5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

• 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 contain different Traffic Class labels.

[**1.5.4.2**](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:

• 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

• Error checking

• TLP acknowledgement and retry Messages

• Error indication for error reporting and logging

[**1.5.4.3**](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

• Lane polarity inversion

Symbol and special Ordered Set generation:

• 8b/10b 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 Design For Testability (DFT) support features:

• Compliance pattern

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

• Modified Compliance pattern

[**1.5.4.4**](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

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

**Transaction Layer Specification**

**2.1 Transaction Layer Overview**

**2.**

|  |  |  |  |
| --- | --- | --- | --- |
| Trans | | action | |
|  | |  |  |
| Data | | Link | |
|  | |  |  |
| Phy | | sical | |
| Logical S | | ub-block |  |
|  | Electrical | Sub-block |
| RX | | TX | |

|  |  |  |
| --- | --- | --- |
| Trans | | action |
| Data | | Link |
| Phy | | sical ub-block  Sub-block  TX |
|  | Logical S |
|  | Electrical |
| RX | |

OM14295

Figure 2-1 Layering Diagram Highlighting the Transaction Layer

At a high level, the key aspects of 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 transaction-level mechanisms with device resources including:

◦ Flow Control

◦ Virtual Channel management

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

• 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.

**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, and different Transaction types are defined, each with its own unique intended usage, as shown in [Table 2-1 .](#bookmark1)

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 Function 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 formats are described later in this chapter.

[**2.1.1.1**](2.1.1.1) **Memory Transactions**

Memory Transactions include the following types:

• Read Request/Completion

• Write Request

• AtomicOp Request/Completion

Memory Transactions use two different address formats:

• Short Address Format: 32-bit address

• Long Address Format: 64-bit address

Certain Memory Transactions can optionally have a PASIDTLP Prefix containing the Process Address Space ID (PASID). See Section 6.20for details.

[**2.1.1.2**](2.1.1.2) **I/O Transactions**

PCI Express supports I/O Space for compatibility with legacy devices that require their use. Future revisions of this specification may deprecate the use of I/O Space. I/O Transactions include the following types:

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

• Read Request/Completion

• Write Request/Completion

I/O Transactions use a single address format:

• Short Address Format: 32-bit address

[**2.1.1.3**](2.1.1.3) **Configuration Transactions**

Configuration Transactions are used to access configuration registers of Functions within devices. Configuration Transactions include the following types:

• Read Request/Completion

• Write Request/Completion

[**2.1.1.4**](2.1.1.4) **Message Transactions**

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

In addition to specific Messages defined in this document, PCI Express provides support for vendor-defined Messages using specified Message codes. Except for Vendor-Defined Messages that use the PCI-SIG® Vendor ID (0001h), 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 [Section 2.2.8.6)](#bookmark2).

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](#bookmark3)shows a high level serialized view of a TLP, consisting of one or more optional TLP Prefixes, a TLP header, a data payload (for some types of packets), and an optional TLP Digest.[Figure 2-3](#bookmark4)shows a more detailed view of the TLP. The following sections of this chapter define the detailed structure of the packet headers and digest.

byte

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TLP Prefixes (optional) | | | TLP  Header  H  H+1  H+2  · · · | Data Byte 0  J  J+1  J+2 | Data Payload  (included when applicable) | Data Byte N- 1 | TLP Digest (optional)  K  K+1  K+2  K+3 |
| 0 | 1 | I 2 |

OM14547A

Figure 2-2 Serial View ofa TLP

PCI Express conceptually transfers information as a serialized stream of bytes as shown in [Figure 2-2 .](#bookmark3) Note that at the byte level, information is transmitted/received over the interconnect with the left-most byte of the TLP as shown in

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

[Figure 2-2](#bookmark3)being transmitted/received first (byte 0 if one or more optional TLP Prefixes are present else byte H). Refer to Section 4.2for details on how individual bytes of the packet are encoded and transmitted over the physical media.

Detailed layouts of the TLP Prefix, TLP Header and TLP Digest (presented in generic form in [Figure 2-3)](#bookmark5) 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 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 TLP header, the most significant byte of the address field is transferred first so that it may be used for early address decode.

Byte 0

Byte H-4 Byte H

Byte J

Byte K-4 Byte K

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| +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 |
| TLP Prefix Byte 0  TLP Prefixes | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

(Optional)

 TLP Prefix Byte H-1 Header Byte 0 

Header

|  |  |
| --- | --- |
| Data Byte 0 |  |

Data

(included when applicable)

Data Byte N-1

TLP Digest (optional)

31 24 23 16 15 8 7 0

OM13756A

Figure 2-3 Generic TLP Format

Payload data within a TLP is depicted with the lowest addressed byte (byte J in [Figure 2-3)](#bookmark5) shown to the upper left.

Detailed layouts depicting data structure organization (such as the Configuration Space depictions inChapter 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

• Length for any associated data

• Transaction Descriptor, including:

◦ Transaction ID

◦ Attributes

◦ Traffic Class

• Address/routing information

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

• Byte Enables

• Message encoding

• Completion status

**2.2 Transaction Layer Protocol- Packet 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.

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 is formed. Values in such fields must be ignored by Receivers and forwarded unmodified by Switches. 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 TLP prefixes and headers contain the following fields (see[Figure 2-4](#bookmark7)):

• Fmt[2:0] - Format of TLP (see[Table 2-2](#bookmark8)) - bits 7:5 of byte 0

• Type[4:0] - Type of TLP - bits 4:0 of byte 0

Byte 0 →

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| +0  7  6  5  4  3  2  1  0 | | | +1  7  6  5  4  3  2  1  0 | +2  7  6  5  4  3  2  1  0 | +3  7  6  5  4  3  2  1  0 |
| Fmt | Type | {Fields in bytes 1 through 3 depend on Fmt and Type fields} | | | |

Figure 2-4 Fields Present in All TLPs

The Fmt field(s) indicates the presence of one or more TLP Prefixes and the Type field(s) indicates the associated TLP Prefix type(s).

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

The Fmt, Type, TD, and Length fields of the TLP Header contain all information necessary to determine the overall size of the non-prefix portion of the TLP. The Type field, in addition to defining the type of the TLP also determineshow the TLP is routed by a Switch. Different typesofTLPs are discussed in more detail in the following sections.

• Permitted Fmt[2:0] and Type[4:0] field values are shown in . 。 All other encodings are Reserved (see [Section 2.3)](#bookmark9).

• TC[2:0] - Traffic Class (see[Section 2.2.6.6](#bookmark10)) - bits [6:4] of byte 1

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

• Lightweight Notification (LN) - 1b indicates that a Memory Request is an LN Read or LN Write, or that a Completion is an LN Completion.

• TLP Hints (TH) - 1b indicates the presence of TLP Processing Hints (TPH) in the TLP header and optional [TPH](#bookmark11) [TLP Prefix](#bookmark12) (if present) - bit 0 of byte 1 (see[Section 2.2.7.1)](#bookmark13)

• Attr[1:0] - Attributes (see [Section 2.2.6.3](#bookmark14)) - bits [5:4] of byte 2

• Attr[2] - Attribute (see [Section 2.2.6.3](#bookmark15)) - bit 2 of byte 1

• TD - 1b indicates presence of TLP Digest in the form of a single Double Word (DW) at the end of the TLP (see [Section 2.2.3](#bookmark16)) - bit 7 of byte 2

• Error Poisoned (EP) - indicates the TLP is poisoned (see[Section 2.7](#bookmark17)) - bit 6 of byte 2

• Length[9:0] - Length of data payload in DW (see[Table 2-4](#bookmark18)) - 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 DW.

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

Byte 0 →

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| +0  7  6  5  4  3  2  1  0 | | +1  7  6  5  4  3  2  1  0 | | | | | | +2  7  6  5  4  3  2  1  0 | | | | | +3  7  6  5  4  3  2  1  0 |
| Fmt | Type | T9 | TC | T8 | Attr | LN | TH | TD | EP | Attr | AT | Length | |

Figure 2-5 Fields Present in All TLP Headers Table 2-2 Fmt[2:0] Field Values

|  |  |
| --- | --- |
| Fmt[2:0] | Corresponding TLP Format |
| 000b | 3 DW header, no data |
| 001b | 4 DW header, no data |
| 010b | 3 DW header, with data |
| 011b | 4 DW header, with data |
| 100b | TLP Prefix |
|  | All encodings not shown above are Reserved (see [Section 2.3)](#bookmark20). |

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

|  |  |  |  |
| --- | --- | --- | --- |
| TLP Type | Fmt [2:0]3  (b) | Type [4:0] (b) | Description |
| MRd | 000  001 | 0 0000 | Memory Read Request |
| MRdLk | 000  001 | 0 0001 | Memory Read Request-Locked |



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

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

|  |  |  |  |
| --- | --- | --- | --- |
| TLP Type | Fmt [2:0]  (b) | Type [4:0] (b) | Description |
| MWr | 010  011 | 0 0000 | Memory Write Request |
| IORd | 000 | 0 0010 | I/O Read Request |
| IOWr | 010 | 0 0010 | I/OWrite Request |
| CfgRd0 | 000 | 0 0100 | Configuration Read Type 0 |
| CfgWr0 | 010 | 0 0100 | Configuration Write Type 0 |
| CfgRd1 | 000 | 0 0101 | Configuration Read Type 1 |
| CfgWr1 | 010 | 0 0101 | Configuration Write Type 1 |
| TCfgRd | 000 | 1 1011 | Deprecated TLP Type4 |
| TCfgWr | 010 | 1 1011 | Deprecated TLP Type5 |
| Msg | 001 | 1 0r2r1r0 | Message Request - The sub-field r[2:0] specifies the Message routing mechanism (see [Table](#bookmark22) [2-17 )](#bookmark23). |
| MsgD | 011 | 1 0r2r1r0 | Message Request with data payload - The sub-field r[2:0] specifies the Message routing mechanism (see [Table 2-17)](#bookmark24). |
| Cpl | 000 | 0 1010 | Completion without Data - Used for I/O and Configuration Write Completions with any Completion Status. Also used for AtomicOp Completions and Read Completions (I/O, Configuration, or Memory) with Completion Status other than Successful Completion. |
| CplD | 010 | 0 1010 | Completion with Data - Used for Memory, I/O, and Configuration Read Completions. Also used for AtomicOp Completions. |
| CplLk | 000 | 0 1011 | Completion for Locked Memory Read without Data - Used only in error case. |
| CplDLk | 010 | 0 1011 | Completion for Locked Memory Read - Otherwise like CplD. |
| FetchAdd | 010  011 | 0 1100 | Fetch and Add AtomicOp Request |
| Swap | 010  011 | 0 1101 | Unconditional Swap AtomicOp Request |
| CAS | 010  011 | 0 1110 | Compare and Swap AtomicOp Request |
| LPrfx | 100 | 0 L3L2L1L0 | Local TLP Prefix - The sub-field L[3:0] specifies the Local TLP Prefix type (see [Table 2-36)](#bookmark25). |
| EPrfx | 100 | 1 E3E2E1E0 | End-End TLP Prefix - The sub-field E[3:0] specifies the End-End TLP Prefix type (see [Table](#bookmark26) [2-37 )](#bookmark27). |
|  |  |  | All encodings not shown above are Reserved (see [Section 2.3)](#bookmark28). |



4. Deprecated TLP Types: previously used for Trusted Configuration Space (TCS), which is no longer supported by this specification. If a Receiver does not implement TCS, the Receiver must treat such Requests as Malformed Packets.

5. Deprecated TLP Types: previously used for Trusted Configuration Space (TCS), which is no longer supported by this specification. If a Receiver does not implement TCS, the Receiver must treat such Requests as Malformed Packets.

Page 109

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

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 that explicitly refer to a data length 。 Refer to the Message Code tables in [Section 2.2.8 .](#bookmark29)

• The Transmitter of a TLP with a data payload must not allow the data payload length as given by the TLP's Length field to exceed the length specified by the value in the Max\_Payload\_Sizefield of the Transmitter's Device Control registertaken as an integral number of DW (seeSection 7.5.3.4).

。 For ARI Devices, the Max\_Payload\_Sizeis determined solely by the setting in Function 0. The Max\_Payload\_Sizesettings in other Functions are ignored.

。 For an Upstream Port associated with a non-ARI Multi-Function Device (MFD) whose

Max\_Payload\_Sizesettings are identical across all Functions, a transmitted TLP's data payload must not exceed the commonMax\_Payload\_Sizesetting.

。 For an Upstream Port associated with a non-ARI MFD whose Max\_Payload\_Sizesettings are not

identical across all Functions, a transmitted TLP's data payload must not exceed a Max\_Payload\_Size setting whose determination is implementation specific.

▪ Transmitter implementations are encouraged to use theMax\_Payload\_Sizesetting from the Function that generated the transaction, or else the smallest Max\_Payload\_Sizesetting

across all Functions.

▪ Software should not set theMax\_Payload\_Sizein different Functions to different values unless software is aware of the specific implementation.

。 Note:Max\_Payload\_Sizeapplies 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 controlled by the Length field.

• The size of the data payload of a Received TLP as given by the TLP's Length field must not exceed the length specified by the value in theMax\_Payload\_Sizefield of the Receiver's Device Control registertaken as an

integral number of DW (seeSection 7.5.3.4).

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

▪ This is a reported error associated with the Receiving Port (seeSection 6.2).

。 For ARI Devices, the Max\_Payload\_Sizeis determined solely by the setting in Function 0. The Max\_Payload\_Sizesettings in other Functions are ignored.

。 For an Upstream Port associated with a non-ARI MFD whose Max\_Payload\_Sizesettings are identical across all Functions, the Receiver is required to check the TLP's data payload size against the

commonMax\_Payload\_Sizesetting.

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

。 For an Upstream Port associated with a non-ARI MFD whose Max\_Payload\_Sizesettings are not identical across all Functions, the Receiver is required to check the TLP's data payload against a Max\_Payload\_Sizesetting whose determination is implementation specific.

▪ Receiver implementations are encouraged to use theMax\_Payload\_Sizesetting from the

Function targeted by the transaction, or else the largestMax\_Payload\_Sizesetting across all Functions.

▪ Software should not set theMax\_Payload\_Sizein different Functions to different values unless software is aware of the specific implementation.

• 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.

▪ This is a Reported Error associated with the Receiving Port (seeSection 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 associated with a byte address is included in a TLP other than an AtomicOp Request or an AtomicOp Completion, 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.

。 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.

• The data payload in AtomicOp Requests and AtomicOp Completions must be formatted such that the first byte of data following the TLP header is the least significant byte of the first data value, and subsequent bytes of

data are strictly increasing insignificance. With Compare And Swap (CAS) Requests, the second data value immediately follows the first data value, and must be in the same format.

。 The endian format used by AtomicOp Completers to read and write data at the target location is

implementation specific, and is permitted to be whatever the Completer determines is appropriate for the target memory (e.g., little endian, big endian, etc.) Endian format capability reporting and controls for AtomicOp Completers are outside the scope of this specification.

。 Little endian example: For a 64-bit (8-byte) Swap Request targeting location 100h with the target

memory in little endian format, the first byte following the header is written to location 100h, the second byte is written to location 101h, and so on,with the final byte written to location 107h. Note that before performing the writes, the Completer first reads the target memory locations so it can return the original value in the Completion. The byte address correspondence to the data in the

Completion is identical to that in the Request.

。 Big endian example: For a 64-bit (8-byte) Swap Request targeting location 100h with the target

memory in big endian format, the first byte following the header is written to location 107h, the

second byte is written to location 106h, and so on,with the final byte written to location 100h. Note that before performing the writes, the Completer first reads the target memory locations so it can return the original value in the Completion. The byte address correspondence to the data in the

Completion is identical to that in the Request.

。[Figure 2-6](#bookmark30)shows little endian and big endian examples of Completer target memory access for a

64-bit (8-byte) FetchAdd. The bytes in the operands and results are numbered 0-7,with byte 0 being least significant and byte 7 being most significant. In each case, the Completer fetches the target

memory operand using the appropriate endian format. Next, AtomicOp compute logic in the

Completer performs the FetchAdd operation using the original target memory value and the “add” value from the FetchAdd Request. Finally, the Completer stores the FetchAdd result back to target memory using the same endian format used for the fetch.

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

FetchAdd example with target memory in little endian format

|  |  |
| --- | --- |
|  |  |
| 100h | |
| 101h 102h | |
| 103h | |
| 104h 105h | |
| 106h | |
| 107h | |
|  |  |

y v target memory locations

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |



|  |
| --- |
| 7 6 5 4 3 2 1 0 |

original value

logic

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |



AtomicOp compute

FetchAdd result

"add" value

FetchAdd example with target memory in big endian format

|  |  |
| --- | --- |
|  |  |
| 100h | |
| 101h 102h | |
| 103h | |
| 104h 105h | |
| 106h | |
| 107h | |
|  |  |

y v target memory locations

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |



|  |
| --- |
| 7 6 5 4 3 2 1 0 |

original value

logic



AtomicOp compute

FetchAdd result

"add" value

A-0742

Figure 2-6 Examples of Completer Target Memory Access for FetchAdd

**IMPLEMENTATION NOTE**

Endian Format Support by RC AtomicOp Completers

One key reason for permitting an AtomicOp Completer to access target memory using an endian format of its

choice is so that PCI Express devices targeting host memory with AtomicOps can interoperate with host software that uses atomic operation instructions (or instruction sequences). Some host environments have limited endian format support with atomic operations, and by supporting the “right” endian format(s), an RC AtomicOp

Completer may significantly improve interoperability.

For an RC with AtomicOp Completer capability on a platform supporting little-endian-only processors, there is

little envisioned benefit for the RC AtomicOp Completer to support any endian format other than little endian. For an RC with AtomicOp Completer capability on a platform supporting bi-endian processors, there may be benefit in supporting both big endian and little endian formats, and perhaps having the endian format configurable for different regions of host memory.

There is no PCI Express requirement that an RC AtomicOp Completer support the host processor's “native” format (if there is one), nor is there necessarily significant benefit to doing so. For example, some processors can use

load-link/store-conditional or similar instruction sequences to do atomic operations in non-native endian formats and thus not need the RC AtomicOp Completer to support alternative endian formats.



5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

**IMPLEMENTATION NOTE**

Maintaining Alignment in Data Payloads

[Section 2.3.1.1](#bookmark31)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 bit indicates the presence of the TLP Digest field including an end-to-end CRC (ECRC) value at the end of the TLP.

。 ATLP where the TD bit 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 (seeSection 6.2).

• If an intermediate or ultimate PCI Express Receiver of the TLP does not support ECRC checking, the Receiver must ignore the TLP Digest6 .

。 If the Receiver of the TLP supports ECRC checking, the Receiver interprets the value in the TLP Digest field as an ECRC value, according to the rules in [Section 2.7.1 .](#bookmark32)

**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, andis covered in [Section 2.2.8](#bookmark34)

.

[**2.2.4.1**](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 2-7)](#bookmark36) and a 32-bit format used with a 3 DW header (see[Figure 2-8)](#bookmark37).



6. An exception is an Intermediate Receiver forwarding a Multicast TLP out an Egress Port with MC\_Overlay enabled. See Section 6.14.5 .

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

Byte 0 → Byte 4 → Byte 8 → Byte 12 →

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| +0  7  6  5  4  3  2  1  0 | | +1  7  6  5  4  3  2  1  0 | | | | | | | | +2  7  6  5  4  3  2  1  0 | | | | | +3  7  6  5  4  3  2  1  0 | |
| Fmt  0  x  1 | Type | T9 | | TC | T8 | | Attr | LN | TH | TD | EP | Attr | AT | Length | | |
|  | | | {fields in bytes 4 through 7 depend on type of Request} | | | | | | | | | | | | | |
|  | | | | | | Address[63:32] | | | | | | | | | | |
| Address[31:2] | | | | | | | | | | | | | | | | PH |

Figure 2-7 64-bit Address Routing

Byte 0 → Byte 4 → Byte 8 →

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| +0  7  6  5  4  3  2  1  0 | | +1  7  6  5  4  3  2  1  0 | | | | | | | +2  7  6  5  4  3  2  1  0 | | | | | +3  7  6  5  4  3  2  1  0 | |
| Fmt  0  x  0 | Type | T9 | | TC | T8 | Attr | LN | TH | TD | EP | Attr | AT | Length | | |
|  | | | {fields in bytes 4 through 7 depend on type of Request} | | | | | | | | | | | | |
| Address[31:2] | | | | | | | | | | | | | | | PH |

Figure 2-8 32-bit Address Routing

• For Memory Read, Memory Write, and AtomicOp Requests, the Address Type (AT) field is encoded as shown in Table 10-1. For all other Requests, the AT field is Reserved unless explicitly stated otherwise. LN Reads and LN Writes have special requirements. See Section 6.21.5 .

• If TH is Set, the PH field is encoded as shown in [Table 2-15 .](#bookmark38) If TH is Clear, the PH field is Reserved.

• Address mapping to the TLP header is shown in [Table 2-5 .](#bookmark39)

Table 2-5 Address Field Mapping

|  |  |  |
| --- | --- | --- |
| Address Bits | 32-bit Addressing | 64-bit Addressing |
| 63:56 | Not Applicable | Bits 7:0 of Byte 8 |
| 55:48 | 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 |

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

• Memory Read, Memory Write, and AtomicOp Requests can use either format.

。 For Addresses below 4 GB, Requesters must use the 32-bit format. The behavior of the Receiver is not specified if a 64-bit format request addressing below 4 GB (i.e., with the upper 32 bits of address all 0) is received.

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

• All 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**](2.2.4.2) **ID Based Routing Rules**

• ID routing is used with Configuration Requests, with ID Routed Messages, and with Completions. This

specification defines several Messages that are ID Routed (Table F-1 ). Other specifications are permitted to define additional ID Routed Messages.

• ID routing uses the Bus, Device, and Function Numbers (as applicable) to specify the destination for the TLP:

。 For non-ARI Routing IDs, Bus, Device, and (3-bit) Function Number to TLP header mapping is shown in [Table 2-6 ,](#bookmark41)[Figure 2-9 ,](#bookmark42) and [Figure 2-11 .](#bookmark43)

。 For ARI Routing IDs, the Bus and (8-bit) Function Number to TLP header mapping is shown in [Table](#bookmark44) [2-7 ,](#bookmark45)[Figure 2-10 ,](#bookmark46) and [Figure 2-12 .](#bookmark47)

• Two ID routing formats are specified, one used with a 4 DW header (see[Figure 2-9](#bookmark48)and [Figure 2-10)](#bookmark49) and one used with a 3 DW header (see[Figure 2-12](#bookmark50)and [Figure 2-10)](#bookmark51).

。 Header field locations are the same for both formats (see[Figure 2-5)](#bookmark19).

Table 2-6 Header Field Locations for non-ARI 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 |

Table 2-7 Header Field Locations for ARI ID Routing

|  |  |
| --- | --- |
| Field | Header Location |
| Bus Number[7:0] | Bits 7:0 of Byte 8 |
| Function Number[7:0] | Bits 7:0 of Byte 9 |

Byte 0 → Byte 4 → Byte 8 → Byte 12 →

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| +0  7  6  5  4  3  2  1  0 | | +1  7  6  5  4  3  2  1  0 | | | | | | | +2  7  6  5  4  3  2  1  0 | | | | | +3  7  6  5  4  3  2  1  0 |
| Fmt  0  x  1 | Type | T9 | | TC | T8 | Attr | LN | TH | TD | EP | Attr | AT | Length | |
|  | | | {fields in bytes 4 through 7 depend on type of Request} | | | | | | | | | | | |
| Bus Number | | Device Num | | | | Fcn Num | | | {fields in bytes 10 and 11 depend on type of Request} | | | | | |
| {fields in bytes 12 through 15 depend on type of Request} | | | | | | | | | | | | | | |

Figure 2-9 Non-ARI ID Routing with 4 DW Header

Byte 0 → Byte 4 → Byte 8 → Byte 12 →

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| +0  7  6  5  4  3  2  1  0 | | +1  7  6  5  4  3  2  1  0 | | | | | | | +2  7  6  5  4  3  2  1  0 | | | | | +3  7  6  5  4  3  2  1  0 |
| Fmt  0  x  1 | Type | T9 | | TC | T8 | Attr | LN | TH | TD | EP | Attr | AT | Length | |
|  | | | {fields in bytes 4 through 7 depend on type of Request} | | | | | | | | | | | |
| Bus Number | | Function Number | | | | | | | {fields in bytes 10 and 11 depend on type of Request} | | | | | |
| {fields in bytes 12 through 15 depend on type of Request} | | | | | | | | | | | | | | |

Figure 2-10 ARI ID Routing with 4 DW Header

Byte 0 → Byte 4 → Byte 8 →

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| +0  7  6  5  4  3  2  1  0 | | +1  7  6  5  4  3  2  1  0 | | | | | | | +2  7  6  5  4  3  2  1  0 | | | | | +3  7  6  5  4  3  2  1  0 |
| Fmt  0  x  0 | Type | T9 | | TC | T8 | Attr | LN | TH | TD | EP | Attr | AT | Length | |
|  | | | {fields in bytes 4 through 7 depend on type of Request} | | | | | | | | | | | |
| Bus Number | | Device Num | | | | Fcn Num | | | {fields in bytes 10 and 11 depend on type of Request} | | | | | |

Figure 2-11 Non-ARI ID Routing with 3 DW Header

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

Byte 0 → Byte 4 → Byte 8 →

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| +0  7  6  5  4  3  2  1  0 | | +1  7  6  5  4  3  2  1  0 | | | | | | | +2  7  6  5  4  3  2  1  0 | | | | | +3  7  6  5  4  3  2  1  0 |
| Fmt  0  x  0 | Type | T9 | | TC | T8 | Attr | LN | TH | TD | EP | Attr | AT | Length | |
|  | | | {fields in bytes 4 through 7 depend on type of Request} | | | | | | | | | | | |
| Bus Number | | Function Number | | | | | | | {fields in bytes 10 and 11 depend on type of Request} | | | | | |

Figure 2-12 ARI 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 2-13)](#bookmark53). For Memory

Read Requests that have the TH bit Set, the Byte Enable fields are repurposed to carry the ST[7:0] field (refer to[Section](#bookmark54) [2.2.7.1 f](#bookmark55)or details), and values for the Byte Enables are implied as defined below. The TH bit must only be Set in Memory Read Requests when it is acceptable to complete those Requests as if all bytes for the requested data were enabled.

• For Memory Read Requests that have the TH bit Set, the following values are implied for the Byte Enables. See [Section 2.2.7f](#bookmark56)or additional requirements.

。 If the Length field for this Request indicates a length of 1 DW, then the value for the First DW Byte Enables is implied to be 1111band the value for the Last DW Byte Enables is implied to be 0000b.

。 If the Length field for this Request indicates a length of greater than 1 DW, then the value for the First DW Byte Enables and the Last DW Byte Enables is implied to be 1111b.

Byte 0 → Byte 4 →

**IMPLEMENTATION NOTE**

Read Request with TPH to Non-Prefetchable Space

Memory Read Requests with the TH bit Set and that target Non-Prefetchable Memory Space should only be

issued when it can be guaranteed that completion of such reads will not create undesirable side effects. See

Section 7.5.1.2.1for consideration of certain BARs that may have the Prefetchable bit Set even though they map some locations with read side-effects.



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| +0  7  6  5  4  3  2  1  0 | | | +1  7  6  5  4  3  2  1  0 | | | | | | +2  7  6  5  4  3  2  1  0 | | | | | +3  7  6  5  4  3  2  1  0 | |
| Fmt  0  x  x | Type | | T9 | TC | T8 | Attr | LN | TH | TD | EP | Attr | AT | Length | | |
|  | | {fields in bytes 4 through 6 depend on type of Request} | | | | | | | | | | | | Last DW BE | First DW BE |

Figure 2-13 Location of Byte Enables in TLP Header

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

• The First 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.

。 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-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 First DW BE field for all Requests with length of 1 DW.

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

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

• All non-QW aligned Memory Requests with length of 2 DW (1 QW) and 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.

。 Contiguous Byte Enables examples:

First DW BE: 1100b, Last DW BE: 0011b First DW BE: 1000b, Last DW BE: 0111b

• [Table 2-8](#bookmark57)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 2-8 Byte Enables Location and Correspondence

|  |  |  |
| --- | --- | --- |
| Byte Enables | Header Location | Affected Data Byte7 |
| First DW BE[0] | Bit 0 of Byte 7 | Byte 0 |
| First DW BE[1] | Bit 1 of Byte 7 | Byte 1 |
| First DW BE[2] | Bit 2 of Byte 7 | Byte 2 |
| First 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 |

• A Write Request with a length of 1 DW with no bytes enabled is permitted, and has no effect at the Completer unless otherwise specified.



7. 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.

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

**IMPLEMENTATION NOTE**

Zero-Length Write

A Memory Write Request of 1 DW with no bytes enabled, or “zero-length Write,” may be used by devices under certain protocols,in order to achieve an intended side effect. One example is LN protocol. See Section 6.21 .



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

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. If a Receiver implementing such checks determines that a TLP violates one or more Byte Enables rules, the TLP is a

Malformed TLP. These checks are independently optional (see Section 6.2.3.4).

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

**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. To be effective in all cases, the address for the zero-length Read must target the same device as the Posted Writes that are being flushed. One recommended approach is using the same address as one of the Posted Writes being flushed.



The flush semantic has wide application, and all Completers must implement the functionality associated with this

semantic. Since 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**](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

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

[Figure 2-14](#bookmark58)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.

Transaction ID

|  |
| --- |
| 9:0 |

|  |
| --- |
| 2:0 |

|  |
| --- |
| 2:0 |

|  |
| --- |
| 15:0 |

Requester ID Tag Attributes Traffic

Class

OM13757B

Figure 2-14 Transaction Descriptor

[**2.2.6.2**](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 2-15 .](#bookmark59)

Requester ID

|  |
| --- |
|  |
| 15:0 |

Tag

|  |
| --- |
| 9:0 |

OM13758B

Figure 2-15 Transaction ID

10-Bit Tag capability, introduced in [PCIe-4.0] increases the total Tag field size from 8 bits to 10 bits. The two additional Tag bits, Tag[8] (T8) and Tag[9] (T9), are not contiguous with other Tag[7:0] bits in the TLP Header. The two additional bits were Reserved in previous versions of this specification.

• Tag[9:0] is a 10-bit field generated by each Requester, and it must be unique for all outstanding Requests that require a Completion for that Requester. Requesters that do not support 10-Bit Tag Requester capability must set Tag[9:8] to 00b.

◦ Functions8 (including those in Switches) that support 16.0 GT/s data rates or greater must support 10-Bit Tag Completer capability. If a Function supports 10-Bit Tag Completer capability, it may

optionally support 10-Bit Tag Requester capability. See Section 7.5.3.15and the "Considerations for Implementing 10-Bit Tag Capabilities" Implementation Note later in this section.

◦ RCs containing elements that indicate support for 10-Bit Tag Completer capability must handle 10-Bit Tag Requests correctly by all registers and memory regions supported as targets of PCIe Requesters; e.g., host memory targeted by DMA Requests or MMIO regions in RCiEPs.

▪ Each RP indicating support must handle such Requests received by its Ingress Port.



8. An exception is PCI Express to PCI/PCI-X Bridges, since 10-Bit Tag capability is not architected for these Functions.

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

. Each RCiEP indicating support must handle such Requests coming from supported internal paths,including those coming through RPs.

◦ If an RC contains RCiEPs that indicate support for 10-Bit Tag Requester capability, the RC must handle 10-Bit Tag Requests from those RCiEPs correctly by all registers and memory regions supported as

targets of those RCiEPs; e.g., host memory targeted by DMA Requests or MMIO regions in RCiEPs.

◦ Receivers/Completers must handle 8-bit Tag values correctly regardless of the setting of their Extended Tag Field Enable bit (see Section 7.5.3.4). Refer to the PCI Express to PCI/PCI-X Bridge Specification for details on the bridge handling of extended tags.

◦ Receivers/Completers that support 10-Bit Tag Completer capability must handle 10-Bit Tag values correctly, regardless of their 10-Bit Tag Requester Enable bit setting. See Section 7.5.3.16 .

◦ 10-Bit Tag capability is not architected for PCI Express to PCI/PCI-X Bridges, and they must not indicate 10-Bit Tag Requester capability or 10-Bit Tag Completer capability.

◦ If the 10-Bit Tag Requester Enable bit is Clear and the Extended Tag Field Enable bit is Clear, the

maximum number of outstanding Requests per Functionshall be limited to 32, and only the lower 5 bits of the Tag field are used with the remaining upper 5 bits required to be 0 0000b.

◦ If the 10-Bit Tag Requester Enable bit is Clear and the Extended Tag Field Enable bit is Set, the

maximum is increased to 256, and only the lower 8 bits of the Tag field are used with the remaining upper 2 bits required to be 00b.

◦ If the 10-Bit Tag Requester Enable bit is Set, the maximum targeting a single Completer is increased up to 768. The Requester is permitted to use all 10 bits of the Tag field when sending 10-Bit Tag

Requests to Completers it deems suitable, though the Requester is still permitted to send smaller-Tag Requests to other Completers. The following apply to 10-Bit Tag capable Requesters whose 10-Bit Tag Requester Enable bit is Set.

. If an Endpoint9 supports sending Requests to other Endpoints (as opposed to host

memory), the Endpoint must not send 10-Bit Tag Requests to another given Endpoint

unless an implementation-specific mechanism determines that the Endpoint supports

10-Bit Tag Completer capability. Not sending 10-Bit Tag Requests to other Endpoints at all may be acceptable for some implementations. More sophisticated mechanisms are outside the scope of this specification.

. If a PIO Requester has 10-Bit Tag Requester capability, how the Requester determines when to use 10-Bit Tags versus smaller Tags is outside the scope of this specification.

. With 10-Bit Tags, valid Tag[9:8] values are 01b, 10b, or 11b. 10-Bit Tag values with Tag[9:8] equal to 00b are invalid, and must not be generated by the Requester. This enables a

Requester to determine if a Completion it receives that should have a 10-Bit Tag contains an invalid one, usually caused by the Completer not supporting 10-Bit Tag Completer

capability.

. If a Requester sends a 10-Bit Tag Request to a Completer that lacks 10-Bit Completer

capability, the returned Completion(s) will have Tags with Tag[9:8] equal to 00b. Since the Requester is forbidden to generate these Tag values for 10-Bit Tags, such Completions will be handled as Unexpected Completions10 , which by default are Advisory Non-Fatal Errors. The Requester must follow standard PCI Express error handling requirements.

. When a Requester handles a Completion with an invalid 10-Bit Tag as an Unexpected

Completion, the original Request will likely incur a Completion Timeout. If the Requester handles the Completion Timeout condition in some device-specific manner that avoids

data corruption, the Requester is permitted to suppress handling the Completion Timeout by standard PCI Express error handling mechanisms as required otherwise.



9. This includes PCI Express Endpoints, Legacy PCI Express Endpoints, and Root Complex Integrated Endpoints.

10. If a Completion has a higher precedence error, that error should be reported instead.

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

. If a Requester supports sending 10-Bit Tag Requests to some Completers and smaller-Tag Requests to other Completers concurrently, the Requester must honor the Extended Tag

Field Enable bit setting for the smaller-Tag Requests. That is, if the bit is Clear, only the

lower 5 bits of the Tag field may be non-zero; if the bit is Set, only the lower 8 bits of the Tag field may be non-zero.

. If a Requester supports sending 10-Bit Tag Requests to some Completers and smaller-Tag Requests to other Completers concurrently, the Requester must ensure that no outstanding 10-Bit Tags can alias to an outstanding smaller Tag if any 10-Bit Tag Request is completed by a Completer that lacks 10-Bit Tag Completer capability. See the "Using 10-Bit Tags and Smaller Tags Concurrently" Implementation Note later in this section.

◦ The default value of the Extended Tag Field Enable bit is implementation specific. The default value of the 10-Bit Tag Requester Enable bit is 0b.

◦ Receiver/Completer behavior is undefined if multiple uncompleted 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 Posted Requests, the Tag [9:8] field is Reserved.

• For Posted Requests with the TH bit Set, the Tag[7:0] field is repurposed for the ST[7:0] field (refer to[Section](#bookmark60)

[2.2.7.1 f](#bookmark61)or details). For Posted Requests with the TH bit Clear, the Tag[7:0] field is undefined and may contain any value. (Refer toTable F-1for exceptions to this rule for certain Vendor\_Defined Messages.)

◦ For Posted Requests with the TH field Clear, the value in the Tag[7:0] field must not affect Receiver processing of the Request.

◦ For Posted Requests with the TH bit Set, the value in the ST[7:0] field may affect Completer processing of the Request (refer to <2.2.7.1> for details).

• 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 Numbers11 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 Requester ID12 for all Requests initiated by the Device/Function. It is recommended that Numbers are captured for

successfully completed Requests only.

Exception: The assignment of Bus and Device Numbers to the Devices within a Root Complex, and Device Numbers to the Downstream Ports within a Switch, may be done in an implementation specific way.

Note that the Bus Number and Device Number13 may be changed at run time, and so it is necessary to re-capture this information with each and every Configuration Write Request.

It is recommended that Configuration Write Requests addressed to unimplemented Functions not affect captured Bus and Device Numbers.

• When generating Requests on their own behalf (for example, for error reporting), Switches must use the

Requester ID associated with the primary side of the bridge logically associated with the Port (seeSection 7.1) causing the Request generation.



11. InARI Devices, Functions are only required to capture the Bus Number. ARI Devices are permitted to retain the captured Bus Number on either a per-Device or

a per-Function basis. If the captured Bus Number is retained on a per-Device basis, all Functions are required to update and use the common Bus Number.

12.AnARI Requester ID does not contain a Device Number field. See[Section 2.2.4.2 .](#bookmark40)

13. With ARI Devices, only the Bus Number can change.

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

• Prior to the initial Configuration Write to a Function, the Function is not permitted to initiate Non-Posted Requests. (A valid Requester ID is required to properly route the resulting completions.)

◦ Exception: Functions 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 Device must be designed to respond to a unique Function Number for

Configuration Requests addressing that Device. Note: Each non-ARI Device may contain up to eight Functions. Each ARI Device may contain up to 256 Functions.

• A Switch must forward Requests without modifying the Transaction ID.

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

**IMPLEMENTATION NOTE**

Increasing the Number of Outstanding Requests using Phantom Functions

To increase the maximum possible number of outstanding Requests requiring Completion beyond that possible using Tag bits alone, a device may, if the Phantom Functions Enable bit is Set (seeSection 7.5.3.4), use Function Numbers not assigned to implemented Functions to logically extend the Tag identifier. 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.

Phantom Functions have a number of architectural limitations, including a lack of support by ARI Devices,Virtual Functions (VFs), and Physical Functions (PFs) when VFs are enabled. In addition, Address Translation Services

(ATS) and[ID-Based Ordering](#bookmark62)(IDO) do not comprehend Phantom Functions. Thus, for many implementations, the use of 10-Bit Tags is a better way to increase the number of outstanding Non-Posted Requests.



5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0



**IMPLEMENTATION NOTE**

Considerations for Implementing 10-Bit Tag Capabilities

The use of 10-Bit Tags enables a Requester to increase its number of outstanding Non-Posted Requests (NPRs) from 256 to 768, which for very high rates of NPRs can avoid Tag availability from becoming a bottleneck. The following formula gives the basic relationship between payload bandwidth, number of outstanding NPRs, and other factors:

BW = S \* N / RTT, where

**BW** = payload bandwidth

**S** = transaction payload size

**N** = number of outstanding NPRs **RTT** = transaction round-trip time

Generally only high-speed Requesters on high-speed Links using relatively small transactions will benefit from increasing their number of outstanding NPRs beyond 256, although this can also help maintain performance in configurations where the transaction round-trip time is high.

In configurations where a Requester with 10-Bit Tag Requester capability needs to target multiple Completers, one needs to ensure that the Requester sends 10-Bit Tag Requests only to Completers that have 10-Bit Tag

Completer capability. This is greatly simplified if all Completers have this capability.

For general industry enablement of 10-Bit Tags, it is highly recommended that all Functions14 support 10-Bit Tag Completer capability. With new implementations, Completers that don't need to operate on higher numbers of NPRs concurrently themselves can generally track 10-Bit Tags internally and return them in Completions with modest incremental investment.

Completers that actually process higher numbers of NPRs concurrently may require substantial additional hardware resources, but the full performance benefits of 10-Bit Tags generally can't be realized unless

Completers actually do process higher numbers of NPRs concurrently.

For platforms where the RC supports 10-Bit Tag Completer capability, it is highly recommended for platform firmware or operating software that configures PCIe hierarchies to Set the 10-Bit Tag Requester Enable bit

automatically in Endpoints with 10-Bit Tag Requester capability. This enables the important class of 10-Bit Tag capable adapters that send Memory Read Requests only to host memory.

For Endpoints other than RCiEPs, one can determine if the RC supports 10-Bit Tag Completer capability for each one by checking the 10-Bit Tag Completer Supported bit in its associated RP. RCiEPs have no associated RP, so for this reason they are not permitted to have their 10-Bit Tag Requester Supported bit Set unless the RC supports

10-Bit Tag Completer capability for them. Thus, software does not need to perform a separate check for RCiEPs.

Switches that lack 10-Bit Tag Completer capability are still able to forward NPRs and Completions carrying 10-Bit Tags correctly, since the two new Tag bits are in TLP Header bits that were formerly Reserved, and Switches are required to forward Reserved TLP Header bits without modification. However, if such a Switch detects an error

with an NPR carrying a 10-Bit Tag, and that Switch handles the error by acting as the Completer for the NPR, the resulting Completion will have an invalid 10-Bit Tag. Thus, it is strongly recommended that Switches between any components using 10-Bit Tags support 10-Bit Tag Completer capability. Note that Switches supporting 16.0 GT/s data rates or greater must support 10-Bit Tag Completer capability.

For configurations where a Requester with 10-Bit Tag Requester capability targets Completers where some do and some do not have 10-Bit Tag Completer capability, how the Requester determines which NPRs include 10-Bit Tags is outside the scope of this specification.



14. An exception is PCI Express to PCI/PCI-X Bridges, since 10-Bit Tag capability is not architected for these Functions.

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

**IMPLEMENTATION NOTE**

Using 10-Bit Tags and Smaller Tags Concurrently

As stated earlier in this section, if a Requester supports sending 10-Bit Tag Requests to some Completers and smaller-Tag Requests to other Completers concurrently, the Requester must ensure that no outstanding 10-Bit Tags can alias to an outstanding smaller Tag if any 10-Bit Tag Request is completed by a Completer that lacks 10-Bit Tag Completer capability.

One implementation approach is to have the Requester partition its 8-bit Tag space into 2 regions: one that will

only be used for smaller Tags (8-bit or 5-bit Tags), and one that will only be used for the lower 8 bits of 10-Bit Tags. Note that this forces a tradeoff between the Tag space available for 10-Bit Tags and smaller Tags.

For example, if a Requester partitions its 8-bit Tag space to use only the lowest 4 bits for smaller Tags, this

supports up to 16 outstanding smaller Tags, and it reduces the 10-Bit Tag space by 3\*16 values, supporting

768-48=720 outstanding 10-bit Tags. Many other partitioning options are possible, all of which reduce the total number of outstanding Requests. In general, reserving N values for smaller Tags reduces 10-Bit Tag space by 3\*N values,and the total for smaller Tags plus 10-Bit Tags ends up being 768 - 2\*N.



[**2.2.6.3**](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:

• 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. Note that attribute bit 2 is not adjacent to bits 1 and 0 (see [Figure 2-17](#bookmark63)and [Figure](#bookmark64) [2-18 )](#bookmark65).

Attr[2]

|  |
| --- |
|  |

ID-Based

Ordering

Attr[1:0]

|  |  |
| --- | --- |
|  |  |

Relaxed No

Ordering Snoop

OM13759B

Figure 2-16 Attributes Field of Transaction Descriptor

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

[**2.2.6.4**](2.2.6.4) **Relaxed Ordering andID-Based Ordering Attributes**

[Table 2-9](#bookmark67)defines the states of the Relaxed Ordering and ID-Based Ordering attribute fields. These attributes are

discussed in [Section 2.4](#bookmark68). Note that Relaxed Ordering and ID-Based Ordering attributes are not adjacent in location (see [Figure 2-5)](#bookmark19).

Table 2-9 Ordering Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute Bit [2] | Attribute Bit [1] | Ordering Type | Ordering Model |
| 0 | 0 | Default Ordering | PCI Strongly Ordered Model |
| 0 | 1 | Relaxed Ordering | PCI-X Relaxed Ordering Model |
| 1 | 0 | ID-Based Ordering | Independent ordering based on Requester/Completer ID |
| 1 | 1 | Relaxed Ordering plus ID-Based Ordering | Logical “OR” of Relaxed Ordering and IDO |

Attribute bit [1] is not applicable and must be Clear for Configuration Requests, I/O Requests, Memory Requests that are Message Signaled Interrupts, and Message Requests (except where specifically permitted).

Attribute bit [2], IDO, is Reserved for Configuration Requests and I/O Requests. IDO is not Reserved for all Memory Requests, including Message Signaled Interrupts (MSI/MSI-X). IDO is not Reserved for Message Requests unless

specifically prohibited. A Requester is permitted to Set IDO only if the IDO Request Enable bit in the Device Control 2 register is Set.

The value of the IDO bit must not be considered by Receivers when determining if a TLP is a Malformed Packet.

A Completer is permitted to Set IDO only if the IDO Completion Enable bit in the Device Control 2 register is Set. It is not required to copy the value of IDO from the Request into the Completion(s) for that Request. If the Completer has IDO

enabled, it is recommended that the Completer set IDO for all Completions, unless there is a specific reason not to (see Appendix E).

A Root Complex that supports forwarding TLPs peer-to-peer between Root Ports is not required to preserve the IDO bit from the Ingress to Egress Port.

[**2.2.6.5**](2.2.6.5) **No Snoop Attribute**

[Table 2-10](#bookmark69)defines the states of the[No Snoop](#bookmark69)attribute field. Note that the [No Snoop](#bookmark69)attribute does not alter Transaction ordering.

Table 2-10 Cache Coherency Management Attribute

|  |  |  |
| --- | --- | --- |
| [No Snoop](#bookmark69)Attribute (b) | 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 Clear for Configuration Requests, I/O Requests, Memory Requests that are Message Signaled Interrupts, and Message Requests (except where specifically permitted).

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

[**2.2.6.6**](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 2.5](#bookmark70) covers the details of the VC mechanism.

[Table 2-11](#bookmark71)defines the TC encodings.

Table 2-11 Definition of TC Field Encodings

|  |  |
| --- | --- |
| TC Field Value (b) | Definition |
| 000 | TC0: Best Effort service class (General Purpose I/O)  (Default TC - must be supported by every PCI Express device) |
| 001 to 111 | TC1 toTC7: Differentiated service classes  (Differentiation based on Weighted-Round-Robin (WRR) 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 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[9:0], forming the Transaction ID.

◦ Last DW BE[3:0] and First DW BE[3:0]. For Memory Read Requests and AtomicOp Requests with the TH bit Set, the byte location for the Last DW BE[3:0] and First DW BE [3:0] fields in the header are

repurposed to carry ST[7:0] field. For Memory Read Requests with the TH bit Clear, see [Section 2.2.5](#bookmark52) for First/Last DW Byte Enable Rules. For AtomicOp Requests with TH bit Set, the values for the DW BE fields are implied to be Reserved. For AtomicOp Requests with TH bit Clear, the DW BE fields are

Reserved.

For Memory Requests, the following rules apply:

• Memory Requests route by address, using either 64-bit or 32-bit Addressing (see [Figure 2-17](#bookmark72)and [Figure 2-18)](#bookmark73).

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

• For Memory Read Requests, Length must not exceed the value specified by Max\_Read\_Request\_Size (see Section 7.5.3.4).

• For AtomicOp Requests, architected operand sizes and their associated Length field values are specified in [Table 2-12](#bookmark74). If a Completer supports AtomicOps, the following rules apply. The Completer must check the

Length field value. If the value does not match an architected value, the Completer must handle the TLP as a Malformed TLP. Otherwise, if the value does not match an operand size that the Completer supports, the

Completer must handle the TLP as an Unsupported Request (UR). This is a reported error associated with the Receiving Port (seeSection 6.2).

Table 2-12 Length Field Values for AtomicOp Requests

|  |  |  |  |
| --- | --- | --- | --- |
| AtomicOp Request | Length Field Value for Architected Operand Sizes | | |
| 32 Bits | 64 Bits | 128 Bits |
| FetchAdd, Swap | 1 DW | 2 DW | N/A |
| CAS | 2 DW | 4 DW | 8 DW |

• A FetchAdd Request contains one operand, the “add” value.

• A Swap Request contains one operand, the “swap” value.

• A CAS Request contains two operands. The first in the data area is the “compare” value, and the second is the “swap” value.

• For AtomicOp Requests, the Address must be naturally aligned with the operand size. The Completer must check for violations of this rule. If a TLP violates this rule, the TLP is a Malformed TLP. This is a reported error associated with the Receiving Port (seeSection 6.2).

• Requests must not specify an Address/Length combination that 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 (seeSection 6.2).

。 For AtomicOp Requests, the mandatory Completer check for natural alignment of the Address (see above) already guarantees that the access will not cross a 4-KB boundary, so a separate 4-KB

boundary check is not necessary.

。 If a 4-KB boundary check is performed for AtomicOp CAS Requests, this check must comprehend that the TLP Length value is based on the size of two operands, whereas the access to Memory Space is

based on the size of one operand.

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

Byte 0 → Byte 4 → Byte 8 → Byte 12 →

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| +0  7  6  5  4  3  2  1  0 | | +1  7  6  5  4  3  2  1  0 | | | | | | | +2  7  6  5  4  3  2  1  0 | | | | | | | +3  7  6  5  4  3  2  1  0 | | |
| Fmt  0  x  1 | Type | T9 | TC | T8 | | Attr | LN | TH | TD | EP | Attr | | AT | | Length | | | |
| Requester ID | | | | | | | | |  | | | Tag | |  | | Last DW BE | First DW BE | |
|  | | | | | Address[63:32] | | | | | | | | | | | | | |
| Address[31:2] | | | | | | | | | | | | | | | | | | PH |

Figure 2-17 Request Header Format for 64-bit Addressing of Memory

Byte 0 → Byte 4 → Byte 8 →

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| +0  7  6  5  4  3  2  1  0 | | +1  7  6  5  4  3  2  1 0 | | | | | | +2  7  6  5  4  3  2  1  0 | | | | | | | +3  7  6  5  4  3  2  1  0 | | |
| Fmt  0  x  0 | Type | T9 | TC | T8 | Attr | LN | TH | TD | EP | Attr | | AT | | Length | | | |
| Requester ID | | | | | | | |  | | | Tag | |  | | Last DW BE | First DW BE | |
| Address[31:2] | | | | | | | | | | | | | | | | | PH |

Figure 2-18 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 2-19)](#bookmark75)

• I/O Requests have the following restrictions: 。 TC[2:0] must be 000b

。 LN is not applicable to I/O Requests and the bit is Reserved 。 TH is not applicable to I/O Request and the bit is Reserved 。 Attr[2] is Reserved

。 Attr[1:0] must be 00b

。 AT[1:0] must be 00b. Receivers are not required or encouraged to check this.

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

。 Length[9:0] must be 00 0000 0001b 。 Last DW BE[3:0] must be 0000b

Receivers may optionally check for violations of these rules (but must not check Reserved bits). These checks are independently optional (see Section 6.2.3.4). If a Receiver implementing these checks determines that a TLP violates these rules, the TLP is a Malformed TLP.

。 If checked, this is a reported error associated with the Receiving Port (seeSection 6.2).

Byte 0 → Byte 4 → Byte 8 →

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| +0  7  6  5  4  3  2  1  0 | | +1  7  6  5  4  3  2  1 0 | | | | | | +2  7  6  5  4  3  2  1  0 | | | | | | | +3  7  6  5  4  3  2  1  0 | | |
| Fmt  0  x  0 | Type | T9 | TC  0  0  0 | T8 | Attr  R | LN | TH | TD | EP | Attr  0  0 | | AT  0  0 | | Length  0  0 0  0  0  0 0  0  0  1 | | | |
| Requester ID | | | | | | | |  | | | Tag | |  | | Last DW BE  0  0  0  0 | First DW BE | |
| Address[31:2] | | | | | | | | | | | | | | | | | R |

Figure 2-19 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 2-20)](#bookmark76).

。 Register Number[5:0]

。 Extended Register Number[3:0]

• Configuration Requests have the following restrictions: 。 TC[2:0] must be 000b

。 LN is not applicable to Configuration Requests and the bit is Reserved 。 TH is not applicable to Configuration Requests and the bit is Reserved 。 Attr[2] is Reserved

。 Attr[1:0] must be 00b

。 AT[1:0] must be 00b. Receivers are not required or encouraged to check this. 。 Length[9:0] must be 00 0000 0001b

。 Last DW BE[3:0] must be 0000b

Receivers may optionally check for violations of these rules (but must not check reserved bits). These checks are independently optional (see Section 6.2.3.4). If a Receiver implementing these checks determines that a TLP violates these rules, the TLP is a Malformed TLP.

。 If checked, this is a reported error associated with the Receiving Port (seeSection 6.2).

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

Byte 0 → Byte 4 → Byte 8 →

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| +0  7  6  5  4  3  2  1  0 | | +1  7  6  5  4  3  2  1  0 | | | | | | +2  7  6  5  4  3  2  1  0 | | | | | | | +3  7  6  5  4  3  2  1  0 | | |
| Fmt  0  x  0 | Type | T9 | TC  0  0  0 | T8 | Attr R | LN | TH | TD | EP | Attr  0  0 | | AT  0  0 | | Length  0  0 0  0  0  0 0  0  0  1 | | | |
| Requester ID | | | | | | | |  | | | Tag | |  | | Last DW BE  0  0  0  0 | First DW BE | |
| Bus Number | | Device Num | | | Fcn Num | | | Reserved | | | | xt Reg Num | | | Register Number | | R |

Figure 2-20 Request Header Format for Configuration Transactions

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

[**2.2.7.1**](2.2.7.1) **TPH Rules**

• Two formats are specified for TPH. The Baseline TPH format (see[Figure 2-22](#bookmark77)and [Figure 2-23)](#bookmark78) must be used for all Requests that provide TPH. The format with the optional [TPH TLP Prefix](#bookmark79)extends the TPH fields (see [Figure 2-21](#bookmark80)) to provide additional bits for the Steering Tag (ST) field.

Byte 0 →

|  |  |  |  |
| --- | --- | --- | --- |
| +0  7  6  5  4  3  2  1  0 | +1  7  6  5  4  3  2  1  0 | +2  7  6  5  4  3  2  1  0 | +3  7  6  5  4  3  2  1  0 |
| {see Section 2.2.1} | ST [15:8] | Reserved | |

Figure 2-21 **TPH TLP Prefix**

• The optional TPH TLP Prefix is used to extend the TPH fields.

。 The presence of a TPH TLP Prefix is determined by decoding byte 0.

Table 2-13 TPH TLP Prefix Bit Mapping

|  |  |
| --- | --- |
| Fields | TPH TLP Prefix |
| ST(15:8) | Bits 7:0 of byte 1 |
| Reserved | Bits 7:0 of byte 2 |
| Reserved | Bits 7:0 of byte 3 |

• For Requests that target Memory Space, a value of 1b in the TH bit indicates the presence of TPH in the TLP header and optional [TPH TLP Prefix](#bookmark81) (if present).

。 The TH bit must be Set for Requests that provide TPH.

。 The TH bit must be Set for Requests with a [TPH TLP Prefix.](#bookmark82)

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

。 When the TH bit is Clear, the PH field is Reserved.

。 The TH bit and the PH field are not applicable and are Reserved for all other Requests.

• The Processing Hints (PH) fields mapping is shown in [Figure 2-22 ,](#bookmark83)[Figure 2-23 and](#bookmark84)[Table 2-14 .](#bookmark85)

Byte 0 → Byte 4 → Byte 8 → Byte 12 →

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| +0  7  6  5  4  3  2  1  0 | | +1  7  6  5  4  3  2  1 0 | | | | | | | +2  7  6  5  4  3  2  1  0 | | | | | +3  7  6  5  4  3  2  1  0 | |
| Fmt  0  x  1 | Type | T9 | | TC | T8 | Attr | LN | TH | TD | EP | Attr | AT | Length | | |
|  | | | {fields in bytes 4 through 7 depend on type of Request} | | | | | | | | | | | | |
| Address[63:32] | | | | | | | | | | | | | | | |
| Address[31:2] | | | | | | | | | | | | | | | PH |

Figure 2-22 Location of PH[1:0] in a 4 DW Request Header

Byte 0 → Byte 4 → Byte 8 →

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| +0  7  6  5  4  3  2  1  0 | | +1  7  6  5  4  3  2  1  0 | | | | | | | +2  7  6  5  4  3  2  1  0 | | | | | +3  7  6  5  4  3  2  1  0 | |
| Fmt  0  x  0 | Type | T9 | | TC | T8 | Attr | LN | TH | TD | EP | Attr | AT | Length | | |
|  | | | {fields in bytes 4 through 7 depend on type of Request} | | | | | | | | | | | | |
| Address[31:2] | | | | | | | | | | | | | | | PH |

Figure 2-23 Location of PH[1:0] in a 3 DW Request Header

Table 2-14 Location of PH[1:0] in TLP Header

|  |  |  |
| --- | --- | --- |
| PH | 32-bit Addressing | 64-bit Addressing |
| 1:0 | Bits 1:0 of Byte 11 | Bits 1:0 of Byte 15 |

• The PH[1:0] field provides information about the data access patterns and is defined as described in [Table 2-15](#bookmark86)

.

Table 2-15 Processing Hint Encoding

|  |  |  |
| --- | --- | --- |
| PH[1:0] (b) | Processing Hint | Description |
| 00 | Bi-directional data structure | Indicates frequent read and/or write access to data by Host and device |
| 01 | Requester | Indicates frequent read and/or write access to data by device |
| 10 | Target | Indicates frequent read and/or write access to data by Host |

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

|  |  |  |
| --- | --- | --- |
| PH[1:0] (b) | Processing Hint | Description |
| 11 | Target with Priority | Indicates frequent read and/or write access by Host and indicates high temporal locality for accessed data |

The Steering Tag (ST) fields are mapped to the TLP header as shown in [Figure 2-24 ,](#bookmark87)[Figure 2-25 and](#bookmark88)[Table 2-16 .](#bookmark89)

Byte 0 → Byte 4 →

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| +0  7  6  5  4  3  2  1  0 | | | +1  7  6  5  4  3  2  1 0 | | | | | | +2  7  6  5  4  3  2  1  0 | | | | | +3  7  6  5  4  3  2  1  0 | |
| Fmt | Type | T9 | | TC | T8 | Attr | LN | TH | TD | EP | Attr | AT | Length | | |
| Requester ID | | | | | | | | | ST[7:0] | | | | | Last DW BE | First DW BE |

Figure 2-24 Location of ST[7:0] in the Memory Write Request Header

Byte 0 → Byte 4 →

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| +0  7  6  5  4  3  2  1  0 | | +1  7  6  5  4  3  2  1  0 | | | | | | +2  7  6  5  4  3  2  1  0 | | | | | | | +3  7  6  5  4  3  2  1  0 |
| Fmt | Type | T9 | TC | T8 | Attr | LN | TH | TD | EP | Attr | | AT | | Length | |
| Requester ID | | | | | | | |  | | | Tag | |  | | ST[7:0] |

Figure 2-25 Location of ST[7:0] in Memory Read and AtomicOp Request Headers

Table 2-16 Location of ST[7:0] in TLP Headers

|  |  |  |
| --- | --- | --- |
| ST Bits | Memory Write Request | Memory Read Request or AtomicOp Request |
| 7:0 | Bits 7:0 of Byte 6 | Bits 7:0 of Byte 7 |

• ST[7:0] field carries the Steering Tag value

。 A value of all zeroes indicates no Steering Tag preference 。 A total of 255 unique Steering Tag values are provided

• A Function that does not support the TPH Completer or Routing capability and receives a transaction with the TH bit Set is required to ignore the TH bit and handle the Request in the same way as Requests of the same

transaction type without the TH bit Set.

**2.2.8 Message Request Rules**

This document defines the following groups of Messages:

• INTx Interrupt Signaling

• Power Management

• Error Signaling

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

• Locked Transaction Support

• Slot Power Limit Support

• Vendor-Defined Messages

• Latency Tolerance Reporting (LTR) Messages

• Optimized Buffer Flush/Fill (OBFF) Messages

• Device Readiness Status (DRS) Messages

• Function Readiness Status (FRS) Messages

• Precision Time Measurement (PTM) Messages

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 2-37](#bookmark90)): 。 Requester ID[15:0] and Tag[9:0], forming the Transaction ID.

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

• All Message Requests use theMsg or MsgD Type field encoding.

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

• The Attr[2] field is not Reserved unless specifically indicated as Reserved.

• Except as noted, the Attr[1:0] field is Reserved.

• LN is not applicable to Message Requests and the bit is Reserved.

• Except as noted,TH is not applicable to Message Requests and the bit is Reserved.

• AT[1:0] must be 00b. Receivers are not required or encouraged to check this.

• Except as noted, bytes 8 through 15 are Reserved.

• Message Requests are posted and do not require Completion.

• Message Requests follow the same ordering rules as Memory Write Requests.

Many types of Messages, including Vendor-Defined Messages, are potentially usable in non-D0 states, and it is strongly recommended that the handling of Messages by Ports be the same when the Port's Bridge Function is in D1,D2, and

D3Hotas it is in D0. It is strongly recommended that Type 0 Functions support the generation and reception of Messages in non-D0 states.

Byte 0 → Byte 4 → Byte 8 → Byte 12 →

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| +0  7  6  5  4  3  2  1  0 | | +1  7  6  5  4  3  2  1 0 | | | | | | +2  7  6  5  4  3  2  1  0 | | | | | | | +3  7  6  5  4  3  2  1  0 |
| Fmt  0  x  1 | Type  1  0 r2r 1r0 | T9 | TC | T8 | Attr | R | TH | TD | EP | Attr | | AT  0  0 | | Length | |
| Requester ID | | | | | | | |  | | | Tag | |  | | Message Code |
| {Except as noted, bytes 8 through 11 are reserved.} | | | | | | | | | | | | | | | |
| {Except as noted, bytes 12 through 15 are reserved.} | | | | | | | | | | | | | | | |

Figure 2-26 Message Request Header

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

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, but rather the destination is implied by the routing type. The following rules cover Message routing mechanisms:

• 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 2-17](#bookmark91)

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

Table 2-17 Message Routing

|  |  |  |
| --- | --- | --- |
| r[2:0] (b) | Description | Bytes 8 to 1515 |
| 000 | Routed to Root Complex | Reserved |
| 001 | Routed by Address16 | Address |
| 010 | Routed by ID | See [Section 2.2.4.2](#bookmark40) |
| 011 | Broadcast from Root Complex | Reserved |
| 100 | Local - Terminate at Receiver | Reserved |
| 101 | Gathered and routed to Root Complex17 | Reserved |
| 110 to 111 | Reserved - Terminate at Receiver | Reserved |

[**2.2.8.1**](2.2.8.1) **INTx Interrupt Signaling - Rules**

A Message Signaled Interrupt (MSI or MSI-X) is the preferred interrupt signaling mechanism in PCI Express (seeSection 6.1 ). However, in some systems, there may be Functions that cannot support the MSI or MSI-X mechanisms. The INTx

virtual wire interrupt signaling mechanism is used to support Legacy Endpoints and PCI Express/PCI(-X) Bridges in cases where the MSI or MSI-X mechanisms cannot be used. Switches must support this mechanism. The following rules apply to the INTx Interrupt Signaling mechanism:

• The INTx mechanism uses eight distinct Messages (see [Table 2-18)](#bookmark92).

• Assert\_INTx/Deassert\_INTx Messages do not include a data payload (TLP Type is Msg).

• The Length field is Reserved.

• With Assert\_INTx/Deassert\_INTx Messages, the Function Number field in the Requester ID must be 0. Note that the Function Number field is a different size for non-ARI andARI Requester IDs.

• 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 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 (seeSection 6.2).

• 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 (seeSection 6.2).



15. Except as noted, e.g., Vendor\_Defined Messages.

16. Note that no Messages defined in this document use Address routing.

17. This routing type is used only for[PME\_TO\_Ack,](#bookmark93) and is described in Section 5.3.3.2.1 .

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

Table 2-18 INTx Mechanism Messages

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Code[7:0]  (b) | Routing r[2:0] (b) | Support18 | | | | Description/Comments |
| RC | Ep | Sw | Br |
| Assert\_INTA | 0010  0000 | 100 | All: | | | | Assert INTA virtual wire  Note: These Messages are used for Conventional PCI-compatible INTx emulation. |
| r |  | tr |  |
| As Required: | | | |
|  | t |  | t |
| Assert\_INTB | 0010  0001 | 100 | All: | | | | Assert INTB virtual wire |
| r |  | tr |  |
| As Required: | | | |
|  | t |  | t |
| Assert\_INTC | 0010  0010 | 100 | All: | | | | Assert INTC virtual wire |
| r |  | tr |  |
| As Required: | | | |
|  | t |  | t |
| Assert\_INTD | 0010  0011 | 100 | All: | | | | Assert INTD virtual wire |
| r |  | tr |  |
| As Required: | | | |
|  | t |  | t |
| Deassert\_INTA | 0010  0100 | 100 | All: | | | | Deassert INTA virtual wire |
| r |  | tr |  |
| As Required: | | | |
|  | t |  | t |
| Deassert\_INTB | 0010  0101 | 100 | All: | | | | Deassert INTB virtual wire |
| r |  | tr |  |
| As Required: | | | |
|  | t |  | t |
| Deassert\_INTC | 0010  0110 | 100 | All: | | | | Deassert INTC virtual wire |
| r |  | tr |  |
| As Required: | | | |
|  | t |  | t |



18. 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

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Code[7:0]  (b) | Routing r[2:0] (b) | Support | | | | Description/Comments |
| RC | Ep | Sw | Br |
| Deassert\_INTD | 0010  0111 | 100 | All: | | | | Deassert INTD virtual wire |
| r |  | tr |  |
| 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 (seeSection

7.5.1.1.3 ) is Set.

。 Any INTx virtual wires that are active when the Interrupt Disable bit is set must bedeasserted 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 2-19 .](#bookmark94)

• 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 Switch Downstream Port goes to DL\_Down status, the INTx virtual wires associated with that Port must be deasserted, and the Switch Upstream Port virtual wire state updated accordingly.

。 If this results in deassertion 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 bedeasserted, and any associated system interrupt resource request(s) must be discarded.

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

Table 2-19 Bridge Mapping for INTx Virtual Wires

|  |  |  |
| --- | --- | --- |
| Requester ID[7:3] from the Assert\_INTx/Deassert\_INTx Message  received on Secondary Side of Bridge (Interrupt Source19 )  If ARI Forwarding is enabled, the value 0 must be used instead of  Requester ID[7:3]. | 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 |



**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 2-19 is](#bookmark94) applied hierarchically at every Switch. In addition, PCI Express/PCI and PCI/PCI Bridges perform a similar

mapping function.



19. 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.

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

**IMPLEMENTATION NOTE**

Virtual Wire Mapping for INTx Interrupts From ARI Devices

The implied Device Number for an ARI Device is 0. When ARI-aware software (including BIOS and operating

system) enables ARI Forwarding in the Downstream Port immediately above an ARI Device in order to access its Extended Functions, software must comprehend that the Downstream Port will use Device Number 0 for the

virtual wire mappings of INTx interrupts coming from all Functions of the ARI Device. If non-ARI-aware software attempts to determine the virtual wire mappings for Extended Functions, it can come up with incorrect mappings by examining the traditional Device Number field and finding it to be non-0.



[**2.2.8.2**](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 2-20](#bookmark95)defines the Power Management Messages.

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

• The Length field is Reserved.

• With [PM\_Active\_State\_Nak](#bookmark96) Messages, the Function Number field in the Requester ID must contain the Function Number of the Downstream Port that sent the Message, or else 000b for compatibility with earlier revisions of this specification.

• With [PME\_TO\_Ack](#bookmark97) Messages, the Function Number field in the Requester ID must be Reserved, or else for compatibility with earlier revisions of this specification must contain the Function Number of one of the Functions associated with the Upstream Port. Note that the Function Number field is a different size for non-ARI andARI Requester IDs.

• 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 (seeSection 6.2).

Table 2-20 Power Management Messages

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Code[7:0]  (b) | Routing r[2:0] (b) | Support | | | | Description/Comments |
| RC | Ep | Sw | Br |
| **PM\_Active\_State\_Nak** | 0001  0100 | 100 | t | r | tr | r | Terminate at Receiver |
| **PM PME** | 0001  1000 | 000 | All: | | | | Sent Upstream by PME-requesting component. Propagates Upstream. |
| r |  | tr | t |
| If PME supported: | | | |
|  | t |  |  |
| **PME\_Turn\_Off** | 0001  1001 | 011 | t | r |  | r | Broadcast Downstream |

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Code[7:0]  (b) | Routing r[2:0] (b) | Support | | | | Description/Comments |
| RC | Ep | Sw | Br |
| **PME\_TO\_Ack** | 0001  1011 | 101 | r | t |  | t | Sent Upstream by Upstream Port. See Section  5.3.3.2.1 . |
| (Note: Switch handling is special) | | | |

[**2.2.8.3**](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. These Messages are initiated by the agent that detected the error.

• [Table 2-21](#bookmark98)defines the Error Signaling Messages.

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

• The Length field is Reserved.

• With Error Signaling Messages, the Function Number field in the Requester ID must indicate which Function is signaling the error. Note that the Function Number field is a different size for non-ARI andARI Requester IDs.

• 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 handle the TLP as a

Malformed TLP.

。 This is a reported error associated with the Receiving Port (seeSection 6.2).

Table 2-21 Error Signaling Messages

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Code[7:0]  (b) | Routing r[2:0] (b) | Support | | | | Description/Comments |
| RC | Ep | Sw | Br |
| **ERR\_COR** | 0011  0000 | 000 | r | t | tr | t | This Message is issued when the Function or Device detects a correctable error on the PCI Express interface. |
| **ERR\_NONFATAL** | 0011  0001 | 000 | r | t | tr | t | This Message is issued when the Function or Device detects a Non-Fatal, uncorrectable error on the PCI Express interface. |
| **ERR FATAL** | 0011  0011 | 000 | r | t | tr | t | This Message is issued when the Function 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 toSection 6.2for details on uses for these Messages.

• [ERR\_COR](#bookmark99) Messages have an [ERR\_COR Subclass](#bookmark102)(ECS) field in the Message header that enables different

subclasses to be distinguished from each other. See [Figure 2-27 .](#bookmark103)[ERR\_NONFATAL](#bookmark100)and [ERR\_FATAL](#bookmark101) Messages do not have the ECS field.

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

Byte 0 → Byte 4 → Byte 8 → Byte 12 →

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| +0  7  6  5  4  3  2  1  0 | | | +1  7  6  5  4  3  2  1 0 | | | | | | +2  7  6  5  4  3  2  1  0 | | | | | | | +3  7  6  5  4  3  2  1  0 |
| Fmt  0  0  1 | | Type  1  0  0  0  0 | T9  R | TC  0  0  0 | T8  R | Attr  R | LN  R | TH  R | TD | EP | Attr  R  R | | AT  0  0 | | Length | |
| Requester ID | | | | | | | | |  | | | Tag | |  | | Message Code  0  0  1  1  0  0  0  0 |
| ECS | Reserved | | | | | | | | | | | | | | | |
| Reserved | | | | | | | | | | | | | | | | |

Figure 2-27 **ERR\_COR Message**

• The [ERR\_COR Subclass](#bookmark104)(ECS) field is encoded as shown in [Table 2-22](#bookmark105), indicating the [ERR\_COR](#bookmark99) Message subclass.

Table 2-22 **ERR\_COR Subclass** (**ECS**) Field Encodings

|  |  |
| --- | --- |
| ECS  Coding | Description |
| 00 | **ECS Legacy** - The value inherently used if a Requester does not support ECS capability. ECS-capable Requesters must not use this value. See see Section 7.5.3.3 . |
| 01 | **ECS SIG\_SFW** - Must be used by an ECS-capable Requester when signaling a DPCor SFI event with an [ERR\_COR](#bookmark99) Message. |
| 10 | **ECS SIG\_OS** - Must be used by an ECS-capable Requester when signaling an AERor RP PIO event with an [ERR\_COR](#bookmark99) Message. |
| 11 | **ECS Extended**- Intended for possible future use. Requesters must not use this value. Receivers must handle the signal internally the same as [ECS SIG\_OS.](#bookmark106) |

[**2.2.8.4**](2.2.8.4) **Locked Transactions Support**

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

• [Table 2-23](#bookmark107)defines the Unlock Messages.

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

• The Length field is Reserved.

• With Unlock Messages, the Function Number field in the Requester ID 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.

。 This is a reported error associated with the Receiving Port (seeSection 6.2).

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

Table 2-23 Unlock Message

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Code[7:0] (b) | Routing r[2:0] (b) | Support | | | | Description/Comments |
| RC | Ep | Sw | Br |
| Unlock | 0000 0000 | 011 | t | r | tr | r | Unlock Completer |

[**2.2.8.5**](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 (with Endpoint, Switch, or PCI Express-PCI Bridge Functions) attached to the same

Link.

• [Table 2-24](#bookmark108)defines the [Set\_Slot\_Power\_Limit Message.](#bookmark109)

• The [Set\_Slot\_Power\_Limit Message](#bookmark110)includes a 1 DW data payload (TLP Type is MsgD).

• The [Set\_Slot\_Power\_Limit Message](#bookmark111) 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 (seeSection 6.2).

Table 2-24 **Set\_Slot\_Power\_Limit Message**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Code[7:0] (b) | Routing r[2:0] (b) | Support | | | | Description/Comments |
| RC | Ep | Sw | Br |
| Set\_Slot\_Power\_Limit | 0101 0000 | 100 | t | r | tr | r | Set Slot Power Limit in Upstream Port |

The [Set\_Slot\_Power\_Limit Message](#bookmark111)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 1:0 of Byte 1 of the data payload map to the Slot Power Limit Scale field and bits 7:0 of

Byte 0 map to the Slot Power Limit Value field. Bits 7:0 of Byte 3, 7:0 of Byte 2, and 7:2 of Byte 1 of the data payload must all be set to zero by the Transmitter and ignored by the Receiver. This Message must be sent automatically by the

Downstream Port (of a Root Complex or a Switch) when one of the following events occurs:

• On a Configuration Write to the Slot Capabilities register (seeSection 7.5.3.9) 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 (seeSection 2.9.2) and the Auto Slot Power Limit Disable bit is Clear in the Slot Control Register. This transmission is optional if the Slot

Capabilities register has not yet been initialized.

The component on the other side of the Link (with Endpoint, Switch, or Bridge Functions) that receives

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 an adapter where power

consumption of the entire adapter is below the lowest power limit specified for the adapter form factor (as defined in the corresponding form factor specification) are permitted to hardwire the value of all 0's 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 Message payload into that register.

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

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

[**2.2.8.6**](2.2.8.6) **Vendor\_Defined Messages**

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. This section defines the rules associated with these

Messages generically.

• The Vendor\_Defined Messages (see [Table 2-25)](#bookmark112) use the header format shown in [Figure 2-28 .](#bookmark113)

。 The Requester ID is implementation specific. It is strongly recommended that the Requester ID field contain the value associated with the Requester.20

。 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.

。 Bytes 12 through 15 are available for vendor definition.

Table 2-25 Vendor\_Defined Messages

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Code[7:0]  (b) | Routing r[2:0]  (b) | Support | | | | Description/Comments |
| RC | Ep | Sw | Br |
| Vendor\_Defined Type 0 | 0111 1110 | 000, 010, 011,  100 | See Note 1. | | | | Triggers detection of UR by Completer if not implemented. |
| Vendor\_Defined Type 1 | 0111 1111 | 000, 010, 011,  100 | See Note 1. | | | | Silently discarded by Completer if not implemented. |

1. Note 1: Transmission by Endpoint/Root Complex/Bridge is implementation specific. Switches must forward received Messages using Routing r[2:0] field values of 000b, 010b, and 011b.



Byte 0 → Byte 4 → Byte 8 → Byte 12 →

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| +0  7  6  5  4  3  2  1  0 | | +1  7  6  5  4  3  2  1  0 | | | | | | +2  7  6  5  4  3  2  1  0 | | | | | | | +3  7  6  5  4  3  2  1  0 |
| Fmt  0  x  1 | Type  1  0 r2r 1r0 | T9 | TC | T8 | Attr | R | TH | TD | EP | Attr | | AT  0  0 | | Length | |
| Requester ID | | | | | | | |  | | | Tag | |  | | Message Code  0  1  1  1  1  1  1  x |
| Bus Number | | Device Num | | | Fcn Num | | |  | | | | | Vendor ID | | |
| {For Vendor Definition} | | | | | | | | | | | | | | | |

Figure 2-28 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 or MsgD if a data payload is included).



20. ACS Source Validation (seeSection 6.12.1.1) checks the Requester ID on all Requests, including Vendor\_Defined Messages. This validation depends on the Requester ID properly identifying the Requester.

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

• For both types of Vendor\_Defined Messages, the Attr[1:0] and Attr[2] fields are 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.

• Completers silently discard Vendor\_Defined Type 1 Messages that they are not designed to receive -this is not an error condition.

• Completers handle the receipt of an unsupported Vendor\_Defined Type 0 Message as an Unsupported Request, and the error is reported according toSection 6.2 .

[PCIe-to-PCI-PCI-X-Bridge-1.0] defines additional requirements for Vendor\_Defined Messages that are designed to be interoperable with PCI-X Device ID Messages. This includes restrictions on the contents of the Tag[7:0] field and the

Length[9:0] field as well as specific use of Bytes 12 through 15 of the message header. Vendor\_Defined Messages

intended for use solely within a PCI Express environment (i.e., not intended to address targets behind a PCI Express to PCI/PCI-X Bridge) are not subject to the additional rules. Refer to [PCIe-to-PCI-PCI-X-Bridge-1.0] for details. Refer to

[Section 2.2.6.2f](#bookmark59)or considerations regarding 10-Bit Tag capability.

**2.2.8.6.1 PCI-SIG-Defined VDMs**

PCI-SIG-Defined VDMs are Vendor-Defined Type 1 Messages that use the PCI-SIG® Vendor ID (0001h). As a Vendor-Defined Type 1 Message, each is silently discarded by a Completer if the Completer does not implement it.

Beyond the rules for other Vendor-Defined Type 1 Messages,the following rules apply to the formation of the PCI-SIG-Defined VDMs:

• PCI-SIG-Defined VDMs use the Header format shown in [Figure 2-29 .](#bookmark115)

• The Requester ID field must contain the value associated with the Requester.

• The Message Code must be 01111111b.

• The Vendor ID must be 0001h, which is assigned to the PCI-SIG.

• The Subtype field distinguishes the specific PCI-SIG-Defined VDMs. See Appendix Ffor a list of PCI-SIG-Defined VDMs.

Byte 0 → Byte 4 → Byte 8 → Byte 12 →

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| +0  7  6  5  4  3  2  1  0 | | +1  7  6  5  4  3  2  1  0 | | | | | | +2  7  6  5  4  3  2  1  0 | | | | | | | +3  7  6  5  4  3  2  1  0 |
| Fmt  0  x  1 | Type  1  0 r2r 1r0 | T9 | TC | T8 | Attr | LN | TH | TD | EP | Attr | | AT  0  0 | | Length | |
| Requester ID | | | | | | | |  | | | Tag | |  | | Message Code  0  1  1  1  1  1  1  1 |
| Destination ID if ID Routed; otherwise Reserved | | | | | | | | Vendor ID  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1 | | | | | | | |
| Subtype | | {For PCI-SIG-Defined VDM Definition} | | | | | | | | | | | | | |

Figure 2-29 Header for PCI-SIG-Defined VDMs

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

**2.2.8.6.2 LN Messages**

LN protocol (see Section 6.21defines LN Messages, which are PCI-SIG-Defined VDMs. The payload of each Message

generally contains the 64-bit Address of a registered cacheline that has been updated or evicted. The single 64-bit

address format is used both with 64- and 32-bit addresses. Since each LN Message is a Vendor-Defined Type 1 Message, a

Completer that receives a properly formed LN Message is required to silently discard it if the Completer doesn't recognize the Message.

An LN Message can be directed to a single Endpoint using ID-based routing, or broadcast to all devices below a given Root Port. Whether a broadcast LN Message is sent to all Root Ports in the RC is implementation specific.

Beyond the rules for other PCI-SIG-Defined VDMs, the following rules apply to the formation of LN Messages:

• [Table 2-27](#bookmark116)and [Figure 2-30](#bookmark117)define the LN Messages.

• Each Message must include a 2-DW data payload.

• The Fmt field must be 011b (4 DW Header, with data).

• The TLP Type must be MsgD.

• The Length field must be 2.

• The TC[2:0] field must be 000b.

• Attr[2], the[ID-Based Ordering](#bookmark62) (IDO) bit, is not Reserved.

• Attr[1], the[Relaxed Ordering](#bookmark62) (RO) bit, is not Reserved.

• Attr[0], the[No Snoop](#bookmark69) bit, is Reserved.

• The LN bit is Reserved (in contrast, the LN bit must be Set for LN Reads, LN Writes, and LN Completions).

• The Tag field is Reserved.

• If the LN Message is the broadcast version, the Destination ID field is Reserved.

• The Subtype field must be 00h.

• If the cache line size in effect for the system is 128 bytes, bit 6 in the Cacheline Address must be Clear. For a Lightweight Notification Requester (LNR) receiving an LN Message, if the LNR CLS bit in the LNR Control

register is Set, configuring the LNR for 128-byte cachelines, the LNR must ignore the value of bit 6 in the Cacheline Address.

• The Notification Reason (NR) field is encoded as shown in [Table 2-26](#bookmark118), indicating the specific reason that the LN Message was sent. These encodings apply to both the directed and broadcast versions of LN Messages.

Table 2-26 Notification Reason (NR) Field Encodings

|  |  |
| --- | --- |
| NR Coding  (b) | Description |
| 00 | LN Message was sent due to a cacheline update. |
| 01 | LN Message was sent due to the eviction of a single cacheline. |
| 10 | LN Message was sent due to the eviction of all cachelines registered to this Function. For this case, the Cacheline Address is Reserved. |
| 11 | Reserved |

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

Table 2-27 LN Messages

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Code[7:0]  (b) | Routing r[2:0]  (b) | Support | | | | Description/Comments |
| RC | Ep | Sw | Br |
| LN Message (directed) | 0111 1111 | 010 | t | r | tr | r | RC directs to a single Endpoint. |
| LN Message (broadcast) | 0111 1111 | 011 | t | r | tr | r | RC broadcasts to all devices under a given Root Port. |

The format of the LN Message is shown in [Figure 2-30](#bookmark119)below.

Byte 0 → Byte 4 → Byte 8 → Byte 12 → Byte 16 → Byte 20 →

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| +0  7  6  5  4  3  2  1  0 | | +1  7  6  5  4  3  2  1 0 | | | | | | | +2  7  6  5  4  3  2  1  0 | | | | | +3  7  6  5  4  3  2  1  0 | | |
| Fmt  0  1  1 | Type  1  0 r2r 1r0 | T9 R | TC  0  0  0 | | T8 R | Attr | LN R | TH | TD | EP | Attr  R | AT  0  0 | Length  0  0  0  0  0  0  0  0  1  0 | | | |
| Requester ID | | | | | | | | | Reserved | | | | | Message Code  0  1  1  1  1  1  1  1 | | |
| Destination ID if ID Routed; Reserved if broadcast | | | | | | | | | Vendor ID  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1 | | | | | | | |
| Subtype  0  0  0  0  0  0  0  0 | | Reserved | | | | | | | | | | | | | | |
|  | | | | Cacheline Address [63:32] | | | | | | | | | | | | |
| Cacheline Address [31:6] | | | | | | | | | | | | | | | Reserved | NR |

Figure 2-30 LN Message

**2.2.8.6.3 Device Readiness Status (DRS) Message**

The Device Readiness Status (DRS) protocol (see Section 6.23.1) uses the PCI-SIG-DefinedVDM mechanism (see [Section](#bookmark114)

[2.2.8.6.1](#bookmark114)). The DRS Message is a PCI-SIG-DefinedVDM (Vendor-Defined Type 1 Message) with no payload.

Beyond the rules for other PCI-SIG-Defined VDMs, the following rules apply to the formation of DRS Messages:

• [Table 2-28](#bookmark120)and [Figure 2-31](#bookmark121)illustrate and define the DRS Message.

• The TLP Type must be Msg.

• The TC[2:0] field must be 000b.

• The Attr[2:0] field is Reserved.

• The Tag field is Reserved.

• The Subtype field must be 08h.

• The Message Routing field must be set to 100b - Local - Terminate at Receiver.

Receivers may optionally check for violations of these rules (but must not check reserved bits). These checks are

independently optional (see Section 6.2.3.4). If a Receiver implementing these checks determines that a TLP violates these rules, the TLP is a Malformed TLP.

• If checked, this is a reported error associated with the Receiving Port (seeSection 6.2).

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

Table 2-28 DRS Message

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Code[7:0] (b) | Routing r[2:0] (b) | Support | | | | Description/Comments |
| RC | Ep | Sw | Br |
| DRS Message | 0111 1111 | 100 | r | t | tr |  | Device Readiness Status |

The format of the DRS Message is shown in [Figure 2-31](#bookmark122)below:

Byte 0 → Byte 4 → Byte 8 → Byte 12 →

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| +0  7  6  5  4  3  2  1  0 | | +1  7  6  5  4  3  2  1 0 | | | | | | +2  7  6  5  4  3  2  1  0 | | | | | +3  7  6  5  4  3  2  1  0 |
| Fmt  0  0  1 | Type  1  0  1  0  0 | T9 R | TC  0  0  0 | T8 R | Attr R | LN R | TH | TD | EP | Attr R  R | AT  0  0 | Length  0  0  0  0  0  0  0  0  0  0 | |
| Requester ID | | | | | | | | Reserved | | | | | Message Code  0  1  1  1  1  1  1  1 |
| Reserved | | | | | | | | Vendor ID  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1 | | | | | |
| Subtype  0  0  0  0  1  0  0  0 | | Reserved | | | | | | | | | | | |

Figure 2-31 DRS Message

**2.2.8.6.4 Function Readiness Status Message (FRS Message)**

The Function Readiness Status (FRS) protocol (see Section 6.23.2) uses the PCI-SIG-DefinedVDM mechanism (see [Section 2.2.8.6.1](#bookmark114)). The FRS message is a PCI-SIG-DefinedVDM (Vendor-Defined Type 1 Message) with no payload. Beyond the rules for other PCI-SIG-Defined VDMs, the following rules apply to the formation of[FRS Messages:](#bookmark123)

[Table 2-29](#bookmark124)and [Figure 2-32](#bookmark125)illustrate and define the FRS Message.

• The TLP Type must be Msg.

• The TC[2:0] field must be 000b.

• The Attr[2:0] field is Reserved.

• The Tag field is Reserved.

• The Subtype field must be 09h.

• The FRS Reason[3:0] field indicates why the FRS Message was generated:

**0001b: DRS Message Received**

The Downstream Port indicated by the Message Requester ID received a DRS Message and has the DRS Signaling Control field in the Link Control Register set to DRS to FRS Signaling Enabled

**0010b:D3HottoD0Transition Completed**

A D3HottoD0transition has completed, and the Function indicated by the Message Requester ID is now Configuration-Ready and has returned to theD0uninitializedor D0activestate depending on the setting of the No\_Soft\_Reset bit (seeSection 7.5.2.2)

**0011b: FLR Completed**

An FLR has completed, and the Function indicated by the Message Requester ID is now Configuration-Ready

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

**1000b: VF Enabled**

The Message Requester ID indicates a Physical Function (PF) - AllVirtual Functions (VFs) associated with that PF are now Configuration-Ready

**1001b: VF Disabled**

The Message Requester ID indicates a PF- All VFs associated with that PF have been disabled and the Single Root I/O Virtualization (SR-IOV) data structures in that PF may now be accessed.

**Others:**

All other values Reserved

• The Message Routing field must be Cleared to 000b - Routed to Root Complex

Receivers may optionally check for violations of these rules (but must not check reserved bits). These checks are

independently optional (see Section 6.2.3.4). If a Receiver implementing these checks determines that a TLP violates these rules, the TLP is a Malformed TLP.

• If checked, this is a reported error associated with the Receiving Port (seeSection 6.2).

Table 2-29 FRS Message

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Code[7:0] (b) | Routing r[2:0] (b) | Support | | | | Description/Comments |
| RC | Ep | Sw | Br |
| FRS Message | 0111 1111 | 000 | r | t | tr |  | Function Readiness Status |

The format of the FRS Message is shown in [Figure 2-32](#bookmark126)below:

Byte 0 → Byte 4 → Byte 8 → Byte 12 →

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| +0  7  6  5  4  3  2  1  0 | | | +1  7  6  5  4  3  2  1  0 | | | | | | | +2  7  6  5  4  3  2  1  0 | | | | | +3  7  6  5  4  3  2  1  0 | |
| Fmt  0  0  1 | Type  1  0  0  0  0 | T9 R | | TC  0  0  0 | T8 R | Attr R | LN R | TH | TD | | EP | Attr R  R | AT  0  0 | Length  0  0  0  0  0  0  0  0  0  0 | | |
| Requester ID | | | | | | | | | Reserved | | | | | | Message Code  0  1  1  1  1  1  1  1 | |
| Reserved | | | | | | | | | Vendor ID  0  0  0  0  0  0  0  0  0  0  0  0 0  0  0  1 | | | | | | | |
| Subtype  0  0  0  0  1  0  0  1 | | Reserved | | | | | | | | | | | | | | FRS Reason |

Figure 2-32 FRS Message

**2.2.8.6.5 Hierarchy ID Message**

Hierarchy ID uses the PCI-SIG-DefinedVDM mechanism (see[Section 2.2.8.6.1](#bookmark114)). The Hierarchy ID Message is a PCI-SIG-DefinedVDM (Vendor-Defined Type 1 Message) with payload (MsgD).

Beyond the rules for other PCI-SIG-Defined VDMs, the following rules apply to the formation of Hierarchy ID Messages:

• [Table 2-30](#bookmark127)and [Figure 2-33](#bookmark128)illustrate and define the Hierarchy ID Message.

• The TLP Type must be MsgD.

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

• Each Message must include a 4-DWORD data payload.

• The Length field must be 4.

• The TC[2:0] field must be 000b.

• The Attr[2:0] field is Reserved.

• The Tag field is Reserved.

• The Subtype field is 01h.

• The Message Routing field must be 011b - Broadcast from Root Complex.

Receivers may optionally check for violations of these rules (but must not check reserved bits). These checks are

independently optional (see Section 6.2.3.4). If a Receiver implementing these checks determines that a TLP violates these rules, the TLP is a Malformed TLP.

• If checked, this is a reported error associated with the Receiving Port (seeSection 6.2). The payload of each Hierarchy ID Message contains the lower 128-bits of the System GUID. For details of the Hierarchy ID, GUID Authority ID, and System GUID fields see Section 6.26 .

Table 2-30 Hierarchy ID Message

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Code[7:0] (b) | Routing r[2:0] (b) | Support | | | | Description/Comments |
| RC | Ep | Sw | Br |
| Hierarchy ID Message | 0111 1111 | 011 | t | r | tr |  | Hierarchy ID |

The format of the Hierarchy ID Message is shown in [Figure 2-33](#bookmark129)below:

Byte 0 → Byte 4 → Byte 8 → Byte 12 → Byte 16 → Byte 20 → Byte 24 → Byte 28 →

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| +0  7  6  5  4  3  2  1  0 | | +1  7  6  5  4  3  2  1  0 | | | | | | | +2  7  6  5  4  3  2  1  0 | | | | | +3  7  6  5  4  3  2  1  0 |
| Fmt  0  1  1 | Type  1  0  0  1  1 | T9 R | TC  0  0  0 | T8 R | Attr R | LN R | TH | TD | | EP | Attr R  R | AT  0  0 | Length  0  0  0  0  0  0  0  1  0  0 | |
| Requester ID | | | | | | | | Reserved | | | | | | Message Code  0  1  1  1  1  1  1  1 |
| Hierarchy ID | | | | | | | | Vendor ID  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1 | | | | | | |
| Subtype  0  0  0  0  0  0  0  1 | | GUID Authority ID | | | | | | System GUID[135:128] | | | | | | |
| System GUID[127:96] | | | | | | | | | | | | | | |
| System GUID[95:32] | | | | | | | | | | | | | | |
| System GUID[63:32] | | | | | | | | | | | | | | |
| System GUID[31:0] | | | | | | | | | | | | | | |

Figure 2-33 Hierarchy ID Message

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

[**2.2.8.7**](2.2.8.7) **Ignored Messages**

The messages listed in were previously used for a mechanism (Hot-Plug Signaling) that is no longer supported.

Transmitters are strongly encouraged not to transmit these messages, but if message transmission is implemented, it must conform to the requirements of the 1.0a version of this specification.

Receivers are strongly encouraged to ignore receipt of these messages, but are allowed to process these messages in conformance with the requirements of 1.0a version of this specification.

Ignored messages listed in [Table 2-31](#bookmark131)are handled by the Receiver as follows:

• The Physical and Data Link Layers must handle these messages identical to handling any other TLP.

• The Transaction Layer must account for flow control credit but take no other action in response to these messages.

Table 2-31 Ignored Messages

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Code[7:0] (b) | Routing r[2:0] (b) | Support | | | | Description/Comments |
| RC | Ep | Sw | Br |
| Ignored Message | 0100 0001 | 100 |  | | | |  |
| Ignored Message | 0100 0011 | 100 |  | | | |  |
| Ignored Message | 0100 0000 | 100 |  | | | |  |
| Ignored Message | 0100 0101 | 100 |  | | | |  |
| Ignored Message | 0100 0111 | 100 |  | | | |  |
| Ignored Message | 0100 0100 | 100 |  | | | |  |
| Ignored Message | 0100 1000 | 100 |  | | | |  |

[**2.2.8.8**](2.2.8.8) **Latency Tolerance Reporting (LTR) Message**

The LTR Message is optionally used to report device behaviors regarding its tolerance of Read/Write service latencies. Refer toSection 6.18for details on LTR. The following rules apply to the formation of the LTR Message:

• [Table 2-32](#bookmark132)defines the LTR Message.

• The LTR Message does not include a data payload (the TLP Type is Msg).

• The Length field is Reserved.

• The LTR Message must use the default Traffic Class designator (TC0). Receivers that implement LTR support 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 (seeSection 6.2).

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

Table 2-32 LTR Message

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Code[7:0] (b) | Routing r[2:0] (b) | Support1 | | | | Description/Comments |
| RC | Ep | Sw | Br |
| LTR | 0001 0000 | 100 | r | t | tr |  | Latency Tolerance Reporting |

Notes:

1. Support for LTR is optional. Functions that support LTR must implement the reporting and enable mechanisms described in Chapter 7 .



Byte 0 → Byte 4 → Byte 8 → Byte 12 →

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| +0  7  6  5  4  3  2  1  0 | | +1  7  6  5  4  3  2  1 0 | | | | | | +2  7  6  5  4  3  2  1  0 | | | | | | | +3  7  6  5  4  3  2  1  0 |
| Fmt  0  0  1 | Type  1  0  1  0  0 | T9 | TC  0  0  0 | T8 | Attr | R | TH | TD | EP | Attr | | R | | Length  0  0  0  0  0  0  0  0  0  0 | |
| Requester ID | | | | | | | |  | | | Tag | |  | | Message Code  0  0  0  1  0  0  0  0 |
| Reserved | | | | | | | | Reserved | | | | | | | |
| No-Snoop Latency | | | | | | | | Snoop Latency | | | | | | | |

Figure 2-34 LTR Message

[**2.2.8.9**](2.2.8.9) **Optimized Buffer Flush/Fill (OBFF) Message**

The OBFF Message is optionally used to report platform central resource states to Endpoints. This mechanism is described in detail in Section 6.19 .

The following rules apply to the formation of the OBFF Message:

• [Table 2-33](#bookmark133)defines the OBFF Message.

• The OBFF Message does not include a data payload (TLP Type is Msg).

• The Length field is Reserved.

• The Requester ID must be set to the Transmitting Port's ID.

• The OBFF Message must use the default Traffic Class designator (TC0). Receivers that implement OBFF support 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 (seeSection 6.2).

Table 2-33 OBFF Message

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Code[7:0] (b) | Routing r[2:0] (b) | Support1 | | | | Description/Comments |
| RC | Ep | Sw | Br |
| OBFF | 0001 0010 | 100 | t | r | tr |  | Optimized Buffer Flush/Fill |

Notes:



5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Code[7:0] (b) | Routing r[2:0] (b) | Support1 | | | | Description/Comments |
| RC | Ep | Sw | Br |

1. Support for OBFF is optional. Functions that support OBFF must implement the reporting and enable mechanisms described in Chapter 7 , Software Initialization and Configuration.



Byte 0 → Byte 4 → Byte 8 → Byte 12 →

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| +0  7  6  5  4  3  2  1  0 | | +1  7  6  5  4  3  2  1 0 | | | | | | +2  7  6  5  4  3  2  1  0 | | | | | | | +3  7  6  5  4  3  2  1  0 | |
| Fmt  0  0  1 | Type  1  0  1  0  0 | T9 | TC  0  0  0 | T8 | Attr | R | TH | TD | EP | Attr | | R | | Length  0  0 0  0  0  0  0  0  0  0 | | |
| Requester ID | | | | | | | |  | | | Tag | |  | | Message Code  0  0  0  1  0  0  1  0 | |
| Reserved | | | | | | | | | | | | | | | | |
| Reserved | | | | | | | | | | | | | | | | OBFF Code |

Figure 2-35 OBFF Message

[**2.2.8.10**](2.2.8.10) **Precision Time Measurement (PTM) Messages**

[Table 2-34](#bookmark135)defines the PTM Messages.

• The PTM Request and PTM Response Messages must use a TLP Type of Msg, and must not include a data payload. The Length field is reserved.

。[Figure 2-36](#bookmark136)illustrates the format of the PTM Request and Response Messages.

• The PTM ResponseD Message must use a TLP Type of MsgD, and must include a 64 bit PTM Master Time field in bytes 8 through 15 of the TLP header and a 1 DW data payload containing the 32 bit Propagation Delayfield.

。[Figure 2-37](#bookmark137)illustrates the format of the PTM ResponseD Message.

。 Refer toSection 6.22.3.2for details regarding how to populate the PTM ResponseD Message.

• The Requester ID must be set to the Transmitting Port's ID.

• A PTM dialog is defined as a matched pair of messages consisting of a PTM Request and the corresponding PTM Response or PTM ResponseD message.

• The PTM Messages must use the default Traffic Class designator (TC0). Receivers implementing PTM 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 (seeSection 6.2).

Table 2-34 Precision Time Measurement Messages

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Name | TLP Type | Code[7:0]  (b) | Routing r[2:0]  (b) | Support | | | | Description/Comments |
| RC | EP | Sw | Br |
| PTM Request | Msg | 0101 0010 | 100 | r | t | tr |  | Initiates PTM dialog |

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Name | TLP Type | Code[7:0]  (b) | Routing r[2:0]  (b) | Support | | | | Description/Comments |
| RC | EP | Sw | Br |
| PTM  Response | Msg | 0101 0011 | 100 | t | r | tr |  | Completes current PTM dialog - does not carry timing information |
| PTM  ResponseD | MsgD | 0101 0011 | 100 | t | r | tr |  | Completes current PTM dialog - carries timing information |

Byte 0 → Byte 4 → Byte 8 → Byte 12 →

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| +0  7  6  5  4  3  2  1  0 | | +1  7  6  5  4  3  2  1  0 | | | | | | +2  7  6  5  4  3  2  1  0 | | | | | | | +3  7  6  5  4  3  2  1  0 |
| Fmt  0  0  1 | Type  1  0  1  0  0 | T9 | TC  0  0  0 | T8 | Attr | LN | TH | TD | EP | Attr | | R | | Reserved | |
| Requester ID | | | | | | | |  | | | Tag | |  | | Message Code  0  1  0  1  0  0  1  x |
| Reserved | | | | | | | | | | | | | | | |
| Reserved | | | | | | | | | | | | | | | |

Figure 2-36 PTM Request/Response Message

Byte 0 → Byte 4 → Byte 8 → Byte 12 → Byte 16 →

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| +0  7  6  5  4  3  2  1  0 | | +1  7  6  5  4  3  2  1 0 | | | | | | +2  7  6  5  4  3  2  1  0 | | | | | | | +3  7  6  5  4  3  2  1  0 |
| Fmt  0  1  1 | Type  1  0  1  0  0 | T9 | TC | T8 | Attr | LN | TH | TD | EP | Attr | | R | | Length  0  0  0  0  0  0  0  0  0  1 | |
| Requester ID | | | | | | | |  | | | Tag | |  | | Message Code  0  1  0  1  0  0  1  1 |
| PTM Master Time [63:32] | | | | | | | | | | | | | | | |
| PTM Master Time [31:0] | | | | | | | | | | | | | | | |
| Propagation Delay [31:0] | | | | | | | | | | | | | | | |

Figure 2-37 PTMResponseD Message (4 DW header and 1 DW payload)

**2.2.9 Completion Rules**

All Read, Non-Posted Write, and AtomicOp Requests require Completion. Completions include a Completion header

that, for some types of Completions, will be followed by some number of DWs of data. The rules for each of the fields of the Completion header are defined in the following 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.

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

• In addition to the header fields included in all TLPs and the ID routing fields, Completions contain the following additional fields (see[Figure 2-38](#bookmark140)):

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

。 Completion Status[2:0] - Indicates the status for a Completion (see [Table 2-35)](#bookmark141)

▪ Rules for determining the value in the Completion Status[2:0] field are in [Section 2.3.1 .](#bookmark142)

。 BCM - Byte Count Modified -this bit must not be 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.

▪ For Memory Read Completions, Byte Count[11:0] is set according to the rules in [Section](#bookmark143)

[2.3.1.1 .](#bookmark139)

▪ For AtomicOp Completions, the Byte Count value must equal the associated AtomicOp operand size in bytes.

▪ For all other types of Completions, the Byte Count value must be 4.

。 Tag[9:0] - in combination with the Requester ID field, corresponds to the Transaction ID 。 Lower Address[6:0] - lower byte address for starting byte of Completion

▪ 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)](#bookmark144).

▪ For AtomicOp Completions, the Lower Address field is Reserved.

▪ This field is set to all 0's for all remaining types of Completions. Receivers may optionally check for violations of this rule. See [Section 2.3.2 ,](#bookmark145) second bullet, for details.

Byte 0 → Byte 4 → Byte 8 →

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| +0  7  6  5  4  3  2  1  0 | | +1  7  6  5  4  3  2  1  0 | | | | | | +2  7  6  5  4  3  2  1  0 | | | | | | | | +3  7  6  5  4  3  2  1  0 | |
| Fmt  0  x  0 | Type | T9 | TC | T8 | Attr | LN | TH | TD | EP | Attr | | | AT  0  0 | | Length | | |
| Completer ID | | | | | | | | Cpl. Status | | | | BCM | Byte Count | | | | |
| Requester ID | | | | | | | |  | | | Tag | | |  | | R | Lower Address |

Figure 2-38 Completion Header Format Table 2-35 Completion Status Field Values

|  |  |
| --- | --- |
| Cpl. Status[2:0] Field Value (b) | Completion Status |
| 000 | Successful Completion (SC) |
| 001 | Unsupported Request (UR) |
| 010 | Configuration Request Retry Status (CRS) |
| 100 | Completer Abort (CA) |

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

|  |  |
| --- | --- |
| Cpl. Status[2:0] Field Value (b) | Completion Status |
| 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 2-39](#bookmark146)and [Figure 2-40)](#bookmark147)

Completer ID

|  |  |  |
| --- | --- | --- |
| 7:0 | 4:0 | 2:0 |

Bus Number Device

Number

Function Number

OM13770

Figure 2-39 (Non-ARI) Completer ID

Completer ID

|  |  |
| --- | --- |
| 7:0 | 7:0 |

Bus Number Function Number

A-0718

Figure 2-40 ARI Completer ID

• Functions must capture the Bus and Device Numbers21 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 ID22 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 Devices within a Root Complex may be done in an implementation specific way.

• In some cases, a Completion with UR status may be generated by an MFD without associating the Completion with a specific Function within the device - in this case, the Function Number field23 is Reserved.



21. With ARI Devices, Functions are only required to capture the Bus Number. ARI Devices are permitted to retain the captured Bus Number on either a per-Device or a per-Function basis. See[Section 2.2.6.2 .](#bookmark59)

22.AnARI Completer ID does not contain a Device Number field. See [Section 2.2.4.2 .](#bookmark40)

23. Note: with an ARI Completer ID, the Function Number field is 8 bits.

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

。 Example: An MFD receives a Read Request that 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, and Traffic Class as were supplied in the header of the corresponding Request.

• Completion headers must supply the same values for the Attribute as were supplied in the header of the corresponding Request, except as explicitly allowed:

。 when IDO is used (see [Section 2.2.6.4)](#bookmark66)

。 when RO is used in a Translation Completion (see Section 10.2.3

• If the Completer is an LN Completer (LNC)and the targeted memory region supports registrations, the following rules apply; otherwise the LN bit must be Clear.

。 If the Completion Status is Successful Completion and the associated Request was an LN Read, the LN bit must be Set.

。 Otherwise the LN bit must be Clear.

• The TH bit is reserved for Completions.

• AT[1:0] must be 00b. Receivers are not required or encouraged to check this.

• 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 for this case the Requester 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 the TLP data payload length to match the value in the Length field.

**2.2.10 TLP Prefix Rules**

The following rules apply to any TLP that contains a TLP Prefix:

• For any TLP, a value of 100b in the Fmt[2:0] field in byte 0 of the TLP indicates the presence of a TLP Prefix and the Type[4] bit indicates the type of TLP Prefix.

。 A value of 0b in the Type[4] bit indicates the presence of a Local TLP Prefix

。 A value of 1b in the Type[4] bit indicates the presence of an End-End TLP Prefix

• The format for bytes 1 through 3 of a TLP Prefix are defined by its TLP Prefix type.

• ATLP that contains a TLP Prefix must have an underlying TLP Header. A received TLP that violates this rule is handled as a Malformed TLP. This is a reported error associated with the Receiving Port (seeSection 6.2).

• It is permitted for a TLP to contain more than one TLP Prefix of any type

。 When a combination of Local and End-End TLP Prefixes are present in TLP, it is required that all the Local TLP Prefixes precede any End-End TLP Prefixes. A received TLP that violates this rule is handled as a Malformed TLP. This is a reported error associated with the Receiving Port (seeSection 6.2).

• The size of each TLP Prefix is 1 DW. ATLP Prefix may be repeated to provide space for additional data.

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

• If the value in the Fmt and Type field indicates the presence of a Local TLP Prefix, handle according to the Local TLP Prefix handling (see [Section 2.2.10.1)](#bookmark149).

• If the value in the Fmt and Type field indicates the presence of an End-End TLP Prefix, handle according to the End-End TLP Prefix handling (see [Section 2.2.10.2)](#bookmark150).

[**2.2.10.1**](2.2.10.1) **Local TLP Prefix Processing**

The following rules apply to Local TLP Prefixes:

• Local TLP Prefix types are determined using the L[3:0] sub-field of the Type field 。 Type[4] must be 0b

。 Local TLP Prefix L[3:0] values are defined in [Table 2-36](#bookmark151)

Table 2-36 Local TLP Prefix Types

|  |  |  |
| --- | --- | --- |
| Local TLP Prefix Type | L[3:0] (b) | Description |
| MR-IOV | 0000 | **MR-IOVTLP Prefix**- Refer to [MR-IOV] specification for details. |
| VendPrefixL0 | 1110 | **Vendor Defined Local TLP Prefix**- Refer to[Section 2.2.10.1.1f](#bookmark152)or further details. |
| VendPrefixL1 | 1111 | **Vendor Defined Local TLP Prefix**- Refer to[Section 2.2.10.1.1f](#bookmark153)or further details. |
|  |  | All other encodings are Reserved. |

• The size, routing, and flow control rules are specific to each Local TLP Prefix type.

• It is an error to receive a TLP with a Local TLP Prefix type not supported by the Receiver. If the Extended Fmt Field Supported bit is Set, TLPs in violation of this rule are handled as a Malformed TLP unless explicitly stated differently in another specification. This is a reported error associated with the Receiving Port (seeSection 6.2 ). If the Extended Fmt Field Supported bit is Clear, behavior is device specific.

• No Local TLP Prefixes are protected by ECRC even if the underlying TLP is protected by ECRC.

**2.2.10.1.1 Vendor Defined Local TLP Prefix**

As described in [Table 2-36 ,](#bookmark151) Types VendPrefixL0 and VendPrefixL1 are Reserved for use as Vendor Defined Local TLP

Prefixes. To maximize interoperability and flexibility the following rules are applied to such prefixes:

• Components must not send TLPs containing Vendor Defined Local TLP Prefixes unless this has been explicitly enabled (using vendor-specific mechanisms).

• Components that support any usage of Vendor Defined Local TLP Prefixes must support the 3-bit definition of the Fmt field and have the Extended Fmt Field Supported bit Set (seeSection 7.5.3.15).

• It is recommended that components be configurable (using vendor-specific mechanisms) so that all vendor defined prefixes can be sent using either of the two Vendor Defined Local TLP Prefix encodings. Such

configuration need not be symmetric (for example each end of a Link could transmit the same Prefix using a

different encoding).

[**2.2.10.2**](2.2.10.2) **End-End TLP Prefix Processing**

The following rules apply to End-End TLP Prefixes

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

• End-End TLP Prefix types are determined using the E[3:0] sub-field of the Type field 。 Type[4] must be 1b

。 End-End TLP Prefix E[3:0] values are defined in [Table 2-37](#bookmark26)

Table 2-37 End-End TLP Prefix Types

|  |  |  |
| --- | --- | --- |
| End-End TLP Prefix Type | E[3:0] (b) | Description |
| TPH | 0000 | **TPH** - Refer to[Section 2.2.7.1](#bookmark55)and Section 6.17for further details. |
| PASID | 0001 | **PASID** - Refer toSection 6.20for further details. |
| VendPrefixE0 | 1110 | **Vendor Defined End-End TLP Prefix**- Refer to[Section 2.2.10.2.1f](#bookmark154)or further details. |
| VendPrefixE1 | 1111 | **Vendor Defined End-End TLP Prefix**- Refer to[Section 2.2.10.2.1f](#bookmark155)or further details. |
|  |  | All other encodings are Reserved. |

• The maximum number of End-End TLP Prefixes permitted in a TLP is 4:

。 A Receiver supporting TLP Prefixes must check this rule. If a Receiver determines that a TLP violates this rule, the TLP is a Malformed TLP. This is a reported error associated with the Receiving Port (see Section 6.2).

• The presence of an End-End TLP Prefix does not alter the routing of a TLP. TLPs are routed based on the routing rules covered in [Section 2.2.4 .](#bookmark33)

• Functions indicate how many End-End TLP Prefixes they support by the Max End-End TLP Prefixes field in the Device Capabilities 2 register (see Section 7.5.3.15).

。 For Root Ports, the Max End-End TLP Prefixes field is permitted to return a value indicating support for fewer End-End TLP Prefixes than what the Root Port hardware actually implements; however, the error handling semantics must still be based on the value contained in the field. TLPs received that contain more End-End TLP Prefixes than are supported by the Root Port must be handled as follows. It is recommended that Requests be handled as Unsupported Requests, but otherwise they must be handled as Malformed TLPs. It is recommended that Completions be handled as Unexpected

Completions, but otherwise they must be handled as Malformed TLPs. For TLPs received by the

Ingress Port, this is a reported error associated with the Ingress Port. For TLPs received internally to be transmitted out the Egress Port, this is a reported error associated with the Egress Port. See

Section 6.2 .

。 For all other Function types,TLPs received that contain more End-End TLP Prefixes than are

supported by a Function must be handled as Malformed TLPs. This is a reported error associated with the Receiving Port (seeSection 6.2).

Advanced Error Reporting (AER) logging (if supported) occurs as specified in Section 6.2.4.4 .

• Switches must support forwarding of TLPs with up to 4 End-End TLP Prefixes if the End-End TLP Prefix Supported bit is Set.

• Different Root Ports with the End-End TLP Prefix Supported bit Set are permitted to report different values for Max End-End TLP Prefixes.

• All End-End TLP Prefixes are protected by ECRC if the underlying TLP is protected by ECRC.

• It is an error to receive a TLP with an End-End TLP Prefix by a Receiver that does not support End-End TLP

Prefixes. ATLP in violation of this rule is handled as a Malformed TLP. This is a reported error associated with the Receiving Port (seeSection 6.2).

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

• Software should ensure that TLPs containing End-End TLP Prefixes are not sent to components that do not support them. Components where the Extended Fmt Field Supported bit is Clear may misinterpret TLPs

containing TLP Prefixes.

• If one Function of an Upstream Port has the End-End TLP Prefix Supported bit Set, all Functions of that

Upstream Port must handle the receipt of a Request addressed to them that contains an unsupported End-End TLP Prefix type as an Unsupported Request. This is a reported error associated with the Receiving Port (see

Section 6.2).

• If one Function of an Upstream Port has the End-End TLP Prefix Supported bit Set, all Functions of that

Upstream Port must handle the receipt of a Completion addressed to them that contains an unsupported

End-End TLP Prefix type as an Unexpected Completion. This is a reported error associated with the Receiving Port (seeSection 6.2).

• For Routing Elements,the End-End TLP Prefix Blocking bit in each Egress Port determines whether TLPs

containing End-End TLP Prefixes can be transmitted via that Egress Port (seeSection 7.5.3.16). If forwarding is blocked the entire TLP is dropped and a TLP Prefix Blocked Error is reported. If the blocked TLP is a

Non-Posted Request, the Egress Port returns a Completion with Unsupported Request Completion Status. The TLP Prefix Blocked Error is a reported error associated with the Egress Port (seeSection 6.2).

• For routing elements where Multicast is enabled (see Section 6.14). End-End TLP Prefixes are replicated in all Multicast copies of a TLP. TLP Prefix Egress Blocking of Multicast packets is performed independently at each Egress Port.

**2.2.10.2.1 Vendor Defined End-End TLP Prefix**

As described in [Table 2-37 ,](#bookmark26) Types VendPrefixE0 and VendPrefixE1 are Reserved for use as Vendor Defined End-End TLP Prefixes. To maximize interoperability and flexibility the following rules are applied to such prefixes:

• Components must not send TLPs containing Vendor Defined End-End TLP Prefixes unless this has been explicitly enabled (using vendor-specific mechanisms).

• It is recommended that components be configurable (using vendor-specific mechanisms) to use either of the two Vendor Defined End-End TLP Prefix encodings. Doing so allows two different Vendor Defined End-End TLP Prefixes to be in use simultaneously within a single PCI Express topology while not requiring that every source understand the ultimate destination of every TLP itsends.

**2.2.10.2.2 Root Ports with End-End TLP Prefix Supported**

Support for peer-to-peer routing of TLPs containing End-End TLP Prefixes between Root Ports is optional and

implementation dependent. If an RC supports End-End TLP Prefix routing capability between two or more Root Ports, it must indicate that capability in each associated Root Port via the End-End TLP Prefix Supported bit in the Device

Capabilities 2 register.

An RC is not required to support End-End TLP Prefix routing between all pairs of Root Ports that have the End-End TLP Prefix Supported bit Set. A Request with End-End TLP Prefixes that would require routing between unsupported pairs of Root Ports must be handled as a UR. A Completion with End-End TLP Prefixes that would require routing between

unsupported pairs of Root Ports must be handled as an Unexpected Completion (UC). In both cases, this error is reported by the “sending” Port.

The End-End TLP Prefix Supported bit must be Set for any Root Port that supports forwarding of TLPs with End-End TLP Prefixes initiated by host software or Root Complex Integrated Endpoints (RCiEPs). The End-End TLP Prefix Supported bit must be Set for any Root Ports that support forwarding of TLPs with End-End TLP Prefixes received on their Ingress Port to RCiEPs.

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

Different Root Ports with the End-End TLP Prefix Supported bit Set are permitted to report different values for Max End-End TLP Prefixes.

An RC that splits a TLP into smaller TLPs when performing peer-to-peer routing between Root Ports must replicate the original TLP's End-End TLP Prefixes in each of the smaller TLPs (seeSection 1.3.1).

**2.3 Handling of Received TLPs**

This section describeshow 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 2-41 .](#bookmark156)

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

• If the value in the Fmt field indicates the presence of at least one TLP Prefix:

。 Detect if additional TLP Prefixes are present in the header by checking the Fmt field in the first byte of subsequent DWs until the Fmt field does not match that of a TLP Prefix.

。 Handle all received TLP Prefixes according to TLP Prefix Handling Rules (see [Section 2.2.10)](#bookmark148).

• If the Extended Fmt Field Supported bit is Set, Received TLPs that use encodings of Fmt and Type that are Reserved are Malformed TLPs (see[Table 2-1](#bookmark1)and [Table 2-3)](#bookmark21).

。 This is a reported error associated with the Receiving Port (seeSection 6.2).

• If the Extended Fmt Field Supported bit is Clear, processing of Received TLPs that have Fmt[2] Set is undefined.24

• All Received TLPs with Fmt[2] Clear and that use undefined Type field values are Malformed TLPs. This is a reported error associated with the Receiving Port (seeSection 6.2).

• All Received Malformed TLPs must be discarded.

。 Received Malformed TLPs that are ambiguous with respect to which buffer to release or are mapped to an uninitialized or disabled Virtual Channel must be discarded without updating Receiver Flow

Control information.

。 All other Received Malformed TLPs must be discarded, optionally not updating Receiver Flow Control information.

• Otherwise, update Receiver Flow Control tracking information (see [Section 2.6)](#bookmark157).

• 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).



24. An earlier version of this specification reserved the bit now defined for Fmt[2].

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

Start

No



Ignoring

Reserved Fields\*, Does

TLP follow formation

rules?

Yes



Is value in Type field Defined?

Yes

No

End

|  |
| --- |
| TLP is Malformed:  Discard TLP  . Report Malformed Packet |

TLP is a Completion-

See rules for

Completion Handling

|  |
| --- |
| Update Flow Control tracking |



No

Is TLP a Request?

Yes

TLP is a Request-

See rules for Request Handling

\*TLP fields which are marked Reserved are not checked at the Receiver

OM13771A

Figure 2-41 Flowchart for Handling of Received TLPs

Switches must process both TLPs that address resources within the Switch as well as TLPs that address resources

residing outside the Switch. Switches handle all TLPs that address internal resources of the Switch according to the rules above. TLPs that pass through the Switch, or that address the Switch as well as passing through it, are handled

according to the following rules (see[Figure 2-42](#bookmark158)):

• 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 [Section 2.2.4.1](#bookmark35)and [Section 2.2.4.2)](#bookmark40).

• 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 (seeSection 6.2

).

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

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

。 If the value in r[2:0] indicates the Msg/MsgDis 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 that must be comprehended by the Switch, the Switch must process the Message according to the Message processing rules.

。 If the value in r[2:0] indicates Gathered and routed to Root Complex (101b), see Section 5.3.3.2.1for Message handling rules.

。 It is an error to receive any Msg/MsgD Request other than a [PME\_TO\_Ack](#bookmark93)that specifies 101b routing. It is an error to receive a [PME\_TO\_Ack](#bookmark93)at the Upstream Port of a Switch. Switches may optionally

check for violations of these rules. These checks are independently optional (see Section 6.2.3.4). If checked, violations are Malformed TLPs, and are reported errors associated with the Receiving Port (see Section 6.2).

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

Start

No

Route TLP to Egress Port according to routing rules



Is TLP a

Msg or MsgD

Request?

Yes

End

|  |
| --- |
| Propagate to Egress Port(s)  according to r[2:0] sub-field  of Type field |

Is Message (also)

No

End

directed to Receiving

Port of Switch?

Yes

Is value in Message Code field defined?

Yes

End

No

End

|  |
| --- |
| Process Message according to Message handling rules |

|  |
| --- |
| Unsupported Request |

OM13772A

Figure 2-42 Flowchart for Switch Handling of TLPs

**2.3.1 Request Handling Rules**

This section describeshow Received Requests are handled, following the initial processing done with all TLPs. The rules are diagramed in the flowchart shown in [Figure 2-43 .](#bookmark159)

• If the Request Type is not supported (by design or because of configuration settings) by the device, the Request is an Unsupported Request, and is reported according toSection 6.2

◦ If the Request requires Completion, a Completion Status of UR is returned (see [Section 2.2.8.10)](#bookmark134)

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0



**IMPLEMENTATION NOTE**

When Requests are Terminated Using Unsupported Request

In Conventional PCI, a device “claims” a request on the busby asserting DEVSEL#. If no device claims a request after a set number of clocks, the request is terminated as a Master Abort. Since PCI Express is a point to point

interconnect, there is no equivalent mechanism for claiming a request on a Link, since all transmissions by one component are always sent to the other component on the Link. Therefore, it is necessary for the receiver of a request to determine if the request should be “claimed” If the request is not claimed, then it is handled as an

Unsupported Request, which is the PCI Express equivalent of Conventional PCI's Master Abort termination. In

general, one can determine the correct behavior by asking the question: Would the device assert DEVSEL# for this request in conventional PCI?

For device Functions with Type 0 headers (all types of Endpoints), it is relatively simple to answer this question. For Memory and I/O Requests, this determination is based on the address ranges the Function has been

programmed to respond to. For Configuration requests, the Type 0 request format indicates the device is by definition the “target”, although the device will still not claim the Configuration Request if it addresses an unimplemented Function.

For device Functions with Type 1 headers (Root Ports, Switches and Bridges), the same question can generally be applied, but since the behavior of a conventional PCI bridge is more complicated than that of a Type 0 Function, it is somewhat more difficult to determine the answers. One must consider Root Ports and Switch Ports as if they

were actually composed of conventional PCI to PCI bridges, and then at each stage consider the configuration settings of the virtual bridge to determine the correct behavior.

PCI Express Messages do not exist in conventional PCI, so the above guideline cannot be applied. This

specification describes specifically for each type of Message when a device must handle the request as an

Unsupported Request. Messages pass through Root and Switch Ports unaffected by conventional PCI control mechanisms including Bus Master Enable and power state setting.

Note that CA, which is the PCI Express equivalent to Target Abort, is used only to indicate a serious error that

makes the Completer permanently unable to respond to a request that it would otherwise have normally

responded to. Since Target Abort is used in conventional PCI only when a target has asserted DEVSEL#, is

incorrect to use a CA for any case where a Conventional PCI target would have ignored a request by not asserting DEVSEL#.

• If the Request is a Message, and the Message Code, routing field, or Msg / MsgDindication corresponds to a

combination that is undefined, or that corresponds to a Message not supported by the device Function, (other than Vendor\_Defined Type 1, which is not treated as an error - see Table F-1), the Request is an Unsupported Request, and is reported according toSection 6.2

。 If the Message Code is a supported value, process the Message according to the corresponding

Message processing rules; if the Message Code is an Ignored Message and the Receiver is ignoring it, ignore the Message without reporting any error (see[Section 2.2.8.7)](#bookmark130)

• If the Request is a Message with a routing field that indicates Routed by ID, and if the Request is received by a device Function with Type 0 headers, it is strongly recommended that the device be treated as the target of the Message regardless of the Bus Number and Device Number specified in the destination ID field of the Request

。 If the Function specified in the destination ID is unimplemented, it is strongly recommended that the Request be handled as an Unsupported Request, and that it is reported as specified in Section 6.2

If the Request is not a Message, and is a supported Type, specific implementations may be optimized based on a defined

programming model that ensures that certain types of (otherwise legal) Requests will never occur. Such implementations may take advantage of the following rule:

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

• If the Request violates the programming model of the device Function, the Function 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 Function (see Section 6.2)

。 If the Request requires Completion, a Completion Status of CA is returned (see [Section 2.2.8.10)](#bookmark134)

**IMPLEMENTATION NOTE**

Optimizations Based on Restricted Programming Model

When a device's programming model restricts (versus what is otherwise permitted in PCI Express) the

characteristics of a Request, that device is permitted to return a CA Completion Status for any Request that violates 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 that may be accessed directly by operating system

software or by applications that may not comprehend the restricted programming model of the device (typically devices that implement legacy capabilities) should be designed to support all types of Requests that are possible in the existing usage model for the device. If this is not done, the device may fail to operate with existing software.

If the Request arrives between the time an FLR has been initiated and the completion of the FLR by the targeted Function, the Request is permitted to be silently discarded (following update of flow control credits) without

logging or signaling it as an error. It is recommended that the Request be handled as an Unsupported Request (UR).



• 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 Function, if the error can be isolated to a specific 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 (CRS) Completion Status is used (see Section 6.6). Valid reset conditions after which a device is permitted to return CRS are:

▪ Cold,Warm, and Hot Resets

▪ FLRs

▪ A reset initiated in response to a D3HottoD0uninitializeddevice state transition

。 A device Function is explicitly not permitted to return CRS following a software-initiated reset (other than an FLR) of the device, e.g., by the device's software driver writing to a device-specific reset bit. A device Function is not permitted to return CRS after it has indicated that it is Configuration-Ready

(see Section 6.23.) without an intervening valid reset (i.e., FLR or Conventional Reset) condition, or if the Immediate Readiness bit in the Function's Status register is Set. Additionally, a device Function is not permitted to return CRS after having previously returned a Successful Completion without an

intervening valid reset (i.e., FLR or Conventional Reset) condition.

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

。 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 Space 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 that requires a Completion to be returned, generate a Completion according to the rules for Completion formation (see [Section 2.2.9)](#bookmark138)

。 The Completion Status is determined by the result of handling the Request

。 If the Request has an ECRC Check Failed error, then it is implementation-specific whether to return a Completion or not, and if so, which of the architected values to use for its Completion Status.

However, it is strongly recommended that the Completer return a Completion with a UR Completion Status.

• Under normal operating conditions, PCI Express Endpoints and Legacy Endpoints must never delay the

acceptance of a Posted Request for more than 10 μs, which is called the Posted Request Acceptance Limit. The device must either (a) be designed to process received Posted Requests and return associated Flow Control

credits within the necessary time limit, or (b) rely on a restricted programming model to ensure that a Posted Request is never sent to the device either by software or by other devices while the device is unable to accept a new Posted Request within the necessary time limit.

。 The following are not considered normal operating conditions under which the Posted Request Acceptance Limit applies:

▪ The period immediately following a Fundamental Reset (seeSection 6.6)

▪ TLP retransmissions or Link retraining

▪ One or more dropped Flow Control Packets (FCPs)

▪ The device being in a diagnostic mode

▪ The device being in a device-specific mode that is not intended for normal use

。 The following are considered normal operating conditions, but any delays they cause do not count against the Posted Request Acceptance Limit:

▪ Upstream TLP traffic delaying Upstream FCPs

▪ The Link coming out of a low-power state

▪ Arbitration with traffic on other VCs

。 Though not a requirement, it is strongly recommended that RCiEPs also honor the Posted Request Acceptance Limit.

• If the device supports being a target for I/O Write Requests, which are Non-Posted Requests, it is strongly

recommended that each associated Completion be returned within the same time limit as for Posted Request acceptance, although this is not a requirement.

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

**IMPLEMENTATION NOTE**

Restricted Programming Model for Meeting the Posted Request Acceptance Limit

Some hardware designs may not be able to process every Posted Request within the required acceptance time limit. An example is writing to a command queue where commands can take longer than the acceptance time

limit to complete. Subsequent writes to such a device when it is currently processing a previous write could

experience acceptance delays that exceed the limit. Such devices may rely on a restricted programming model, where the device driver limits the rate of memory writes to the device, the driver polls the device to determine buffer availability before issuing the write transaction, or the driver implements some other software-based flow control mechanism.



5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

Start

No



Is Request Type Supported?

Yes

|  |
| --- |
| Unsupported Request |

Yes

Does Request

No

require a

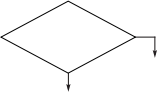
Completion?



Request Type = Msg?

No

End



Does Request

require a

Completion?

Yes

Send Completion:

. Completion Status = UR

End

Send Completion:

. Completion Status = CA

|  |  |  |
| --- | --- | --- |
| Yes  Optional  Does Request violate  Device programming  model?  No | Yes | No  End |

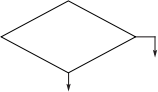
End

No Illid?e

 Yes

|  |
| --- |
| Process Request according to  Request handling rules  (determine Completion Status,  if applicable) |





Does Request require a Completion?

Yes

|  |
| --- |
| Process Message  according to Message  Handling Rules |

|  |
| --- |
| Unsupported Request |

End

No

End

End

Send Completion

End

OM13773

Figure 2-43 Flowchart for Handling of Received Request

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0



**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 225 (PCI) or 226 (PCI-X) clock “recovery time” Trhfa following reset to provide the required self-initialization time for such devices.Section 6.6.1specifies a 1.0 s recovery period for PCIe devices. PCIe architecture also provides an alternative to waiting for this worst-case recovery period via the Configuration Request Retry Status (CRS)

Completion Status mechanism. A device in receipt of a Configuration Request following a valid reset condition may respond with a CRS 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 CRS Completion Status in response to a Configuration

Request. Sending this Completion Status in response to any other Request type is illegal (see [Section 2.3.2)](#bookmark160).

Readiness Notifications (see Section 6.23) and Immediate Readiness (see Section 7.5.1.1.4and Section 7.5.2.1) also forbid the use of CRS Completion Status in certain situations.

Receipt by the Requester of a Completion with CRS Completion Status terminates the Configuration Request on PCI Express. Further action by the Root Complex regarding the original Configuration Request is specified in

[Section 2.3.2 .](#bookmark161)

Root Complexes that implement CRS Software Visibility have the ability to report the receipt of CRS Completion Status to software, enabling software to attend to other tasks rather than being stalled while the device

completes its self-initialization. Software that intends to take advantage of this mechanism must ensure that the first access made to a device following a valid reset condition is a Configuration Read Request accessing both

bytes of the Vendor ID field in the device's Configuration Space header. For this case only, the Root Complex, if enabled, will synthesize a special read-data value for the Vendor ID field to indicate to software that CRS

Completion Status has been returned by the device. For other Configuration Requests, or when CRS Software Visibility is not enabled, the Root Complex will generally re-issue the Configuration Request until it completes with a status other than CRS as described in [Section 2.3.2 .](#bookmark162)

To avoid misbehaviors in systems that contain PCI Express to PCI/PCI-X Bridges, system software and/or the Root Complex should comprehend the limit Trhfa for PCI/PCI-X agents as described in Section 2.8and Section 6.6 .

Similarly, systems that contain PCIe components whose self-initialization time may require them to return a CRS Completion Status (by the rules in Section 6.6) should 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.

Refer toSection 6.6for more information on reset.

[**2.3.1.1**](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.

。 Completions for different Requests cannot be combined.

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

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

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

▪ 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 Max\_Payload\_Size. 。 Receivers must check for violations of this rule. Refer to[Section 2.2 .](#bookmark6)

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

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

• Read Completion Boundary (RCB) determines the naturally aligned address boundaries on which a Completer is permitted to break up the response for a single Read Request into multiple Completions.

。 For a Root Complex, RCB is 64 bytes or 128 bytes.

▪ This value is reported in the Link Control register (seeSection 7.5.3.7).

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

• For all other System Elements, RCB is 128 bytes.

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

• Requests that do cross the address boundaries at integer multiples of RCB bytes are permitted to be completed using more than one Completion subject to the following rules:

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

▪ The address that satisfies the entire Request

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

。 If the final Completion is sucessful, it must end at the address that satisfies the entire 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 (seeSection 6.2).

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

• If all the Memory Read Completions for a single Read Request have a Successful Completion Status, the sum of their payloads must equal the size requested.

• 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

bit is Set.25

。 The total number of bytes required to complete a Memory Read Request is calculated as shown in [Table 2-38 .](#bookmark163)



25. Only PCI-X completers Set the BCM bit. PCI Express completers are not permitted to set the BCM bit.

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

◦ 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.

• The Completion Data area begins at the DW address specified by the Request. In the first or only Data DW of the first or only Completion, only the bytes configured as active in the First BE field in the Request contain valid

data. Bytes configured as inactive in the First BE field in the Request will return undefined content.

• In the last Data DW of the last successful Completion, only the bytes configured as active in the Last BE field in the Request contain valid data. Bytes configured as inactive in the Last BE field in the Request will return

undefined content.

• All the Completion Data bytes, including those with undefined content, are included in all CRC calculations.

• [Figure 2-44](#bookmark164)presents an example of the above. The example assumes a single Completion TLP is returned.

Request

Address (DW)

START

START + 1

START + 2

START + 3

Byte 0 Byte 1 Byte 2 Byte 2

|  |  |  |  |
| --- | --- | --- | --- |
| undefined content | undefined content | undefined content |  |
|  |  |  |  |
|  |  |  |  |
|  | undefined content | undefined content | undefined content |

Request

Byte Enables

First BE: 1000

Last BE: 0001

Length = 4d;

Byte Count = 10d;

Figure 2-44 Example Completion Data when some Byte Enables are 0b

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

**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] 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 Requester 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 2-38 Calculating Byte Count from Length and Byte Enables

|  |  |  |
| --- | --- | --- |
| First DW BE[3:0] (b) | Last DW BE[3:0] (b) | Total Byte Count |
| 1xx1 | 000026 | 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 | Length27 \* 4 |
| xxx1 | 01xx | (Length \* 4) - 1 |
| xxx1 | 001x | (Length \* 4) - 2 |



26. Note that Last DW BE of 0000b is permitted only with a Length of 1 DW.

27. 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.

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

|  |  |  |
| --- | --- | --- |
| First DW BE[3:0] (b) | Last DW BE[3:0] (b) | Total Byte Count |
| 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 |

• 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 5 bits of the address of the Request concatenated with 2 bits of byte-level address formed as shown in [Table 2-39 .](#bookmark165)

。 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 2-39 Calculating Lower Address from First DW BE

|  |  |
| --- | --- |
| First DW BE[3:0] (b) | Lower Address[1:0] (b) |
| 0000 | 00 |
| xxx1 | 00 |
| xx10 | 01 |
| x100 | 10 |
| 1000 | 11 |

• 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

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

. 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 that 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 that the device will complete with

Successful Completion status.

**Examples:**

1. Memory Read Request with Address of 1 0000hand 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 1 0000hand 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 1 0020hand 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 1 0020hand 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

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

**2.3.2 Completion Handling Rules**

• When a device receives a Completion that does not match the Transaction ID for any of the outstanding Requests issued by that device, the Completion is called an “Unexpected Completion”

• If a received Completion matches the Transaction ID of an outstanding Request, but in some other way does not match the corresponding Request (e.g., a problem with Attributes, Traffic Class, Byte Count, Lower

Address, etc.), it is strongly recommended for the Receiver to handle the Completion as a Malformed TLP.

◦ The Completer must not check the IDO Attribute (Attribute Bit 2) in the Completion, since the

Requester is not required to copy the value of IDO from the Request into the Completion for that request as stated in [Section 2.2.6.4](#bookmark66)and [Section 2.2.9 .](#bookmark138)

◦ However, if the Completion is otherwise properly formed, it is permitted28 for the Receiver to handle the Completion as an Unexpected Completion.

• When an Ingress Port of a Switch receives a Completion that cannot be forwarded, that Ingress Port must handle the Completion as an Unexpected Completion. This includes Completions that target:

◦ a non-existent Function in the Device associated with the Upstream Port,

◦ a non-existent Device on the Bus associated with the Upstream Port,

◦ a non-existent Device or Function on the internal switching fabric, or

◦ a Bus Number within the Upstream Port's Bus Number aperture but not claimed by any Downstream Port.

• 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 (seeSection 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.

◦ Handle the error via a Requester-specific mechanism (see Section 6.2.3.2.5).

If the Completion arrives between the time an FLR has been initiated and the completion of the FLR by the targeted Function, the Completion is permitted to be handled as an Unexpected Completion or to be silently discarded (following update of flow control credits) without logging or signaling it as an error. Once the FLR has completed, received Completions corresponding to Requests issued prior to the FLR must be handled as Unexpected Completions, unless the Function has been re-enabled to issue Requests.

• 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 (seeSection 6.6). For Root Complexes

that support CRS Software Visibility, the following rules apply:

◦ If CRS Software Visibility is not enabled, the Root Complex must re-issue the Configuration Request as a new Request.

◦ If CRS Software Visibility is enabled (see below):



28. For the case where only the Byte Count or Lower Address fields mismatch the expected values for a Memory Read Request, it is actually recommended for the Receiver to handle the Completion as an Unexpected Completion, since the mismatch might be caused by a previous Completion being misrouted.

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

. For a Configuration Read Request that includes both bytes of the Vendor ID field of a device Function's Configuration Space Header, the Root Complex must complete the Request to the host by returning a read-data value of 0001h for the Vendor ID field and all ‘1’s for any additional bytes included in the request. This read-data value has been reserved

specifically for this use by the PCI-SIG and does not correspond to any assigned Vendor ID.

. For a Configuration Write Request or for any other Configuration Read Request, the Root Complex must re-issue the Configuration Request as a new Request.

A Root Complex implementation may choose to limit the number of Configuration Request/ CRS Completion Status loops before determining that something is wrong with the target of the Request and taking appropriate action, e.g., complete the Request to the host as a

failed transaction.

CRS Software Visibility may be enabled through the CRS Software Visibility Enable bit in the Root Control register (seeSection 7.5.3.12) to control Root Complex behavior on an

individual Root Port basis. Alternatively, Root Complex behavior may be managed through the CRS Software Visibility Enable bit in theRoot Complex Register Block (RCRB) Control register as described in Section 7.9.7.4 , permitting the behavior of one or more Root Ports or RCiEPs to be controlled by a single Enable bit. For this alternate case, each Root Port or RCiEP declares its association with a particular Enable bit via an RCRB header association in a Root Complex Link Declaration Capability (seeSection 7.9.8). Each Root Port or RCiEP is permitted to be controlled by at most one Enable bit. Thus, for example, it is prohibited for a Root Port whose Root Control register contains an Enable bit to declare an RCRB

header association to an RCRBthat also includes an Enable bit in its RCRB Header

Capability. The presence of an Enable bit in a Root Port or RCRB Header Capability is

indicated by the corresponding CRS Software Visibility bit (seeSection 7.5.3.13and Section 7.9.7.3 , respectively).

• Completions with a Configuration Request Retry Status in response to a Request other than a Configuration Request are illegal. Receivers may optionally report these violations as Malformed TLPs.

◦ This is a reported error associated with the Receiving Port (seeSection 6.2).

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

• Completions with a Completion Status of Unsupported Request or Completer Abort are reported using the conventional PCI reporting mechanisms (seeSection 7.5.1.1.4).

◦ Note that the error condition that triggered the generation of such a Completion is reported by the Completer as described in Section 6.2 .

• When a Read Completion or an AtomicOp 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)

◦ 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 had issued, then it receives a Completion with the Completer Abort Completion Status. The

Requester then must free the internal resources that had been allocated for that particular Read Request.

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

**IMPLEMENTATION NOTE**

Read Data Values with UR Completion Status

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.



**2.4 Transaction Ordering**

**2.4.1 Transaction Ordering Rules**

[Table 2-40](#bookmark166)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 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 2-40](#bookmark167), 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. (When blocking occurs, the second transaction is required to pass the first transaction. Fairness must be comprehended to prevent

starvation.)

**Y/N**

There are no requirements. The second transaction may optionally pass the first transaction or be blocked by it.

**No**

The second transaction must not be allowed to pass the first transaction. This is required to support the producer/ consumer strong ordering model.

Table 2-40 Ordering Rules Summary

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Row Pass Column? | | Posted Request  (Col 2) | Non-Posted Request | | Completion (Col 5) |
| Read Request  (Col 3) | NPR with Data  (Col 4) |
| Posted Request (Row A) | | [a) No](#bookmark169)  [b) Y/N](#bookmark170) | [Yes](#bookmark171) | [Yes](#bookmark172) | [a) Y/N](#bookmark173)  [b) Yes](#bookmark174) |
| Non-Posted Request | Read Request (Row B) | [a) No](#bookmark177)  [b) Y/N](#bookmark178) | [Y/N](#bookmark179) | [Y/N](#bookmark180) | [Y/N](#bookmark181) |
| NPR with Data (Row C) | [a) No](#bookmark183)  [b) Y/N](#bookmark184) | [Y/N](#bookmark185) | [Y/N](#bookmark186) | [Y/N](#bookmark187) |

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Row Pass Column? | Posted Request  (Col 2) | Non-Posted Request | | Completion (Col 5) |
| Read Request  (Col 3) | NPR with Data  (Col 4) |
| Completion (Row D) | [a) No](#bookmark189)  [b) Y/N](#bookmark190) | [Yes](#bookmark191) | [Yes](#bookmark192) | [a) Y/N](#bookmark193)  [b) No](#bookmark194) |

Explanation of the row and column headers in [Table 2-40 :](#bookmark167)

A **Posted Request** is a Memory Write Request or a Message Request.

A **Read Request** is a Configuration Read Request, an I/O Read Request, or a Memory Read Request.

An **NPR** (Non-Posted Request) **with Data** is a Configuration Write Request, an I/O Write Request, or an AtomicOp Request.

A **Non-Posted Request** is a Read Request or an NPR with Data. Explanation of the entries in [Table 2-40 :](#bookmark167)

[**A2a**](#bookmark168)[**A2b**](#bookmark168)

A Posted Request must not pass another Posted Request unless A2bapplies.

A Posted Request with RO29 Set is permitted to pass another Posted Request.30 A Posted Request with IDO Set is

permitted to pass another Posted Request if the two Requester IDs are different or if both Requests contain a PASID TLP Prefix and the two PASID values are different.

[**A3,A4**](#bookmark168)

A Posted Request must be able to pass Non-Posted Requests to avoid deadlocks.

[**A5a**](#bookmark168)

 A Posted Request is permitted to pass a Completion, but is not required to be able to pass Completions unless A5b applies.

[**A5b**](#bookmark168)

Inside a PCI Express to PCI/PCI-X Bridge whose PCI/PCI-X bus segment is operating in conventional PCI mode, for transactions traveling in the PCI Express to PCI direction, a Posted Request must be able to pass Completions to avoid deadlock.

[**B2a**](#bookmark176)

 A Read Request must not pass a Posted Request unless B2bapplies.

[**B2b**](#bookmark176)

A Read Request with IDO Set is permitted to pass a Posted Request if the two Requester IDs are different or if both Requests contain a PASIDTLP Prefix and the two PASID values are different.

[**C2a**](#bookmark175)

 An NPR with Data must not pass a Posted Request unless C2bapplies.



29. In this section, “RO” is an abbreviation for the[Relaxed Ordering](#bookmark66)Attribute field.

30. Some usages are enabled by not implementing this passing (see the No RO-enabled PR-PR Passing bit inSection 7.5.3.15).

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

[**C2b**](#bookmark182)

An NPR with Data and with RO Set31 is permitted to pass Posted Requests. An NPR with Data and with IDO Set is permitted to pass a Posted Request if the two Requester IDs are different or if both Requests contain a PASIDTLP Prefix and the two PASID values are different.

[**B3,B4,**](#bookmark176)[**C3,C4**](#bookmark182)

A Non-Posted Request is permitted to pass another Non-Posted Request.

[**B5,**](#bookmark176)[**C5**](#bookmark182)

A Non-Posted Request is permitted to pass a Completion.

[**D2a**](#bookmark188)

 A Completion must not pass a Posted Request unless D2bapplies.

[**D2b**](#bookmark188)

 An I/O or Configuration Write Completion32 is permitted to pass a Posted Request. A Completion with RO Set is permitted to pass a Posted Request. A Completion with IDO Set is permitted to pass a Posted Request if the

Completer ID of the Completion is different from the Requester ID of the Posted Request.

[**D3,D4**](#bookmark188)

A Completion must be able to pass Non-Posted Requests to avoid deadlocks.

[**D5a**](#bookmark188)

 Completions with different Transaction IDs are permitted to pass each other.

[**D5b**](#bookmark188)

 Completions with the same Transaction ID must not pass each other. This ensures that multiple Completions

associated with a single Memory Read Request will remain in ascending address order. Additional Rules:

• PCI Express Switches are permitted to allow a Memory Write or Message Request with the[Relaxed Ordering](#bookmark66) bit set to pass any previously posted Memory Write or Message Request moving in the same direction. Switches must forward the [Relaxed Ordering](#bookmark66)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]) 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.

• The [No Snoop](#bookmark69) bit does not affect the required ordering behavior.

• For Root Ports and Switch Downstream Ports, acceptance of a Posted Request or Completion must not depend upon the transmission of a Non-Posted Request within the same traffic class.33

• For Switch Upstream Ports, acceptance of a Posted Request or Completion must not depend upon the transmission on a Downstream Port of Non-Posted Request within the same traffic class.34



31. Note: Not all NPR with Data transactions are permitted to have RO Set.

32. Note: Not all components can distinguish I/O and Configuration Write Completions from other Completions. In particular, routing elements not serving as the associated Requester or Completer generally cannot make this distinction. A component must not apply this rule for I/O and Configuration Write Completions unless it is certain of the associated Request type.

33. Satisfying the above rules is a necessary, but not sufficient condition to ensure deadlock free operation. Deadlock free operation is dependent upon the system topology, the number of Virtual Channels supported and the configured Traffic Class to Virtual Channel mappings. Specification of platform and system constraints to ensure deadlock free operation is outside the scope of this specification (see Appendix Dfor a discussion of relevant issues).

34. Satisfying the above rules is a necessary, but not sufficient condition to ensure deadlock free operation. Deadlock free operation is dependent upon the system topology, the number of Virtual Channels supported and the configured Traffic Class to Virtual Channel mappings. Specification of platform and system constraints to ensure deadlock free operation is outside the scope of this specification (see Appendix Dfor a discussion of relevant issues).

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

• For Endpoint, Bridge, and Switch Upstream Ports, the acceptance of a Posted Request must not depend upon the transmission of any TLP from that same Upstream Port within the same traffic class.35

• For Endpoint, Bridge, and Switch Upstream Ports, the acceptance of a Non-posted Request must not depend upon the transmission of a Non-Posted Request from that same Upstream Port within the same traffic class.36

• For Endpoint, Bridge, and Switch Upstream Ports, the acceptance of a Completion must not depend upon the transmission of any TLP from that same Upstream Port within the same traffic class.37

Note that Endpoints are never permitted to block acceptance of a Completion.

• Completions issued for Non-Posted requests must be returned in the same Traffic Class as the corresponding Non-Posted request.

• Root Complexes that support peer-to-peer operation and Switches must enforce these transaction ordering rules for all forwarded traffic.

To ensure deadlock-free operation, devices should not forward traffic from one Virtual Channel to another. The

specification of constraints used to avoid deadlock in systems where devices forward or translate transactions between Virtual Channels is outside the scope of this document (seeAppendix Dfor a discussion of relevant issues).



35. Satisfying the above rules is a necessary, but not sufficient condition to ensure deadlock free operation. Deadlock free operation is dependent upon the system topology, the number of Virtual Channels supported and the configured Traffic Class to Virtual Channel mappings. Specification of platform and system constraints to ensure deadlock free operation is outside the scope of this specification (see Appendix Dfor a discussion of relevant issues).

36. Satisfying the above rules is a necessary, but not sufficient condition to ensure deadlock free operation. Deadlock free operation is dependent upon the system topology, the number of Virtual Channels supported and the configured Traffic Class to Virtual Channel mappings. Specification of platform and system constraints to ensure deadlock free operation is outside the scope of this specification (see Appendix Dfor a discussion of relevant issues).

37. Satisfying the above rules is a necessary, but not sufficient condition to ensure deadlock free operation. Deadlock free operation is dependent upon the system topology, the number of Virtual Channels supported and the configured Traffic Class to Virtual Channel mappings. Specification of platform and system constraints to ensure deadlock free operation is outside the scope of this specification (see Appendix Dfor a discussion of relevant issues).

Page 180

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

**IMPLEMENTATION NOTE**

Large Memory Reads vs. Multiple Smaller Memory Reads

Note that the rule associated with entry D5b in [Table 2-40](#bookmark167)ensures that for a single Memory Read Request

serviced with multiple Completions, the Completions will be returned in address order. 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](#bookmark144)) of 128 bytes each, it is guaranteed that the two Completions will return in the following order:

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. 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 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 Bvia 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 QW to host memory, a Requester reading that QW from host memory may observe a portion of the QW 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 DWs or aligned QWs to host memory, the update granularity observed by a PCI Express read will not be

smaller than a DW.

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

**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 DWs and the[Relaxed Ordering](#bookmark66) 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 DW.

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

**2.5 Virtual Channel (VC) Mechanism**

The Virtual Channel (VC) mechanism provides support for carrying, throughout the fabric, traffic that is differentiated

using TC labels. The foundations of VCs are independent fabric resources (queues/buffers and associated control logic). These resources are used to move information across 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 VC and

Multi-Function Virtual Channel (MFVC) mechanisms allow 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](#bookmark195)covers details of TC to VC mapping.

A Virtual Channel is established when one or multiple TCs are associated with a physical VC resource designated by Virtual Channel Identification (VC ID). This process is controlled by configuration software as described in Section 6.3 , Section 7.9.1 , and Section 7.9.2 .

Support for TCs and VCs beyond the default TC0/VC0 pair is optional. The association of TC0 with VC0 is fixed,i.e.,

“hardwired”, and must be supported by all components. Therefore the baselineTC/VC setup does not require any

VC-specific hardware or software configuration. In order to ensure interoperability, components that do not implement the optional Virtual Channel Capability structure or Multi-Function 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.)

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

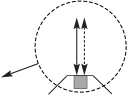
• A Completer must accept requests with TC label other than TC0, and must preserve the TC label. That is, any completion that it generates must have the same TC label as the label of the request.

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

A Device containing Functions capable of generating Requests with TC labels other than TC0 must implement suitable VC or MFVC Capability structures (as applicable), even if it only supports the default VC. Example Function types are

Endpoints and Root Ports. 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 and MFVC Capability structures.

[Figure 2-45](#bookmark196)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.



|  |  |  |
| --- | --- | --- |
| Root Complex |  | |
|  |  |

Packets

VC0

VCn

VC0

VCn

single Link

|  |  |
| --- | --- |
|  |  |



VCn

VCn

|  |
| --- |
| Component B |

|  |
| --- |
| Component E |

|  |
| --- |
| Component A |

|  |
| --- |
| Component C |

|  |
| --- |
| Component D |



with 2 VC's



《 Link



Packets

Switch



VC0

VC0

 Default VC (VC0)

 Another VC

OM13760

Figure 2-45 Virtual Channel Concept - An Illustration

Internal to the Switch, every Virtual Channel requires dedicated phys [Figure 2-46](#bookmark197)shows conceptually the VC resources within the Switch (shown in [Figure 2-45)](#bookmark196) that are required to support traffic flow in the Upstream direction.

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

Downstream Ports

Switch



VC #0



VC #0



VC #0







VC #1

VC #1

Upstream Port

OM13761

Figure 2-46 Virtual Channel Concept - Switch Internals (Upstream Flow)

An MFD may implement Virtual Channel resources similar to a subset of those in a Switch, for the purpose of managing the Quality of Service (QoS) for Upstream requests from the different Functions to the device's Upstream Egress Port.

**IMPLEMENTATION NOTE**

VC and VC Buffering Considerations

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

Buffering beyond the architectural minimums is not required to be identical across all VCs on a given Link. That is, 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.

Implementations may adjust their buffering per VC based on implementation-specific policies derived from configuration and VC enablement. For example, 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.

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)**

PCI Express Ports can support 1 to 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 VC ID mechanism.

Note that while DLLPs contain VC ID information for Flow Control accounting,TLPs do not. 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 .](#bookmark198)

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

All Ports that support more than VC0 must provide at least one VC Capability structure according to the definition in Section 7.9.1 . An MFD is permitted to implement the MFVC Capability structure, as defined in Section 7.9.2 . Providing these extended structures is optional for Ports that support only the default TC0/VC0 configuration. 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 hierarchy and using VC or MFVC Capability registers associated with Ports (that support more than default VC0) to establish the number of VCs for the Link. Rules for assigning VC ID to VC hardware resources within a Port are as follows:

• VC ID assignment must be unique per 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 Ports on both sides of a Link.

• If an MFD implements an MFVC Capability structure, its VC hardware resources are distinct from the VC

hardware resources associated with any VC Capability structures of its Functions. The VC ID uniqueness

requirement (first bullet above) still applies individually for the MFVC and any VC Capability structures. In addition, the VC ID cross-Link matching requirement (second bullet above) applies for the MFVC Capability

structure, but not the VC Capability structures of the Functions.

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

**2.5.2 TC to VC Mapping**

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.

• OneTC must not be mapped to multiple VCs in any Port or Endpoint Function.

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

[Table 2-41](#bookmark199)provides an example of TC to VC mapping.

Table 2-41 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**  **TC(n-m)/VCk**  **TC[n:m]/VC[n:m]** | TCn mapped to VCk  all TCs in the range n-m mapped to VCk (i.e., to the same VC)  TCn/VCn, TCn+1/ VCn+1, ..., TCm/VCm | |

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

[Figure 2-47](#bookmark200)provides a graphical illustration of TC to VC mapping in several different Link configurations. For additional considerations on TC/VC, refer toSection 6.3 .

Link

Switch

|  |
| --- |
| Endpoint  TC[0:7] |

Link



VC0

TC[0:6] TC7

VC0



TC[0:7]

VC1

Mapping

Link

|  |
| --- |
| Endpoint  TC[0:1]  TC[2:4]  TC[5:6]  TC7 |



VC0

VC1

VC2

VC3



Mapping

Switch

TC[0:1]

—、TC[2:4]

Link

TC[5:6]

TC7



VC0

TC[0:1]



VC1

TC[2:4]

Link

|  |
| --- |
| Endpoint  TC[0:1]  TC[2:4]  TC[5:6]  TC7 |



VC2

TC[5:6]



VC0

VC1

VC2

VC3

TC[0:1]

VC3

TC7

TC[2:4]

|  |
| --- |
| TC[0:6]  TC7  Root  Complex  TC[0:1]  TC[2:4]  TC[5:6]  TC7 |

TC[5:6]



TC7

OM13762

Figure 2-47 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 devices must support the general purpose I/O Traffic Class,i.e., TC0 and must implement the default VC0.

• 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.

• An MFD's peer-to-peer capability between different Functions applies to all Virtual Channels supported by the MFD.

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

• Transactions with a TC that is not mapped to any enabled VC in an Ingress Port are treated as Malformed TLPs by the receiving device.

• For Switches, transactions with a TC that is not mapped to any of the 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 the enabled VCs in the targetRCRBare treated as Malformed TLPs.

• For MFDs with an MFVC Capability structure, any transaction with a TC that is not mapped to an enabled VC in the MFVC Capability structure is treated as a Malformed TLP.

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

• A Root Complex must support independent TC/VC mapping configuration for each RCRB, the associated Root Ports, and any RCiEPs.

For more details on the VC and TC mechanisms, including configuration, mapping, and arbitration, refer toSection 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 .](#bookmark68) 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 2-48](#bookmark201). 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.

|  |
| --- |
| Intermediate Component(s) |

|  |
| --- |
| Ultimate Completer |

Link

Link

|  |
| --- |
| Requester |

OM13776

Figure 2-48 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 3.6).

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

Control credit accounting.

Flow Control mechanisms used internally within an MFD are outside the scope of this specification.

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 even if those TLPs are eventually nullified..

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,

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

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 device could use in order to implement a compliant implementation. 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 3.5).

• The unit of Flow Control credit is 4 DW for data.

• For headers:

◦ The unit of Flow Control credit for Receivers that do not support TLP Prefixes is the sum of one maximum-size Header and TLP Digest.

◦ The unit of Flow Control credits for Receivers that support End-End TLP Prefixes is the sum of one

maximum-size Header, TLP Digest, and the maximum number of End-End TLP Prefixes permitted in a TLP.

◦ The management of Flow Control for Receivers that support Local TLP Prefixes is dependent on the Local TLP Prefix type.

• Each Virtual Channel has independent Flow Control.

• Flow Control distinguishes three typesofTLPs (note relationship to ordering rules - see [Section 2.4](#bookmark68)):

◦ Posted Requests (P) - Messages and Memory Writes

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

◦ Completions (Cpl) - Associated with corresponding NP Requests

• In addition, Flow Control distinguishes the following typesofTLP 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](#bookmark202) [2-42 .](#bookmark202)

Table 2-42 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 |

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

• TLPs consume Flow Control credits as shown in [Table 2-43 .](#bookmark203)

Table 2-43 TLP Flow Control Credit Consumption

|  |  |
| --- | --- |
| TLP | Credit Consumed38 |
| Memory, I/O, Configuration Read Request | 1 NPH unit |
| Memory Write Request | 1 PH + n PD units39 |
| I/O, Configuration Write Request | 1 NPH + 1 NPD  Note: size of data written is never more than 1 (aligned) DW |
| AtomicOp Request | 1 NPH + n NPD units |
| 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 |
| AtomicOp Completion | 1 CplH unit + 1 CplD unit  Note: size of data returned is never more than 4 (aligned) DWs. |

• 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 (seeSection 3.2and Section 3.4).

• When other Virtual Channels are enabled by software, each newly enabled VC will follow the Flow Control initialization protocol (see Section 3.4).

◦ Software enables a Virtual Channel by setting the VC Enable bits for that Virtual Channel in both components on a Link (see Section 7.9.1and Section 7.9.2).

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 3.4).

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



38. Each header credit implies the ability to accept a TLP Digest along with the corresponding TLP.

39. For all cases where “n” appears, n = Roundup(Length/FC unit size).

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

• 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](#bookmark204) [2-44 .](#bookmark205)

◦ If Scaled Flow Control is not supported or supported but not activated, use the values in the "Scale Factor 1" column.

• If Scaled Flow Control is supported and activated, use the values in the column for the scaling factor associated with that credit type (see Section 3.4.2).

Table 2-44 Minimum Initial Flow Control Advertisements40

|  |  |  |  |
| --- | --- | --- | --- |
| Credit Type | Minimum Advertisement | | |
| No Scaling or Scale Factor 1 | Scale Factor 4 | Scale Factor 16 |
| PH | 1 unit - credit value of 01h. | 4 Units - credit value of 01h. | 16 Units - credit value of 01h. |
| PD | Largest possible setting of the  Max\_Payload\_Size for the component divided by FC Unit Size. For an MFD, this includes all Functions in the device.  Example: If the largest Max\_Payload\_Size  value supported is 1024 bytes, the smallest permitted initial credit value would be 040h. | Ceiling(Largest Max\_Payload\_Size / (FC Unit Size \* 4)) + 1. For an MFD, this includes all Functions in the  device.  Example: If the largest  Max\_Payload\_Size value supported  is 1024 bytes, the smallest  permitted initial credit value would be 011h. | Ceiling(Largest Max\_Payload\_Size / (FC Unit Size \* 16)) + 1. For an MFD, this includes all Functions in the  device.  Example: If the largest  Max\_Payload\_Size value supported  is 1024 bytes, the smallest  permitted initial credit value would be 005h. |
| NPH | 1 unit - credit value of 01h. | 4 Units - credit value of 01h. | 16 Units - credit value of 01h. |
| NPD | Receiver that supports AtomicOp routing capability or any AtomicOp Completer capability: 2 units - credit value of 002h  All other Receivers: 1 unit - credit value of 001h. | Receiver that supports AtomicOp routing capability or any AtomicOp Completer capability: 8 units -  credit value of 002h  All other Receivers: 4 units - credit value of 001h. | Receiver that supports AtomicOp routing capability or any AtomicOp Completer capability: 32 units -  credit value of 002h  All other Receivers: 16 units - credit value of 001h. |
| 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.41 | Root Complex (supporting  peer-to-peer traffic between all  Root Ports) and Switch: 4 FC units - 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.42 | Root Complex (supporting  peer-to-peer traffic between all  Root Ports) and Switch: 16 FC units - 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.43 |
| 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. | Root Complex (supporting  peer-to-peer traffic between all Root Ports) and Switch:  Ceiling(Largest Max\_Payload\_Size / (FC Unit Size \* 4)) + 1. | Root Complex (supporting  peer-to-peer traffic between all Root Ports) and Switch:  Ceiling(Largest Max\_Payload\_Size / (FC Unit Size \* 16)) + 1. |

|  |
| --- |
| 40. PCI Express to PCI/PCI-X Bridge requirements are addressed in [PCIe-to-PCI-PCI-X-Bridge-1.0]. |
| 41. This value is interpreted as infinite by the Transmitter, which will, therefore, never throttle. |
| 42. This value is interpreted as infinite by the Transmitter, which will, therefore, never throttle. |
| 43. This value is interpreted as infinite by the Transmitter, which will, therefore, never throttle. |

Page 190

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

|  |  |  |  |
| --- | --- | --- | --- |
| Credit Type | Minimum Advertisement | | |
| No Scaling or Scale Factor 1 | Scale Factor 4 | Scale Factor 16 |
|  | Root Complex (not supporting peer-to-peer | Root Complex (not supporting | Root Complex (not supporting |
|  | traffic between all Root Ports) and Endpoint: | peer-to-peer traffic between all | peer-to-peer traffic between all |
|  | infinite FC units - initial credit value of all 0s. | Root Ports) and Endpoint: infinite | Root Ports) and Endpoint: infinite |
|  |  | FC units - initial credit value of all | FC units - initial credit value of all |
|  |  | 0s. | 0s. |

• A Root Complex that supports no peer-to-peer traffic between Root Ports must advertise infinite Completion credits on every Root Port.

• A Root Complex that supports peer-to-peer traffic between some or all of its Root Ports may optionally

advertise non-infinite Completion credits on those Root Ports. 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.

• A Receiver that does not support Scaled Flow Control must never cumulatively issue more than 2047

outstanding unused credits to the Transmitter for data payload or 127 for header. A Receiver that supports Scaled Flow Control must never cumulatively issue more outstanding unused data or header to the

Transmitter than the Max Credits values shown in Table 3-2 .

。 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 (seeSection 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 Clear 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 (seeSection 6.2)

• If only the Data or header advertisement (but not both) for a given type (P, 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 (seeSection 6.2).

• If Scaled Flow Control is activated, the HdrScale and DataScale fields in the UpdateFCs must match the values advertised during initialization (see Section 3.4.2).

。 The Receiver may optionally check for violations 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 (seeSection 6.2).

• A received TLP using a VC that is not enabled is a Malformed TLP. 。 VC0 is always enabled.

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

。 For VCs 1-7, a VC is considered enabled when the corresponding VC Enable bit in the VC Resource Control register has been Set, and once FC negotiation for that VC has exited the FC\_INIT1 state and progressed to the FC\_INIT2 state (seeSection 3.4).

。 This is a reported error associated with the Receiving Port (seeSection 6.2).

• TLP transmission using any VC 0-7 is not permitted until initialization for that VC has completed by exiting FC\_INIT2 state.

For VCs 1-7, software must use the VC Negotiation Pending bit in the VC Resource Status register to ensure that a VC is not used until negotiation has completed by exiting the FC\_INIT2 state in both components on a Link.

The **[Field Size]** parameter used in the following sections is described in [Table 2-45](#bookmark207)(see Section 3.4.2).

Table 2-45 [Field Size] Values

|  |  |  |  |
| --- | --- | --- | --- |
| Scaled Flow  Control Supported | HdrScale or DataScale | [[Field Size]](#bookmark206)  for PH, NPH, CplH | [[Field Size]](#bookmark206)  for PD, NPD, CplD |
| No | x | 8 | 12 |
| Yes | 00b | 8 | 12 |
| Yes | 01b | 8 | 12 |
| Yes | 10b | 10 | 14 |
| Yes | 11b | 12 | 16 |

[**2.6.1.1**](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]](#bookmark206) (where[[Field Size]](#bookmark206) is defined in [Table 2-45)](#bookmark207).

▪ Set to all 0's at interface initialization

▪ Updated for each TLP the Transaction Layer allows to pass the Flow Control gate for Transmission as shown:

[CREDITS\_CONSUMED](#bookmark208):= ([CREDITS\_CONSUMED](#bookmark208)+ Increment) mod 2[[Field Size]](#bookmark206)

Equation 2-1 [CREDITS\_CONSUMED](#bookmark208)

(Where Increment is the size in FC credits of the corresponding part of the TLP passed through the gate, and[[Field Size]](#bookmark206)is defined in [Table 2-45)](#bookmark207)

。 **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]](#bookmark206) (where[[Field Size]](#bookmark206) is defined in [Table 2-45)](#bookmark207).

▪ Undefined at interface initialization

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

. Set to the value indicated during Flow Control initialization . For each FC update received,

. if[CREDIT\_LIMIT](#bookmark209)is not equal to the update value, set[CREDIT\_LIMIT](#bookmark209)to the update value

• If a Transmitter detects that a TLP it is preparing to transmit is malformed, it is strongly recommended that the Transmitter discard the TLP and handle the condition as an Uncorrectable Internal Error.

• If a Transmitter detects that a TLP it is preparing to transmit appears to be properly formed but with badECRC, it is strongly recommended that the Transmitter transmit the TLP and update its internal Flow Control credits

accordingly.

• 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 ,](#bookmark68) which require that certain typesofTLPs must bypass other specific typesofTLPs 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](#bookmark208)+ credit units required for pending TLP) mod 2[[Field Size]](#bookmark206)

Equation 2-2 [CUMULATIVE\_CREDITS\_REQUIRED](#bookmark210)

◦ Unless [CREDIT\_LIMIT](#bookmark209)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](#bookmark209) -[CUMULATIVE\_CREDITS\_REQUIRED)](#bookmark210) mod 2[[Field Size]](#bookmark206) ≤ 2[[Field Size]](#bookmark206)/2

Equation 2-3 Transmitter Gate

◦ If[CREDIT\_LIMIT](#bookmark209)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](#bookmark68)when determining what typesofTLPs 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.

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

• When a Transmitter sends a nullified TLP, the Transmitter does not modify[CREDITS\_CONSUMED](#bookmark208)for that TLP (see Section 3.6.2.1).

[**2.6.1.2**](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]](#bookmark206) (where[[Field Size]](#bookmark206) is defined in [Table 2-45)](#bookmark207)

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

▪ This value is included in the InitFC and UpdateFC DLLPs (seeSection 3.5)

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

Updated as shown:

[CREDITS\_ALLOCATED](#bookmark211):= ([CREDITS\_ALLOCATED](#bookmark211)+ Increment) mod 2[Field Size]

Equation 2-4 [CREDITS\_ALLOCATED](#bookmark211)

(Where Increment corresponds to the credits made available, and[[Field Size]](#bookmark206)is defined in [Table 2-45)](#bookmark207)

。 **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]](#bookmark206) (where[[Field Size]](#bookmark206) is defined in [Table 2-45)](#bookmark207)

▪ Set to all 0's at interface initialization

▪ Updated as shown:

[CREDITS\_RECEIVED](#bookmark212):= ([CREDITS\_RECEIVED](#bookmark212)+ Increment) mod 2[[Field Size]](#bookmark206)

Equation 2-5 [CREDITS\_RECEIVED](#bookmark212)

(Where Increment is the size in FC units of the corresponding part of the received TLP, and

[[Field Size]](#bookmark206) is defined in [Table 2-45)](#bookmark207)

for each Received TLP, provided that TLP:

▪ passes the Data Link Layer integrity checks

▪ is not malformed or (optionally) is malformed and is not ambiguous with respect to which buffer to release and is mapped to an initialized Virtual Channel

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

▪ For a TLP with an ECRC Check Failed error, but which otherwise is unambiguous with respect to which buffer to release, it is strongly recommended that

[CREDITS\_RECEIVED](#bookmark212)be updated.

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

• If a Receiver implements the [CREDITS\_RECEIVED](#bookmark212)counter, then when a nullified TLP is received, the Receiver does not modify[CREDITS\_RECEIVED](#bookmark212)for that TLP (see Section 3.6.2.1)

• A Receiver may optionally check for Receiver Overflow errors (TLPs exceeding [CREDITS\_ALLOCATED](#bookmark211)), by checking the following equation, using unsigned arithmetic:

[(CREDITS\_ALLOCATED](#bookmark211) -[CREDITS\_RECEIVED)](#bookmark212) mod 2[[Field Size]](#bookmark206) ≥ 2s[[Field Size]](#bookmark206)/2

Equation 2-6 Receiver Overflow Error Check

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

。 discard the TLP(s) without modifying the [CREDITS\_RECEIVED](#bookmark212) 。 de-allocate any resources that it had allocated for the TLP(s)

If checked, this is a reported error associated with the Receiving Port (seeSection 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 that 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 events occur:

a. when scaled flow control is not activated and the number of available FC credits of a particular type is zero and one or more units of that type are made available by TLPs processed,

b. when scaled flow control is not activated, the NPD credit drops below 2, the Receiver supports either the AtomicOp routing capability or the 128-bit CAS Completer capability, and one or more NPD

credits are made available by TLPs processed,

c. when scaled flow control is activated and the number of available FC credits of a particular type is zero or is below the scaled threshold and one or more units of that type are made available by TLPs processed so that the number of available credits is equal to or greater than the scaled threshold, which is 0 for HdrScale or Data Scale of 01b, 4 for HdrScale or DataScale of 10b, and 16 for HdrScale or DataScale of 11b.

d. when scaled flow control is activated, the DataScale used for NPD is 01b, the NPD credit drops below 2, the Receiver supports either the AtomicOp routing capability or the 128-bit CAS Completer

capability, and one or more NPD credits are made available by TLPs processed.

• For non-infinite 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

。 For ARI Devices, the Max\_Payload\_Size is determined solely by the setting in Function 0. The Max\_Payload\_Size settings in other Functions are ignored.

。 For a non-ARI MFD whose Max\_Payload\_Size settings are identical across all Functions, the common Max\_Payload\_Size setting or larger must be used.

。 For a non-ARI MFD whose Max\_Payload\_Size settings are not identical across all Functions, the selected Max\_Payload\_Size setting is implementation specific, but it is recommended to use the largest Max\_Payload\_Size setting across all Functions.

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

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

• 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 Synch bit of the

Link Control 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 3.5)

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

▪ if an Infinite Credit advertisement has been made during initialization for all three Flow Control classes, this timeout mechanism must be disabled

Note: The implementation of this optional mechanism is strongly encouraged. 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, in a Device whose Functions have no AtomicOp Completer or

AtomicOp Routing capability, there is no need to implement a Non-Posted Data queue for Virtual Channels other than VC0, since Non-Posted Requests with data are only allowed on Virtual Channel 0 for such Devices. 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.



5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0



**IMPLEMENTATION NOTE**

Flow Control Update Latency

For components subject to receiving streams ofTLPs, it is desirable to implement receive buffers larger than the minimum size required to prevent Transmitter throttling due to lack of available credits. Likewise, it is desirable to transmit UpdateFC FCPs such that the time required to send, receive and process the UpdateFC prevents

Transmitter throttling. Recommended maximum values for UpdateFC transmission latency during normal

operation are shown in [Table 2-46 ,](#bookmark213)[Table 2-47 ,](#bookmark214) and [Table 2-48 .](#bookmark215) Note that the values given in in these tables do not account for any delays caused by the Receiver or Transmitter being in L0s, in Recovery, or for any delays

caused by Retimers (see Section 4.3.8) . 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 in the Tables are measured starting from when the Receiver Transaction Layer makes additional receive buffer space available by processing a received TLP, to when the first Symbol of the corresponding UpdateFC DLLP is transmitted.

Table 2-46 Maximum UpdateFC Transmission Latency Guidelines for

2.5 GT/s (Symbol Times)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | Link Operating Width | | | | | | |
| x1 | x2 | x4 | x8 | x12 | x16 | x32 |
| Max\_Payload\_Size  (bytes) | 128 | 237 | 128 | 73 | 67 | 58 | 48 | 33 |
| 256 | 416 | 217 | 118 | 107 | 90 | 72 | 45 |
| 512 | 559 | 289 | 154 | 86 | 109 | 86 | 52 |
| 1024 | 1071 | 545 | 282 | 150 | 194 | 150 | 84 |
| 2048 | 2095 | 1057 | 538 | 278 | 365 | 278 | 148 |
| 4096 | 4143 | 2081 | 1050 | 534 | 706 | 534 | 276 |

Table 2-47 Maximum UpdateFC Transmission Latency Guidelines for

5.0 GT/s (Symbol Times)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | Link Operating Width | | | | | | |
| x1 | x2 | x4 | x8 | x12 | x16 | x32 |
| Max\_Payload\_Size  (bytes) | 128 | 288 | 179 | 124 | 118 | 109 | 99 | 84 |
| 256 | 467 | 268 | 169 | 158 | 141 | 123 | 96 |
| 512 | 610 | 340 | 205 | 137 | 160 | 137 | 103 |
| 1024 | 1122 | 596 | 333 | 201 | 245 | 201 | 135 |
| 2048 | 2146 | 1108 | 589 | 329 | 416 | 329 | 199 |
| 4096 | 4194 | 2132 | 1101 | 585 | 757 | 585 | 327 |

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

Table 2-48 Maximum UpdateFC Transmission Latency Guidelines for

8.0 GT/s and Higher Data Rates (Symbol Times)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | Link Operating Width | | | | | | |
| x1 | x2 | x4 | x8 | x12 | x16 | x32 |
| Max\_Payload\_Size  (bytes) | 128 | 333 | 224 | 169 | 163 | 154 | 144 | 129 |
| 256 | 512 | 313 | 214 | 203 | 186 | 168 | 141 |
| 512 | 655 | 385 | 250 | 182 | 205 | 182 | 148 |
| 1024 | 1167 | 641 | 378 | 246 | 290 | 246 | 180 |
| 2048 | 2191 | 1153 | 634 | 374 | 461 | 374 | 244 |
| 4096 | 4239 | 2177 | 1146 | 630 | 802 | 630 | 372 |



**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. ATLP 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 (if supported) by the ultimate PCI Express

Receiver and optionally by intermediate Receivers. A Switch that supports ECRC checking must check ECRC on TLPs

targeting the Switch itself. Such a Switch can optionally check ECRC on TLPs that it forwards. On TLPs that the Switch

forwards, the Switch must preserve the ECRC (forward it untouched) as an integral part of the TLP, regardless of whether the Switch checks the ECRC or if the ECRC check fails.44

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 7.8.4.7).

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

• Switches must pass TLPs with ECRC unchanged from the Ingress Port to the Egress Port45



44. An exception is a Multicast TLP that an Egress Port is modifying due to theMC\_Overlay mechanism. SeeSection 6.14.5 .

45. An exception is a Multicast TLP that an Egress Port is modifying due to theMC\_Overlay mechanism. SeeSection 6.14.5 .

Page 198

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

• If a device supports ECRC generation/checking, at least one of its Functions must support Advanced Error Reporting (AER) (see Section 6.2)

• If a device Function is enabled to check ECRC, it must do so for all TLPs with ECRC where the device is the ultimate PCI Express Receiver

。 Note that it is still possible for the Function to receive TLPs without ECRC, and these are processed normally -this is not an error

Note that a Switch may optionally perform ECRC checking on TLPs passing through the Switch. ECRC Errors detected by the Switch are reported as described in Table 6-5, but do not alter the TLPs' passage through the Switch.46

A 32-bit ECRC is calculated for the TLP (End-End TLP Prefixes, header, and data payload) using the following algorithm and appended to the end of the TLP (see[Figure 2-3](#bookmark5)):

• The ECRC value is calculated using the following algorithm (see Figure 2-49)

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

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

• All header fields, all End-End TLP Prefixes (if present), and the entire data payload (if present) are included in the ECRC calculation. All bits inVariant fields must be Set for ECRC calculations.

。 Bit 0 of the Type field in a TLP header is Variant47 . This bit in an End-End TLP Prefix is invariant. 。 The EP bit is Variant

。 All other fields are Invariant

• 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 2-49 .](#bookmark216)

Table 2-49 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 |
| 9 | 14 |
| 10 | 13 |
| 11 | 12 |



46. An exception is a Multicast TLP that an Egress Port is modifying due to theMC\_Overlay mechanism. SeeSection 6.14.5 .

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

5.0-1.0-PUB — PCI Express® Base Specification Revision 5.0 Version 1.0

|  |  |
| --- | --- |
| ECRC Result Bit | Corresponding Bit Position in the 32-bit TLP Digest Field |
| 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)](#bookmark5)

• For TLPs including a TLP Digest field used for an ECRC value, Receivers that 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, and then

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

• Receivers that support end-to-end data integrity checks report violations as an ECRC Error. This reported error is associated with the Receiving Port (seeSection 6.2).

Beyond the stated error reporting semantics contained elsewhere in this specification, how ultimate PCI Express

Receivers make use of the end-to-end data integrity check provided through the ECRC is beyond the scope of this

document. Intermediate Receivers are still required to forward TLPs whose ECRC checks fail. A PCI Express-to-PCI/PCI-X Bridge is classified as an ultimate PCI Express Receiver with regard to ECRC checking.