0110
D5.6	C5	110 00101	101001 0110	101001 0110
D6.6	C6	110 00110	011001 0110	011001 0110
D7.6	C7	110 00111	111000 0110	000111 0110
D8.6	C8	110 01000	111001 0110	000110 0110
D9.6	C9	110 01001	100101 0110	100101 0110
D10.6	CA	110 01010	010101 0110	010101 0110
D11.6	CB	110 01011	110100 0110	110100 0110
D12.6	CC	110 01100	001101 0110	001101 0110
D13.6	CD	110 01101	101100 0110	101100 0110
D14.6	CE	110 01110	011100 0110	011100 0110
D15.6	CF	110 01111	010111 0110	101000 0110
D16.6	D0	110 10000	011011 0110	100100 0110
D17.6	D1	110 10001	100011 0110	100011 0110
D18.6	D2	110 10010	010011 0110	010011 0110
D19.6	D3	110 10011	110010 0110	110010 0110
D20.6	D4	110 10100	001011 0110	001011 0110
D21.6	D5	110 10101	101010 0110	101010 0110
D22.6	D6	110 10110	011010 0110	011010 0110
D23.6	D7	110 10111	111010 0110	000101 0110
D24.6	D8	110 11000	110011 0110	001100 0110
D25.6	D9	110 11001	100110 0110	100110 0110
D26.6	DA	110 11010	010110 0110	010110 0110
D27.6	DB	110 11011	110110 0110	001001 0110
D28.6	DC	110 11100	001110 0110	001110 0110

Data Byte Name	Data Byte Value (hex)	Bits HGF EDCBA (binary)	Current RD- abcdei fghj(binary)	Current RD+ abcdei fghj (binary)
D29.6	DD	110 11101	101110 0110	010001 0110
D30.6	DE	110 11110	011110 0110	100001 0110
D31.6	DF	110 11111	101011 0110	010100 0110
D0.7	E0	111 00000	100111 0001	011000 1110
D1.7	E1	111 00001	011101 0001	100010 1110
D2.7	E2	111 00010	101101 0001	010010 1110
D3.7	E3	111 00011	110001 1110	110001 0001
D4.7	E4	111 00100	110101 0001	001010 1110
D5.7	E5	111 00101	101001 1110	101001 0001
D6.7	E6	111 00110	011001 1110	011001 0001
D7.7	E7	111 00111	111000 1110	000111 0001
D8.7	E8	111 01000	111001 0001	000110 1110
D9.7	E9	111 01001	100101 1110	100101 0001
D10.7	EA	111 01010	010101 1110	010101 0001
D11.7	EB	111 01011	110100 1110	110100 1000
D12.7	EC	111 01100	001101 1110	001101 0001
D13.7	ED	111 01101	101100 1110	101100 1000
D14.7	EE	111 01110	011100 1110	011100 1000
D15.7	EF	111 01111	010111 0001	101000 1110
D16.7	F0	111 10000	011011 0001	100100 1110
D17.7	F1	111 10001	100011 0111	100011 0001
D18.7	F2	111 10010	010011 0111	010011 0001
D19.7	F3	111 10011	110010 1110	110010 0001
D20.7	F4	111 10100	001011 0111	001011 0001
D21.7	F5	111 10101	101010 1110	101010 0001
D22.7	F6	111 10110	011010 1110	011010 0001
D23.7	F7	111 10111	111010 0001	000101 1110
D24.7	F8	111 11000	110011 0001	001100 1110
D25.7	F9	111 11001	100110 1110	100110 0001
D26.7	FA	111 11010	010110 1110	010110 0001
D27.7	FB	111 11011	110110 0001	001001 1110
D28.7	FC	111 11100	001110 1110	001110 0001
D29.7	FD	111 11101	101110 0001	010001 1110

| Data Byte Name	Data Byte Value (hex)	Bits HGF EDCBA (binary)	Current RD- abcdei fghj(binary)	Current RD+ abcdei fghj (binary)
D30.7	FE	111 11110	011110 0001	100001 1110
D31.7	FF	111 11111	101011 0001	010100 1110

Table B-2: 8b/10b Special Character Symbol Codes

Data Byte Name	Data Byte Value	Bits HGF EDCBA	Current RD - abcdei fghj	Current RD + abcdei fghj
K28.0	1C	000 11100	001111 0100	110000 1011
K28.1	3C	001 11100	001111 1001	110000 0110
K28.2	5C	010 11100	001111 0101	110000 1010
K28.3	7C	011 11100	001111 0011	110000 1100
K28.4	9C	100 11100	001111 0010	110000 1101
K28.5	BC	101 11100	001111 1010	110000 0101
K28.6	DC	110 11100	001111 0110	110000 1001
K28.7	FC	111 11100	001111 1000	110000 0111
K23.7	F7	111 10111	111010 1000	000101 0111
K27.7	FB	111 11011	110110 1000	001001 0111
K29.7	FD	111 11101	101110 1000	010001 0111
K30.7	FE	111 11110	011110 1000	100001 0111

C
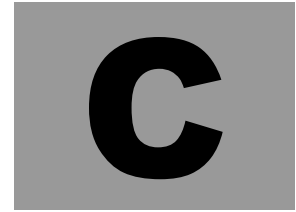
C. Physical Layer Appendix

C.1. Data Scrambling

The following subroutines encode and decode an 8-bit value contained in "inbyte" with the LFSR. This is presented as one example only; there are many ways to obtain the proper output. This example demonstrates how to advance the LFSR eight times in one operation and how to XOR the data in one operation. Many other implementations are possible but they must all produce the same output as that shown here.

The following algorithm uses the "C" programming language conventions, where "<<" and ">>" represent the shift left and shift right operators, ">" is the compare greater than operator, and "^" is the exclusive or operator, and "&" is the logical "AND" operator.

```
/*
   this routine implements the serial descrambling algorithm in parallel form
   for the LSFR polynomial:  x^16+x^5+x^4+x^3+1
   this advances the LSFR 8 bits every time it is called
   this requires fewer than 25 xor gates to implement (with a static register)

   The XOR required to advance 8 bits/clock is:
bit   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15

      8   9  10  11  12  13  14  15   0   1   2   3   4   5   6   7
                  8   9  10  11  12  13  14  15
                      8   9  10  11  12  13  14  15
                          8   9  10  11  12  13  14  15

   The serial data is just the reverse of the upper byte:
bit   0   1   2   3   4   5   6   7
     15  14  13  12  11  10   9   8
*/

int scramble_byte(int inbyte)
{

static int scrambit[16];
static int bit[16];
static int bit_out[16];
static unsigned short lfsr = 0xffff; // 16 bit short for polynomial
int i, outbyte;

    if (inbyte == COMMA)   // if this is a comma
    {
        lfsr = 0xffff;     // reset the LFSR
        return (COMMA);    // and return the same data
    }
```

```
    if (inbyte == SKIP)     // don't advance or encode on skip
        return (SKIP);

    for (i=0; i<16;i++)     // convert LFSR to bit array for legibility
        bit[i] = (lfsr >> i) & 1;

    for (i=0; i<8; i++)     // convert byte to be scrambled for legibility
        scrambit[i] = (inbyte >> i) & 1;


    // apply the xor to the data
    if (! (inbyte & 0x100) &&    // if not a KCODE, scramble the data
         ! (TrainingSequence == TRUE))  // and if not in the middle of

    {                                   // a training sequence
        scrambit[0] ^= bit[15];
        scrambit[1] ^= bit[14];
        scrambit[2] ^= bit[13];
        scrambit[3] ^= bit[12];
        scrambit[4] ^= bit[11];
        scrambit[5] ^= bit[10];
        scrambit[6] ^= bit[9];
        scrambit[7] ^= bit[8];
    }

    // Now advance the LFSR 8 serial clocks
    bit_out[ 0] = bit[ 8];
    bit_out[ 1] = bit[ 9];
    bit_out[ 2] = bit[10];
    bit_out[ 3] = bit[11] ^ bit[ 8];
    bit_out[ 4] = bit[12] ^ bit[ 9] ^ bit[ 8];
    bit_out[ 5] = bit[13] ^ bit[10] ^ bit[ 9] ^ bit[ 8];
    bit_out[ 6] = bit[14] ^ bit[11] ^ bit[10] ^ bit[ 9];
    bit_out[ 7] = bit[15] ^ bit[12] ^ bit[11] ^ bit[10];
    bit_out[ 8] = bit[ 0] ^ bit[13] ^ bit[12] ^ bit[11];
    bit_out[ 9] = bit[ 1] ^ bit[14] ^ bit[13] ^ bit[12];
    bit_out[10] = bit[ 2] ^ bit[15] ^ bit[14] ^ bit[13];
    bit_out[11] = bit[ 3]          ^ bit[15] ^ bit[14];
    bit_out[12] = bit[ 4]                    ^ bit[15];
    bit_out[13] = bit[ 5];
    bit_out[14] = bit[ 6];
    bit_out[15] = bit[ 7];
    lfsr = 0;
    for (i=0; i <16; i++) // convert the LFSR back to an integer
        lfsr += (bit_out[i] << i);

    outbyte = 0;
    for (i= 0; i<8; i++) // convert data back to an integer
        outbyte += (scrambit[i] << i);

    return outbyte;
}

/*  NOTE THAT THE DESCRAMBLE ROUTINE IS IDENTICAL TO THE SCRAMBLE ROUTINE
    this routine implements the serial descrambling algorithm in parallel form
    this advances the lfsr 8 bits every time it is called
    this uses fewer than 25 xor gates to implement (with a static register)
    The XOR tree is the same as the scrambling routine
*/

int unscramble_byte(int inbyte)
{
```

```
    static int descrambit[8];
    static int bit[16];
    static int bit_out[16];
    static unsigned short lfsr = 0xffff;  // 16 bit short for polynomial
    int outbyte, i;

    if (inbyte == COMMA)   // if this is a comma
    {
       lfsr = 0xffff;      // reset the LFSR
       return (COMMA);     // and return the same data
    }

    if (inbyte == SKIP)    // don't advance or encode on skip
        return (SKIP);

    for (i=0; i<16;i++)    // convert the LFSR to bit array for legibility
        bit[i] = (lfsr >> i) & 1;

    for (i=0; i<8; i++)    // convert byte to be de-scrambled for legibility
        descrambit[i] = (inbyte >> i) & 1;

    // apply the xor to the data
    if (! (inbyte & 0x100) &&    // if not a KCODE, scramble the data
         ! (TrainingSequence == TRUE))  // and if not in the middle of

    {                                   // a training sequence
        descrambit[0] ^= bit[15];
        descrambit[1] ^= bit[14];
        descrambit[2] ^= bit[13];
        descrambit[3] ^= bit[12];
        descrambit[4] ^= bit[11];
        descrambit[5] ^= bit[10];
        descrambit[6] ^= bit[9];
        descrambit[7] ^= bit[8];
    }

    // Now advance the LFSR 8 serial clocks
    bit_out[ 0] = bit[ 8];
    bit_out[ 1] = bit[ 9];
    bit_out[ 2] = bit[10];
    bit_out[ 3] = bit[11] ^ bit[ 8];
    bit_out[ 4] = bit[12] ^ bit[ 9] ^ bit[ 8];
    bit_out[ 5] = bit[13] ^ bit[10] ^ bit[ 9] ^ bit[ 8];
    bit_out[ 6] = bit[14] ^ bit[11] ^ bit[10] ^ bit[ 9];
    bit_out[ 7] = bit[15] ^ bit[12] ^ bit[11] ^ bit[10];
    bit_out[ 8] = bit[ 0] ^ bit[13] ^ bit[12] ^ bit[11];
    bit_out[ 9] = bit[ 1] ^ bit[14] ^ bit[13] ^ bit[12];
    bit_out[10] = bit[ 2] ^ bit[15] ^ bit[14] ^ bit[13];
    bit_out[11] = bit[ 3]            ^ bit[15] ^ bit[14];
    bit_out[12] = bit[ 4]                      ^ bit[15];
    bit_out[13] = bit[ 5];
    bit_out[14] = bit[ 6];
    bit_out[15] = bit[ 7];
    lfsr = 0;
    for (i=0; i <16; i++) // convert the LFSR back to an integer
        lfsr += (bit_out[i] << i);

    outbyte = 0;
    for (i= 0; i<8; i++) // convert data back to an integer
        outbyte += (descrambit[i] << i);

    return outbyte;
}
```

The initial 16-bit values of the LFSR for the first 128 LFSR advances following a reset are listed below:

	0, 8	1, 9	2, A	3, B	4, C	5, D	6, E	7, F
00	FFFF	E817	0328	284B	4DE8	E755	404F	4140
08	4E79	761E	1466	6574	7DBD	B6E5	FDA6	B165
10	7D09	02E5	E572	673D	34CF	CB54	4743	4DEF
18	E055	40E0	EE40	54BE	B334	2C7B	7D0C	07E5
20	E5AF	BA3D	248A	8DC4	D995	85A1	BD5D	4425
28	2BA4	A2A3	B8D2	CBF8	EB43	5763	6E7F	773E
30	345F	5B54	5853	5F18	14B7	B474	6CD4	DC4C
38	5C7C	70FC	F6F0	E6E6	F376	603B	3260	64C2
40	CB84	9743	5CBF	B3FC	E47B	6E04	0C3E	3F2C
48	29D7	D1D1	C069	7BC0	CB73	6043	4A60	6FFA
50	F207	1102	01A9	A939	2351	566B	6646	4FF6
58	F927	3081	85B0	AC5D	478C	82EF	F3F2	E43B
60	2E04	027E	7E72	79AE	A501	1A7D	7F2A	2197
68	9019	0610	1096	9590	8FCD	D0E7	F650	46E6
70	E8D6	C228	3AB2	B70A	129F	9CE2	FC3C	2B5C
78	5AA3	AF6A	70C7	CDF0	E3D5	C0AB	B9C0	D9C1

An 8-bit value of 0 repeatedly encoded with the LFSR after reset produces the following consecutive 8-bit values:

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	FF	17	C0	14	B2	E7	02	82	72	6E	28	A6	BE	6D	BF	8D
10	BE	40	A7	E6	2C	D3	E2	B2	07	02	77	2A	CD	34	BE	E0
20	A7	5D	24	B1	9B	A1	BD	22	D4	45	1D	D3	D7	EA	76	EE
30	2C	DA	1A	FA	28	2D	36	3B	3A	0E	6F	67	CF	06	4C	26
40	D3	E9	3A	CD	27	76	30	FC	94	8B	03	DE	D3	06	52	F6
50	4F	88	80	95	C4	6A	66	F2	9F	0C	A1	35	E2	41	CF	27
60	74	40	7E	9E	A5	58	FE	84	09	60	08	A9	F1	0B	6F	62
70	17	43	5C	ED	48	39	3F	D4	5A	F5	0E	B3	C7	03	9D	9B
80	8B	0D	8E	5C	33	98	77	AE	2D	AC	0B	3E	DA	0B	42	7A
90	7C	D1	CF	A8	1C	12	EE	41	C2	3F	38	7A	0D	69	F4	01
A0	DA	31	72	C5	A0	D7	93	0E	DC	AF	A4	55	E7	F0	72	16
B0	68	D5	38	84	DD	00	CD	18	9E	CA	30	59	4C	75	1B	77
C0	31	C5	ED	CF	91	64	6E	3D	FE	E8	29	04	CF	6C	FC	C4
D0	0B	5E	DA	62	BA	5B	AB	DF	59	B7	7D	37	5E	E3	1A	C6
E0	88	14	F5	4F	8B	C8	56	CB	D3	10	42	63	04	8A	B4	F7
F0	84	01	A0	01	83	49	67	EE	3E	2A	8B	A4	76	AF	14	D5
100	4F	AC	60	B6	79	D6	62	B7	43	E7	E5	2A	40	2C	6E	7A
110	56	61	63	20	6A	97	4A	38	05	E5	DD	68	0D	78	4C	53
120	8B	D6	86	57	B2	AA	1A	80	18	DC	BA	FC	03	A3	4B	30

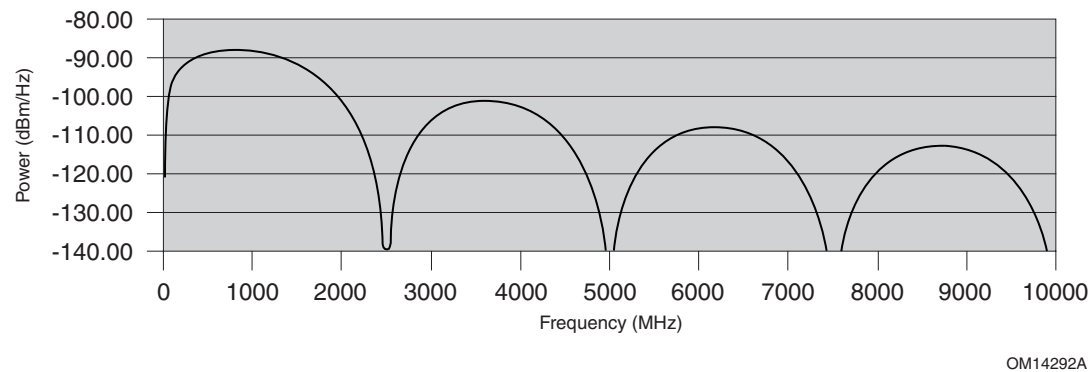Scrambling produces the power spectrum (in the 10-bit domain) shown in Figure C-1.

OM14292A

Figure C-1: Scrambling Spectrum for Data Value of 0