PCI Express®
Base Specification
Revision 4.0 Version 1.0

September 27, 2017

Revision	Revision History	Date
1.0	Initial release.	07/22/2002
1.0a	Incorporated Errata C1-C66 and E1-E4.17.	04/15/2003
1.1	Incorporated approved Errata and ECNs.	03/28/2005
2.0	Added 5.0 GT/s data rate and incorporated approved Errata and ECNs.	12/20/2006
2.1	Incorporated *Errata for the PCI Express Base Specification, Rev. 2.0* (February 27, 2009), and added the following ECNs: • Internal Error Reporting ECN (April 24, 2008) • Multicast ECN (December 14, 2007, approved by PWG May 8, 2008) • Atomic Operations ECN (January 15, 2008, approved by PWG April 17, 2008) • Resizable BAR Capability ECN (January 22, 2008, updated and approved by PWG April 24, 2008) • Dynamic Power Allocation ECN (May 24, 2008) • ID-Based Ordering ECN (January 16, 2008, updated 29 May 2008) • Latency Tolerance Reporting ECN (22 January 2008, updated 14 August 2008) • Alternative Routing-ID Interpretation (ARI) ECN (August 7, 2006, last updated June 4, 2007) • Extended Tag Enable Default ECN (September 5, 2008) • TLP Processing Hints ECN (September 11, 2008) • TLP Prefix ECN (December 15, 2008)	03/04/2009
3.0	Added 8.0 GT/s data rate, latest approved Errata, and the following ECNs: • Optimized Buffer Flush/Fill ECN (8 February 2008, updated 30 April 2009) • ASPM Optionality ECN (June 19, 2009, approved by the PWG August 20, 2009) • Incorporated End-End TLP Changes for RCs ECN (26 May 2010) and Protocol Multiplexing ECN (17 June 2010)	11/10/2010
3.1	Incorporated feedback from Member Review Incorporated Errata for the PCI Express® Base Specification Revision 3.0 Incorporated M-PCIe Errata (3p1_active_errata_list_mpcie_28Aug2014.doc and 3p1_active_errata_list_mpcie_part2_11Sept2014.doc) Incorporated the following ECNs: • ECN: Downstream Port containment (DPC) • ECN: Separate Refclk Independent SSC (SRIS) Architecture • ECN: Process Address Space ID (PASID) • ECN: Lightweight Notification (LN) Protocol • ECN: Precision Time Measurement • ECN: Enhanced DPC (eDPC) • ECN: 8.0 GT/s Receiver Impedance • ECN: L1 PM Substates with CLKREQ • ECN: Change Root Complex Event Collector Class Code • ECN: M-PCIe • ECN: Readiness Notifications (RN) • ECN: Separate Refclk Independent SSC Architecture (SRIS) JTOL and SSC Profile Requirements	10/8/2014
3.1a	Minor update: Corrected: Equation 4.3.9 in Section 4.3.8.5., Separate Refclk With Independent SSC (SRIS) Architecture. Added missing square (exponent=2) in the definition of B. B = 2.2 × 10^12 × (2.π)^2 where ^= exponent.	12/5/2015
4.0	Version 0.3: Based on PCI Express® Base Specification Revision 3.1 (October 8, 2014) with some editorial feedback received in December 2013. • Added Chapter 9, Electrical Sub-block: Added Chapter 9 (Rev0.3-11-30-13_final.docx) • Changes related to Revision 0.3 release • Incorporated PCIe-relevant material from PCI Bus Power Management Interface Specification (Revision 1.2, dated March 3, 2004). This initial integration of the material will be updated as necessary and will supercede the standalone Power Management Interface specification. Version 0.5 (12/22/14, minor revisions on 1/26/15, minor corrections 2/6/15) • Added front matter with notes on expected discussions and changes. • Added ECN:Retimer (dated October 6, 2014) • Corrected Chapter 4 title to, “Physical Layer Logical Block”. • Added Encoding subteam feedback on Chapter 4 • Added Electrical work group changes from PCIe Electrical Specification Rev 0.5 RC1 into Chapter 9 Version 0.7: Based on PCI Express® Base Specification Version 4.0 Revision 0.5 (11/23/2015) • Added ECN_DVSEC-2015-08-04	2/6/2015 11/24/2015

Revision	Revision History	Date
	• Applied ECN PASID-ATS dated 2011-03-31 • Applied PCIE Base Spec Errata: PCIe_Base_r3 1_Errata_2015-09-18 except: o B216; RCIE o B256; grammar is not clear • Changes to Chapter 7. Software Initialization and Configuration per PCIe_4.0_regs_0-3F_gord_7.docx • Added Chapter SR-IOV Spec Rev 1.2 (Rev 1.1 dated September 8, 2009 plus: o SR-IOV_11_errata_table.doc o DVSEC o 3.1 Base Spec errata) • Added Chapter ATS Spec Rev 1.2 (Rev 1.1 dated January 26, 2009 plus: o ECN-PASID-ATS o 3.1 Base Spec errata)	
	2/18/2016 Changes from the Protocol Working Group • Applied changes from the following documents: o FC Init/Revision | scaled-flow-control-pcie-base40-2016-01-07.pdf (Steve.G) o Register updates for integrated legacy specs | PCIe_4.0_regs_0-3F_gord_8.docx (GordC) o Tag Scaling PCIe 4_0 Tag Field scaling 2015-11-23 clean.docx (JoeC) o MSI/MSI-X | PCIe 4_0 MSI & MSI-X 2015-12-18 clean.docx (JoeC); register diagrams TBD on next draft. o REPLAY_TIMER/Ack/FC Limits | Ack_FC_Replay_Timers_ver8 (PeterJ)	2/18/16
	Chapter 10. SR-IOV related changes: • Incorporated "SR-IOV and Sharing Specification" Revision 1.1 dated January 20, 2010 (sr-iov1_1_20Jan10.pdf) as Chapter 10, with changes from the following documents o Errata for the PCI Express® Base Specification Revision 3.1, Single Root I/O Virtualization and Sharing Revision 1.1, Address Translation and Sharing Revision 1.1, and M.2 Specification Revision 1.0: PCIe_Base_r3 1_Errata_2015-09-18_clean.pdf o ECN__Integrated_Endpoints_and_IOV_updates__19 Nov 2015_Final.pdf o Changes marked "editorial" only in marked PDF: sr-iov1_1_20Jan10-steve-manning-comments.pdf	4/26/16 [snapshot]
	Chapter 9. Electrical Sub-Block related changes: o Source: WG approved word document from Dan Froelich (FileName: Electrical-PCI_Express_Base_4.0r0.7_April_7_wg_approved_redo_for_figure_corruption.docx.)	5/23/16[snapshot]
4.0	Version 0.7 continued… Chapter 4. PHY Logical Changes based on: • Chapter4-PCI_Express_Base_4 0r0 7_May3_2016_draft.docx Chapter 7. . PHY Logical Changes based on: • PCI_Express_Base_4 0r0 7_Phy-Logical_Ch7_Delta_28_Apr_2016.docx - - - - - - - - - Changes incorporated into the August 2016 4.0 r0.7 Draft PDF - -- - - - - - - - - - June 16 Feedback from PWG on the May 2016 snapshot PWG Feedback on 4.0 r0.7 Feb-Apr-May-2016 Drafts *EWG Feedback: ---CB-PCI_Express_Base_4.0r0.7_May-2016 (Final).fdf ---EWG f/b: Electrical-PCI_Express_Base_4.0r0.7_April_7_wg_approved_redo_for_figure_corruption_Broadco.docx *PWG Feedback: -PWG 0.7 fix list part1 and part 2.docx -PWG 0 7 fix list part3a.docx	8/30/16

Revision	Revision History	Date
	---PCI_Express_Base_4.0r0.7_pref_April-2016_chp5_PM_stuff_only_ver3.docx ---PCI_Express_Base_4.0r0.7_pref_April-2016_chp5_PM_stuff_only_ver3.docx ---scaled-flow-control-pcie-base40-2016-07-07.pdf ---ECN_NOP_DLLP-2014-06-11_clean.pdf ---ECN_RN_29_Aug_2013.pdf ---3p1_active_errata_list_mpcie_28Aug2014.doc ---3p1_active_errata_list_mpcie_part2_11Sept2014.doc ---lane-margining-capability-snapshot-2016-06-16.pdf ---Emergency Power Reduction Mechanism with PWRBRK Signal ECN -PWG 0 7 fix list part4.docx ---ECN_Conventional_Adv_Caps_27Jul06.pdf ---10-bit Tag related SR-IOV Updates *Other: ---Merged Acknowledgements back pages from SR-IOV and ATS specifications into the main base spec. Acknowledgements page.	
	- - - - - - - - - Changes since August 2016 for the September 2016 4.0 r0.7 Draft PDF- - - - - Applied: PWG Feedback/Corrections on August draft ECN_SR-IOV_Table_Updates_16-June-2016.doc	9/28/16
	- - - - - - - Changes since September 28 2016 for the October 2016 4.0 r0.7 Draft PDF- - - - EWG: Updates to Chapter 9- Electrical Sub-block (Sections: 9.4.1.4, 9.6.5.1, 9.6.5.2, 9.6.7)	10/7/16
	PWG: Updates to Sections: 3.2.1, 3.3, 3.5.1, 7.13, 7.13.3 (Figure: Data Link Status Register)	10/21/16
	- - - - - - - Changes to the October 13 2016 4.0 r0.7 Draft PDF- - - - EWG: Updates to Chapter 9- Electrical Sub-block (Section 9.3.3.9 and Figure 9-9 caption)	
	- - - - - - - Changes to the November 3 2016 4.0 r0.7 Draft PDF- - - - Section 2.6.1 Flow Control Rules: Updated Scaled Flow Control sub-bullet under FC initialization bullet (before Table 2-43)	11/3/16
	- - - - - - - Changes to the November 11 2016 4.0 r0.7 Draft PDF- - - - Added M-PCIe statement to the Open Issues page Updated date to November 11, 2016	11/11/16
	- Version 0.9: Based on PCI Express® Base Specification Version 4.0 Revision 0.7 (11/11/2016) Incorporated the following ECNs: ---ECN-Hierarchy_ID-2017-02-23 ---ECN_FPB_9_Feb_2017 ---ECN Expanded Resizable BARs 2016-04-18 ---ECN-VF-Resizable-BARs_6-July-2016 --- Chapter 7 reorganized: • New section 7.6 created per a PWG-approved reorganization to move sections 7.5, 7.6,. and 7.10 to subsections 7.6.1 through 7.6.3 resp. • New section 7.7 created per a PWG-approved reorganization to move sections 7.7, 7.8,.7.12, 7.13, 7.40, 7.41 and 7.20 to subsections 7.7.1 through 7.7.7 resp. • New section 7.9 created per a PWG-approved reorganization to move sections 7.15, 7.22, 7.16, 7.23, 7.39, 7.24, 7.17, 7.18, 7.21, 7.25, 7.28, 7.30, 7.33, 7.34, 7.35, 7.38, and 7.42 to subsections 7.9.1 through 7.9.17 resp. ---Removed Chapter 8: M-PCIe Logical Sub-Block ---Updated Chapter 9 (8 now), EWG Updates to Chapter 9- Electrical Sub-block per: Chapter9-PCI_Express_Base_4 0r09_March_30--2017_approved.docx ---Updated Chapter 4: Physical Layer Logical Block per PCI_Express_Base_4 0_r0 9_Chapter4_Final_Draft.docx ---Updated Figures in Chapter 10: ATS Specification	April 28 2017

Revision	Revision History	Date
	---Removed Appendix H: M-PCIe timing Diagrams ---Removed Appendix I: M-PCIe Compliance Patterns, pursuant to removing the M-PCIe Chapter this 0.9 version of the 4.0 Base Spec. ---Added Appendix H: Flow Control Update Latency and ACK Update Latency Calculations ---Added Appendix I: Vital Product Data (VPD) ---Updated editorial feedback on the Appendix section per: PCI_Express_Base_4.0r0.7_appendixes_November-11-2016_combined-editorial.docx ---Deleted references to M-PCIe throughout the document. ---Updated Chapter 9 (8 now), EWG Updates to Chapter 9- Electrical Sub-block per: Chapter9-PCI_Express_Base_4 0r09_March_30--2017_approved.docx ---Updated Chapter 4: Physical Layer Logical Block per PCI_Express_Base_4 0_r0 9_Chapter4_Final_Draft.docx ---Updated Figures in Chapter 10: ATS Specification ---Added Appendix H: Flow Control Update Latency and ACK Update Latency Calculations ---Following items that were marked deleted in the Change Bar version of the April 28th snapshot have been "accepted" to no longer show up: **pp 1070: Lane Equalization Control 2 Register (Offset TBD)** Comment: Deleted per: PCI_Express_Base_4 0r0 7_Phy-Logical_Ch7_Delta_28_Apr_2016.docx **pp 1074: Physical Layer 16.0 GT/s Margining Extended Capability section** Comment: Deleted per: PCI_Express_Base_4 0r0 7_Phy-Logical_Ch7_Delta_28_Apr_2016.docx Comment: Replaced by Section, "Lane Margining at the Receiver Extended Capability" per Fix3a #83 lane-margining-capability-snapshot-2016-06-16.pdf ---Incorporated: PCIe 4_0 Tag Field scaling 2017-03-31.docx ---Vital Product Data (VPD) -----Added Section 6.28 -----Added Section 7.9.4 ---Incorporated feedback from April 28th snapshot.[source: 3 fdf files] ---Completed editorial feedback on the Appendix section per: PCI_Express_Base_4.0r0.7_appendixes_November-11-2016_combined-editorial.docx ---Incorporated ECN EMD for MSI 2016-05-10 ---Updated per: **PWG F2F changes** from: PCI_Express_Base_4.0r0.7_pref_November-11-2016-F2F-2017-03-16-2017-03-30-sdg.docx ---Updated figures per following lists (Gord Caruk): PCIe_4 0_fix_drawing_items.doc PCIe_4 0_fix_drawing_items_part2.doc	May 26, 2017
	Version 0.91 ****Note this version will be used as the base for the PCI Express® Base Specification Revision 5.0**** *Item numbers are with reference to PWG CheckList (https://members.pcisig.com/wg/PCIe-Protocol/document/10642)* ---Moved Flattening Portal Bridge Section 7.10 to Section 7.8.10. PWG Checklist Items #12.1 ---Fixed misc. feedback that needed clarification from the 0.9 version. Issues fall under the categories of figure updates, broken cross references. Also incorporated feedback received from member review of the 4.0 version rev. 0.9 Base Spec. ---Updated to reconcile issues related to incorporating the Extended Message Data for MSI ECN. PWG Checklist Items #22 ---Completed incorporating all resolved editorial items from PWG Checklist Items #14, 14.1,15.1, 36, 42. TBD: Some minor editorial items from #13, #14 and #15 have been deferred to post 0.91 by reviewers. TBD: Errata and NPEM ECN	August 17, 2017

Revision	Revision History	Date
1.0	ECN: ECN_Native_PCIe_Enclosure_Management_v10August2017.docx Deleted Section 5.11.1 through Section 5.14 Changes tracked by items 34.01 34.02 34.04 34.05 34.11 in the PWG checklist Errata: B265, C266, 267, 268, B269, A270, A271, B274, C275, B276, B277, B278, B279, B280, B281, B283, B284, B285, B286, B288, B289, B292, B293, B294, B295, B297, B299, B300, B301 Other minor edits per: NCB-PCI_Express_Base_4.0r0.91_August-17-2017__dh_sdg_Annot_2.fdf	August 28, 2017
	Applied fixes and corrections captured in NCB-PCI_Express_Base_4.0r1.0_August-28-2017.fdf (Revision 8): https://members.pcisig.com/wg/PCIe-Protocol/document/10770 Updated contributor list in Appendix section.	September 20, 2017
	Updated contributor list in Appendix section. Inserted correct Figure 6-2. Applied minor fixes and corrections captured in: NCB-PCI_Express_Base_4.0r1.0_September-20-2017 https://members.pcisig.com/wg/PCIe-Protocol/document/10770	September 27, 2017

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

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

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

Membership Services
www.pcisig.com
E-mail: administration@pcisig.com
Phone: 503-619-0569
Fax: 503-644-6708

Technical Support
techsupp@pcisig.com

DISCLAIMER

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

PCI, PCI Express, PCIe, and PCI-SIG are trademarks or registered trademarks of PCI-SIG.
All other product names are trademarks, registered trademarks, or servicemarks of their respective owners.

Copyright © 2002-2017 PCI-SIG

Contents

OBJECTIVE OF THE SPECIFICATION **42**
DOCUMENT ORGANIZATION **42**
DOCUMENTATION CONVENTIONS **42**
TERMS AND ACRONYMS **43**
REFERENCE DOCUMENTS **53**
1 INTRODUCTION **54**
1.1 A THIRD GENERATION I/O INTERCONNECT 54
1.2 PCI EXPRESS LINK 56
1.3 PCI EXPRESS FABRIC TOPOLOGY 57
1.3.1 Root Complex *57*
1.3.2 Endpoints *58*
1.3.3 Switch *60*
1.3.4 Root Complex Event Collector *61*
1.3.5 PCI Express to PCI/PCI-X Bridge *62*
1.4 HARDWARE/SOFTWARE MODEL FOR DISCOVERY, CONFIGURATION AND OPERATION .. 62
1.5 PCI EXPRESS LAYERING OVERVIEW 63
1.5.1 Transaction Layer *64*
1.5.2 Data Link Layer *64*
1.5.3 Physical Layer *65*
1.5.4 Layer Functions and Services *65*
2 TRANSACTION LAYER SPECIFICATION **69**
2.1 TRANSACTION LAYER OVERVIEW 69
2.1.1 Address Spaces, Transaction Types, and Usage *70*
2.1.2 Packet Format Overview *72*
2.2 TRANSACTION LAYER PROTOCOL - PACKET DEFINITION 74
2.2.1 Common Packet Header Fields *74*
2.2.2 TLPs with Data Payloads - Rules *77*
2.2.3 TLP Digest Rules *81*
2.2.4 Routing and Addressing Rules *81*
2.2.5 First/Last DW Byte Enables Rules *84*
2.2.6 Transaction Descriptor *87*
2.2.7 Memory, I/O, and Configuration Request Rules *97*
2.2.8 Message Request Rules *104*
2.2.9 Completion Rules *125*
2.2.10 TLP Prefix Rules *128*
2.3 HANDLING OF RECEIVED TLPS 132
2.3.1 Request Handling Rules *135*
2.3.2 Completion Handling Rules *147*
2.4 TRANSACTION ORDERING 150
2.4.1 Transaction Ordering Rules *150*

2.4.2 Update Ordering and Granularity Observed by a Read Transaction ... 154
2.4.3 Update Ordering and Granularity Provided by a Write Transaction ... 155
2.5 VIRTUAL CHANNEL (VC) MECHANISM ... 155
2.5.1 Virtual Channel Identification (VC ID) ... 157
2.5.2 TC to VC Mapping ... 158
2.5.3 VC and TC Rules ... 159
2.6 ORDERING AND RECEIVE BUFFER FLOW CONTROL ... 160
2.6.1 Flow Control Rules ... 161
2.7 DATA INTEGRITY ... 173
2.7.1 ECRC Rules ... 174
2.7.2 Error Forwarding ... 178
2.8 COMPLETION TIMEOUT MECHANISM ... 180
2.9 LINK STATUS DEPENDENCIES ... 181
2.9.1 Transaction Layer Behavior in DL_Down Status ... 181
2.9.2 Transaction Layer Behavior in DL_Up Status ... 182
2.9.3 Transaction Layer Behavior During Downstream Port Containment ... 183

3 DATA LINK LAYER SPECIFICATION ... 185

3.1 DATA LINK LAYER OVERVIEW ... 185
3.2 DATA LINK CONTROL AND MANAGEMENT STATE MACHINE ... 187
3.2.1 Data Link Control and Management State Machine Rules ... 188
3.3 DATA LINK FEATURE EXCHANGE ... 191
3.4 FLOW CONTROL INITIALIZATION PROTOCOL ... 193
3.4.1 Flow Control Initialization State Machine Rules ... 193
3.4.2 Scaled Flow Control ... 197
3.5 DATA LINK LAYER PACKETS (DLLPS) ... 198
3.5.1 Data Link Layer Packet Rules ... 198
3.6 DATA INTEGRITY ... 204
3.6.1 Introduction ... 204
3.6.2 LCRC, Sequence Number, and Retry Management (TLP Transmitter) ... 205
3.6.3 LCRC and Sequence Number (TLP Receiver) ... 217

4 PHYSICAL LAYER LOGICAL BLOCK ... 225

4.1 INTRODUCTION ... 225
4.2 LOGICAL SUB-BLOCK ... 225
4.2.1 Encoding for 2.5 GT/s and 5.0 GT/s Data Rates ... 226
4.2.2 Encoding for 8.0 GT/s and Higher Data Rates ... 234
4.2.3 Link Equalization Procedure for 8.0 GT/s and Higher Data Rates ... 253
4.2.4 Link Initialization and Training ... 263
4.2.5 Link Training and Status State Machine (LTSSM) Descriptions ... 283
4.2.6 Link Training and Status State Rules ... 287
4.2.7 Clock Tolerance Compensation ... 360
4.2.8 Compliance Pattern in 8b/10b Encoding ... 368
4.2.9 Modified Compliance Pattern in 8b/10b Encoding ... 370
4.2.10 Compliance Pattern in 128b/130b Encoding ... 371
4.2.11 Modified Compliance Pattern in 128b/130b Encoding ... 373
4.2.12 Jitter Measurement Pattern in 128b/130b ... 374

4.2.13 Lane Margining at Receiver ... 374
4.3 RETIMERS ... 388
4.3.1 Retimer Requirements ... 388
4.3.2 Supported Topologies ... 389
4.3.3 Variables ... 391
4.3.4 Receiver Impedance Propagation Rules ... 391
4.3.5 Switching Between Modes ... 391
4.3.6 Forwarding Rules ... 392
4.3.7 Execution Mode Rules ... 409
4.3.8 Retimer Latency ... 416
4.3.9 SRIS ... 416
4.3.10 L1 PM Substates Support ... 418
4.3.11 Retimer Configuration Parameters ... 421
4.3.12 In Band Register Access ... 423

5 POWER MANAGEMENT ... 424

5.1 OVERVIEW ... 424
5.2 LINK STATE POWER MANAGEMENT ... 425
5.3 PCI-PM SOFTWARE COMPATIBLE MECHANISMS ... 431
5.3.1 Device Power Management States (D-States) of a Function ... 431
5.3.2 PM Software Control of the Link Power Management State ... 436
5.3.3 Power Management Event Mechanisms ... 441
5.4 NATIVE PCI EXPRESS POWER MANAGEMENT MECHANISMS ... 448
5.4.1 Active State Power Management (ASPM) ... 448
5.5 L1 PM SUBSTATES ... 467
5.5.1 Entry conditions for L1 PM Substates and L1.0 Requirements ... 471
5.5.2 L1.1 Requirements ... 472
5.5.3 L1.2 Requirements ... 474
5.5.4 L1 PM Substates Configuration ... 477
5.5.5 L1 PM Substates Timing Parameters ... 478
5.6 AUXILIARY POWER SUPPORT ... 479
5.7 POWER MANAGEMENT SYSTEM MESSAGES AND DLLPS ... 479
5.8 PCI FUNCTION POWER STATE TRANSITIONS ... 480
5.9 FUNCTION POWER MANAGEMENT POLICIES ... 481
5.9.1 State Transition Recovery Time Requirements ... 485
5.10 PCI BRIDGES AND POWER MANAGEMENT ... 486
5.10.1 Switches and PCI Express to PCI Bridges ... 487
5.11 POWER MANAGEMENT EVENTS ... 487

6 SYSTEM ARCHITECTURE ... 489

6.1 INTERRUPT AND PME SUPPORT ... 489
6.1.1 Rationale for PCI Express Interrupt Model ... 489
6.1.2 PCI-compatible INTx Emulation ... 490
6.1.3 INTx Emulation Software Model ... 490
6.1.4 MSI and MSI-X Operation ... 490
6.1.5 PME Support ... 498
6.1.6 Native PME Software Model ... 499

6.1.7 Legacy PME Software Model ... 499
6.1.8 Operating System Power Management Notification ... 500
6.1.9 PME Routing Between PCI Express and PCI Hierarchies ... 500
6.2 ERROR SIGNALING AND LOGGING ... 500
6.2.1 Scope ... 501
6.2.2 Error Classification ... 501
6.2.3 Error Signaling ... 503
6.2.4 Error Logging ... 511
6.2.5 Sequence of Device Error Signaling and Logging Operations ... 518
6.2.6 Error Message Controls ... 519
6.2.7 Error Listing and Rules ... 519
6.2.8 Virtual PCI Bridge Error Handling ... 524
6.2.9 Internal Errors ... 526
6.2.10 Downstream Port Containment (DPC) ... 526
6.3 VIRTUAL CHANNEL SUPPORT ... 536
6.3.1 Introduction and Scope ... 536
6.3.2 TC/VC Mapping and Example Usage ... 537
6.3.3 VC Arbitration ... 539
6.3.4 Isochronous Support ... 547
6.4 DEVICE SYNCHRONIZATION ... 550
6.5 LOCKED TRANSACTIONS ... 551
6.5.1 Introduction ... 551
6.5.2 Initiation and Propagation of Locked Transactions - Rules ... 552
6.5.3 Switches and Lock - Rules ... 553
6.5.4 PCI Express/PCI Bridges and Lock - Rules ... 553
6.5.5 Root Complex and Lock - Rules ... 554
6.5.6 Legacy Endpoints ... 554
6.5.7 PCI Express Endpoints ... 554
6.6 PCI EXPRESS RESET - RULES ... 554
6.6.1 Conventional Reset ... 554
6.6.2 Function Level Reset (FLR) ... 557
6.7 PCI EXPRESS HOT-PLUG SUPPORT ... 561
6.7.1 Elements of Hot-Plug ... 561
6.7.2 Registers Grouped by Hot-Plug Element Association ... 568
6.7.3 PCI Express Hot-Plug Events ... 570
6.7.4 Firmware Support for Hot-Plug ... 573
6.7.5 Async Removal ... 573
6.8 POWER BUDGETING CAPABILITY ... 574
6.8.1 System Power Budgeting Process Recommendations ... 574
6.9 SLOT POWER LIMIT CONTROL ... 575
6.10 ROOT COMPLEX TOPOLOGY DISCOVERY ... 578
6.11 LINK SPEED MANAGEMENT ... 580
6.12 ACCESS CONTROL SERVICES (ACS) ... 581
6.12.1 ACS Component Capability Requirements ... 582
6.12.2 Interoperability ... 587
6.12.3 ACS Peer-to-Peer Control Interactions ... 587

6.12.4 ACS Violation Error Handling ... 588
6.12.5 ACS Redirection Impacts on Ordering Rules ... 589
6.13 ALTERNATIVE ROUTING-ID INTERPRETATION (ARI) ... 591
6.14 MULTICAST OPERATIONS ... 595
6.14.1 Multicast TLP Processing ... 595
6.14.2 Multicast Ordering ... 599
6.14.3 Multicast Capability Structure Field Updates ... 599
6.14.4 MC Blocked TLP Processing ... 600
6.14.5 MC_Overlay Mechanism ... 600
6.15 ATOMIC OPERATIONS (ATOMICOPS) ... 603
6.15.1 AtomicOp Use Models and Benefits ... 605
6.15.2 AtomicOp Transaction Protocol Summary ... 605
6.15.3 Root Complex Support for AtomicOps ... 607
6.15.4 Switch Support for AtomicOps ... 608
6.16 DYNAMIC POWER ALLOCATION (DPA) CAPABILITY ... 608
6.16.1 DPA Capability with Multi-Function Devices ... 610
6.17 TLP PROCESSING HINTS (TPH) ... 610
6.17.1 Processing Hints ... 610
6.17.2 Steering Tags ... 611
6.17.3 ST Modes of Operation ... 612
6.17.4 TPH Capability ... 613
6.18 LATENCY TOLERANCE REPORTING (LTR) MECHANISM ... 613
6.19 OPTIMIZED BUFFER FLUSH/FILL (OBFF) MECHANISM ... 620
6.20 PASID TLP PREFIX ... 624
6.20.1 Managing PASID TLP Prefix Usage ... 624
6.20.2 PASID TLP Layout ... 625
6.21 LIGHTWEIGHT NOTIFICATION (LN) PROTOCOL ... 628
6.21.1 LN Protocol Operation ... 630
6.21.2 LN Registration Management ... 631
6.21.3 LN Ordering Considerations ... 632
6.21.4 LN Software Configuration ... 632
6.21.5 LN Protocol Summary ... 633
6.22 PRECISION TIME MEASUREMENT (PTM) MECHANISM ... 634
6.22.1 Introduction ... 634
6.22.2 PTM Link Protocol ... 636
6.22.3 Configuration and Operational Requirements ... 638
6.23 READINESS NOTIFICATIONS (RN) ... 644
6.23.1 Device Readiness Status (DRS) ... 644
6.23.2 Function Readiness Status (FRS) ... 645
6.23.3 FRS Queuing ... 646
6.24 ENHANCED ALLOCATION ... 647
6.25 EMERGENCY POWER REDUCTION STATE ... 649
6.26 HIERARCHY ID MESSAGE ... 651
6.27 FLATTENING PORTAL BRIDGE (FPB) ... 655
6.27.1 Introduction ... 655
6.27.2 Hardware and Software Requirements ... 659

6.28 VITAL PRODUCT DATA (VPD) 666
6.29 NATIVE PCIE ENCLOSURE MANAGEMENT 671

7 SOFTWARE INITIALIZATION AND CONFIGURATION 675

7.1 CONFIGURATION TOPOLOGY 675
7.2 PCI EXPRESS CONFIGURATION MECHANISMS 676
7.2.1 PCI-compatible Configuration Mechanism 677
7.2.2 PCI Express Enhanced Configuration Access Mechanism (ECAM) 677
7.2.3 Root Complex Register Block 682
7.3 CONFIGURATION TRANSACTION RULES 682
7.3.1 Device Number 682
7.3.2 Configuration Transaction Addressing 683
7.3.3 Configuration Request Routing Rules 683
7.3.4 PCI Special Cycles 685
7.4 CONFIGURATION REGISTER TYPES 685
7.5 PCI AND PCIE CAPABILITIES REQUIRED BY THE BASE SPEC FOR ALL PORTS 687
7.5.1 PCI-Compatible Configuration Registers 687
7.5.2 PCI Power Management Capability Structure 717
7.5.3 PCI Express Capability Structure 724
7.6 PCI EXPRESS EXTENDED CAPABILITIES 796
7.6.1 Extended Capabilities in Configuration Space 796
7.6.2 Extended Capabilities in the Root Complex Register Block 796
7.6.3 PCI Express Extended Capability Header 797
7.7 PCI AND PCIE CAPABILITIES REQUIRED BY THE BASE SPEC IN SOME SITUATIONS 798
7.7.1 MSI Capability Structures 798
7.7.2 MSI-X Capability and Table Structure 806
7.7.3 Secondary PCI Express Extended Capability 815
7.7.4 Data Link Feature Extended Capability 821
7.7.5 Physical Layer 16.0 GT/s Extended Capability 824
7.7.6 Lane Margining at the Receiver Extended Capability 832
7.7.7 ACS Extended Capability 836
7.8 COMMON PCI AND PCIE CAPABILITIES 842
7.8.1 Power Budgeting Capability 842
7.8.2 Latency Tolerance Reporting (LTR) Capability 847
7.8.3 L1 PM Substates Extended Capability 850
7.8.4 Advanced Error Reporting Capability 856
7.8.5 Enhanced Allocation (EA) Capability Structure 874
7.8.6 Resizable BAR Capability 881
7.8.7 ARI Capability 887
7.8.8 PASID Extended Capability Structure 889
7.8.9 Function Readiness Status (FRS) Queuing Extended Capability 892
7.8.10 Flattening Portal Bridge (FPB) Capability 897
7.9 ADDITIONAL PCI AND PCIE CAPABILITIES 908
7.9.1 Virtual Channel Capability 908
7.9.2 Multi-Function Virtual Channel Capability 923
7.9.3 Device Serial Number Capability 935
7.9.4 Vendor-Specific Capability 937

7.9.5 Vendor-Specific Extended Capability ... 938
7.9.6 Designated Vendor-Specific Extended Capability (DVSEC) ... 941
7.9.7 RCRB Header Capability ... 943
7.9.8 PCI Express Root Complex Link Declaration Capability ... 946
7.9.9 PCI Express Root Complex Internal Link Control Capability ... 952
7.9.10 PCI Express Root Complex Event Collector Endpoint Association Capability ... 960
7.9.11 Multicast Capability ... 962
7.9.12 Dynamic Power Allocation (DPA) Capability ... 968
7.9.13 TPH Requester Capability ... 972
7.9.14 LNR Extended Capability ... 976
7.9.15 DPC Extended Capability ... 978
7.9.16 Precision Time Management (PTM) Capability ... 992
7.9.17 Readiness Time Reporting Extended Capability ... 995
7.9.18 Hierarchy ID Extended Capability ... 999
7.9.19 VPD Capability ... 1008
7.9.20 Native PCIe Enclosure Management Extended Capability ... 1010

8 ELECTRICAL SUB-BLOCK ... 1016

8.1 ELECTRICAL SPECIFICATION INTRODUCTION ... 1016
8.2 INTEROPERABILITY CRITERIA ... 1016
8.2.1 Data Rates ... 1016
8.2.2 Refclk Architectures ... 1016
8.3 TRANSMITTER SPECIFICATION ... 1017
8.3.1 Measurement Setup for Characterizing Transmitters ... 1017
8.3.2 Voltage Level Definitions ... 1018
8.3.3 Tx Voltage Parameters ... 1019
8.3.4 Transmitter Margining ... 1029
8.3.5 Tx Jitter Parameters ... 1030
8.3.6 Data Rate Dependent Parameters ... 1037
8.3.7 Tx and Rx Return Loss ... 1040
8.3.8 Transmitter PLL Bandwidth and Peaking ... 1041
8.3.9 Data Rate Independent Tx Parameters ... 1042
8.4 RECEIVER SPECIFICATIONS ... 1043
8.4.1 Receiver Stressed Eye Specification ... 1043
8.4.2 Stressed Eye Test ... 1053
8.4.3 Common Receiver Parameters ... 1061
8.4.4 Lane Margining at the Receiver – Electrical Requirements ... 1064
8.4.5 Low Frequency and Miscellaneous Signaling Requirements ... 1067
8.5 CHANNEL TOLERANCING ... 1070
8.5.1 Channel Compliance Testing ... 1070
8.6 REFCLK SPECIFICATIONS ... 1079
8.6.1 Refclk Test Setup ... 1079
8.6.2 REFCLK AC Specifications ... 1080
8.6.3 Data Rate Independent Refclk Parameters ... 1083
8.6.4 Refclk Architectures Supported ... 1084
8.6.5 Filtering Functions Applied to Raw Data ... 1084
8.6.6 Common Refclk Rx Architecture (CC) ... 1085

8.6.7 Jitter Limits for Refclk Architectures ... 1088
8.6.8 Form Factor Requirements for RefClock Architectures ... 1088

9 SINGLE ROOT I/O VIRTUALIZATION AND SHARING ... 1090

9.1 ARCHITECTURAL OVERVIEW ... 1090
9.1.1 PCI Technologies Interoperability ... 1102
9.2 INITIALIZATION AND RESOURCE ALLOCATION ... 1103
9.2.1 SR-IOV Resource Discovery ... 1103
9.2.2 Reset Mechanisms ... 1107
9.2.3 IOV Re-initialization and Reallocation ... 1108
9.2.4 VF Migration ... 1108
9.3 CONFIGURATION ... 1112
9.3.1 Overview ... 1112
9.3.2 Configuration Space ... 1112
9.3.3 SR-IOV Extended Capability ... 1112
9.3.4 PF/VF Configuration Space Header ... 1130
9.3.5 PCI Express Capability ... 1134
9.3.6 PCI Standard Capabilities ... 1140
9.3.7 PCI Express Extended Capabilities ... 1141
9.4 ALL VFS ASSOCIATED WITH THE SAME PF SHALL REPORT THE SAME TIME VALUES.ERROR HANDLING ... 1152
9.4.1 Baseline Error Reporting ... 1152
9.4.2 Advanced Error Reporting ... 1153
9.5 INTERRUPTS ... 1158
9.5.1 Interrupt Mechanisms ... 1158
9.6 POWER MANAGEMENT ... 1159
9.6.1 VF Device Power Management States ... 1160
9.6.2 PF Device Power Management States ... 1161
9.6.3 Link Power Management State ... 1161
9.6.4 VF Power Management Capability ... 1161
9.6.5 VF EmergencyPower Reduction State ... 1162

10 ATS SPECIFICATION ... 1163

10.1 ARCHITECTURAL OVERVIEW ... 1163
10.1.1 Address Translation Services (ATS) Overview ... 1165
10.1.2 Page Request Interface Extension ... 1170
10.1.3 Process Address Space ID (PASID) ... 1172
10.2 ATS TRANSLATION SERVICES ... 1173
10.2.1 Memory Requests with Address Type ... 1173
10.2.2 Translation Requests ... 1174
10.2.3 Translation Completion ... 1177
10.2.4 Completions with Multiple Translations ... 1185
10.3 ATS INVALIDATION ... 1187
10.3.1 Invalidate Request ... 1187
10.3.2 Invalidate Completion ... 1188
10.3.3 Invalidate Completion Semantics ... 1190
10.3.4 Request Acceptance Rules ... 1191

10.3.5 Invalidate Flow Control ... *1191*
10.3.6 Invalidate Ordering Semantics ... *1192*
10.3.7 Implicit Invalidation Events ... *1193*
10.3.8 PASID TLP Prefix and Global Invalidate ... *1194*
10.4 PAGE REQUEST SERVICES ... 1195
10.4.1 Page Request Message ... *1196*
10.4.2 Page Request Group Response Message ... *1200*
10.5 CONFIGURATION ... 1202
10.5.1 ATS Extended Capability Structure ... *1202*
10.5.2 Page Request Extended Capability Structure ... *1205*

A. ISOCHRONOUS APPLICATIONS ... **1212**

A.1. INTRODUCTION ... 1212
A.2. ISOCHRONOUS CONTRACT AND CONTRACT PARAMETERS ... 1214
A.2.1. Isochronous Time Period and Isochronous Virtual Timeslot ... *1215*
A.2.2. Isochronous Payload Size ... *1216*
A.2.3. Isochronous Bandwidth Allocation ... *1216*
A.2.4. Isochronous Transaction Latency ... *1217*
A.2.5. An Example Illustrating Isochronous Parameters ... *1218*
A.3. ISOCHRONOUS TRANSACTION RULES ... 1219
A.4. TRANSACTION ORDERING ... 1219
A.5. ISOCHRONOUS DATA COHERENCY ... 1219
A.6. FLOW CONTROL ... 1220
A.7. CONSIDERATIONS FOR BANDWIDTH ALLOCATION ... 1220
A.7.1. Isochronous Bandwidth of PCI Express Links ... *1220*
A.7.2. Isochronous Bandwidth of Endpoints ... *1220*
A.7.3. Isochronous Bandwidth of Switches ... *1220*
A.7.4. Isochronous Bandwidth of Root Complex ... *1221*
A.8. CONSIDERATIONS FOR PCI EXPRESS COMPONENTS ... 1221
A.8.1. An Endpoint as a Requester ... *1221*
A.8.2. An Endpoint as a Completer ... *1221*
A.8.3. Switches ... *1222*
A.8.4. Root Complex ... *1223*

B. SYMBOL ENCODING ... **1224**

C. PHYSICAL LAYER APPENDIX ... **1233**

C.1. 8B/10B DATA SCRAMBLING EXAMPLE ... 1233
C.2. 128B/130B DATA SCRAMBLING EXAMPLE ... 1239

D. REQUEST DEPENDENCIES ... **1242**

E. ID-BASED ORDERING USAGE ... **1245**

E.1. INTRODUCTION ... 1245
E.2. POTENTIAL BENEFITS WITH IDO USE ... 1246
E.2.1. Benefits for MFD/RP Direct Connect ... *1246*
E.2.2. Benefits for Switched Environments ... *1246*
E.2.3. Benefits for Integrated Endpoints ... *1247*

E.2.4. IDO Use in Conjunction with RO 1247
E.3. WHEN TO USE IDO 1247
E.4. WHEN NOT TO USE IDO 1248
E.4.1. When Not to Use IDO with Endpoints 1248
E.4.2. When Not to Use IDO with Root Ports 1248
E.5. SOFTWARE CONTROL OF IDO USE 1249
E.5.1. Software Control of Endpoint IDO Use 1249
E.5.2. Software Control of Root Port IDO Use 1250

F. MESSAGE CODE USAGE **1251**

G. **PROTOCOL MULTIPLEXING 1253**

G.1. PROTOCOL MULTIPLEXING INTERACTIONS WITH PCI EXPRESS 1256
G.2. PMUX PACKETS 1261
G.3. PMUX PACKET LAYOUT 1262
G.3.1. PMUX Packet Layout for 8b/10b Encoding 1262
G.3.2. PMUX Packet Layout at 128b/130b Encoding 1264
G.4. PMUX CONTROL 1267
G.5. PMUX EXTENDED CAPABILITY 1267
G.5.1. PCI Express Extended Header (Offset 00h) 1268
G.5.2. PMUX Capability Register (Offset 04h) 1269
G.5.3. PMUX Control Register (Offset 08h) 1270
G.5.4. PMUX Status Register (Offset 0Ch) 1272
G.5.5. PMUX Protocol Array (Offsets 10h Through 48h) 1275

H. FLOW CONTROL UPDATE LATENCY AND ACK UPDATE LATENCY CALCULATIONS **1277**

H.1. FLOW CONTROL UPDATE LATENCY 1277
H.2. ACK LATENCY 1279

ACKNOWLEDGEMENTS **1282**

Figures

FIGURE 1-1: PCI EXPRESS LINK 56
FIGURE 1-2: EXAMPLE TOPOLOGY 57
FIGURE 1-3: LOGICAL BLOCK DIAGRAM OF A SWITCH 61
FIGURE 1-4: HIGH-LEVEL LAYERING DIAGRAM 63
FIGURE 1-5: PACKET FLOW THROUGH THE LAYERS 64
FIGURE 2-1: LAYERING DIAGRAM HIGHLIGHTING THE TRANSACTION LAYER 69
FIGURE 2-2: SERIAL VIEW OF A TLP 72
FIGURE 2-3: GENERIC TLP FORMAT 73
FIGURE 2-4: FIELDS PRESENT IN ALL TLPS 74
FIGURE 2-5: FIELDS PRESENT IN ALL TLP HEADERS 75
FIGURE 2-6: EXAMPLES OF COMPLETER TARGET MEMORY ACCESS FOR FETCHADD 80
FIGURE 2-7: 64-BIT ADDRESS ROUTING 82
FIGURE 2-8: 32-BIT ADDRESS ROUTING 82
FIGURE 2-9: ID ROUTING WITH 4 DW HEADER 84
FIGURE 2-10: ID ROUTING WITH 3 DW HEADER 84
FIGURE 2-11: LOCATION OF BYTE ENABLES IN TLP HEADER 85
FIGURE 2-12: TRANSACTION DESCRIPTOR 88
FIGURE 2-13: TRANSACTION ID 88
FIGURE 2-14: ATTRIBUTES FIELD OF TRANSACTION DESCRIPTOR 95
FIGURE 2-15: REQUEST HEADER FORMAT FOR 64-BIT ADDRESSING OF MEMORY 98
FIGURE 2-16: REQUEST HEADER FORMAT FOR 32-BIT ADDRESSING OF MEMORY 98
FIGURE 2-17: REQUEST HEADER FORMAT FOR I/O TRANSACTIONS 99
FIGURE 2-18: REQUEST HEADER FORMAT FOR CONFIGURATION TRANSACTIONS 100
FIGURE 2-19: TPH TLP PREFIX 101
FIGURE 2-20: LOCATION OF PH[1:0] IN A 4 DW REQUEST HEADER 101
FIGURE 2-21: LOCATION OF PH[1:0] IN A 3 DW REQUEST HEADER 102
FIGURE 2-22: LOCATION OF ST[7:0] IN THE MEMORY WRITE REQUEST HEADER 102
FIGURE 2-23: LOCATION OF ST[7:0] IN MEMORY READ AND ATOMICOP REQUEST HEADERS ... 103
FIGURE 2-24: MESSAGE REQUEST HEADER 105
FIGURE 2-25: HEADER FOR VENDOR-DEFINED MESSAGES 114
FIGURE 2-26: HEADER FOR PCI-SIG-DEFINED VDMS 116
FIGURE 2-27: LN MESSAGE 117
FIGURE 2-28: DRS MESSAGE 118
FIGURE 2-29: FRS MESSAGE 120
FIGURE 2-30: HIERARCHY ID MESSAGE 121
FIGURE 2-31: LTR MESSAGE 123
FIGURE 2-32: OBFF MESSAGE 123
FIGURE 2-33: PTM REQUEST/RESPONSE MESSAGE 124
FIGURE 2-34: PTM RESPONSED MESSAGE (4 DW HEADER AND 1 DW PAYLOAD) 125
FIGURE 2-35: COMPLETION HEADER FORMAT 126
FIGURE 2-36: (NON-ARI) COMPLETER ID 126
FIGURE 2-37: ARI COMPLETER ID 127
FIGURE 2-38: FLOWCHART FOR HANDLING OF RECEIVED TLPS 133
FIGURE 2-39: FLOWCHART FOR SWITCH HANDLING OF TLPS 135

FIGURE 2-40: FLOWCHART FOR HANDLING OF RECEIVED REQUEST 140
FIGURE 2-41: EXAMPLE COMPLETION DATA WHEN SOME BYTE ENABLES ARE 0B 144
FIGURE 2-42: VIRTUAL CHANNEL CONCEPT – AN ILLUSTRATION 156
FIGURE 2-43: VIRTUAL CHANNEL CONCEPT – SWITCH INTERNALS (UPSTREAM FLOW) 157
FIGURE 2-44: AN EXAMPLE OF TC/VC CONFIGURATIONS 159
FIGURE 2-45: RELATIONSHIP BETWEEN REQUESTER AND ULTIMATE COMPLETER 160
FIGURE 2-46: CALCULATION OF 32-BIT ECRC FOR TLP END TO END DATA INTEGRITY PROTECTION 177
FIGURE 3-1: LAYERING DIAGRAM HIGHLIGHTING THE DATA LINK LAYER 185
FIGURE 3-2: DATA LINK CONTROL AND MANAGEMENT STATE MACHINE 188
FIGURE 3-3: VC0 FLOW CONTROL INITIALIZATION EXAMPLE WITH 8B/10B ENCODING-BASED FRAMING 196
FIGURE 3-4: DLLP TYPE AND CRC FIELDS 198
FIGURE 3-5: DATA LINK LAYER PACKET FORMAT FOR ACK AND NAK 201
FIGURE 3-6: NOP DATA LINK LAYER PACKET FORMAT 201
FIGURE 3-7: DATA LINK LAYER PACKET FORMAT FOR INITFC1 201
FIGURE 3-8: DATA LINK LAYER PACKET FORMAT FOR INITFC2 201
FIGURE 3-9: DATA LINK LAYER PACKET FORMAT FOR UPDATEFC 202
FIGURE 3-10: PM DATA LINK LAYER PACKET FORMAT 202
FIGURE 3-11: VENDOR-SPECIFIC DATA LINK LAYER PACKET FORMAT 202
FIGURE 3-12: DATA LINK FEATURE DLLP 202
FIGURE 3-13: DIAGRAM OF CRC CALCULATION FOR DLLPS 204
FIGURE 3-14: TLP WITH LCRC AND TLP SEQUENCE NUMBER APPLIED 204
FIGURE 3-15: TLP FOLLOWING APPLICATION OF TLP SEQUENCE NUMBER AND RESERVED BITS 207
FIGURE 3-16: CALCULATION OF LCRC 209
FIGURE 3-17: RECEIVED DLLP ERROR CHECK FLOWCHART 215
FIGURE 3-18: ACK/NAK DLLP PROCESSING FLOWCHART 216
FIGURE 3-19: RECEIVE DATA LINK LAYER HANDLING OF TLPS 220
FIGURE 4-1: LAYERING DIAGRAM HIGHLIGHTING PHYSICAL LAYER 225
FIGURE 4-2: CHARACTER TO SYMBOL MAPPING 226
FIGURE 4-3: BIT TRANSMISSION ORDER ON PHYSICAL LANES - X1 EXAMPLE 227
FIGURE 4-4: BIT TRANSMISSION ORDER ON PHYSICAL LANES - X4 EXAMPLE 227
FIGURE 4-5: TLP WITH FRAMING SYMBOLS APPLIED 230
FIGURE 4-6: DLLP WITH FRAMING SYMBOLS APPLIED 231
FIGURE 4-7: FRAMED TLP ON A X1 LINK 231
FIGURE 4-8: FRAMED TLP ON A X2 LINK 232
FIGURE 4-9: FRAMED TLP ON A X4 LINK 232
FIGURE 4-10: LFSR WITH 8B/10B SCRAMBLING POLYNOMIAL 234
FIGURE 4-11: EXAMPLE OF BIT TRANSMISSION ORDER IN A X1 LINK SHOWING 130 BITS OF A BLOCK 235
FIGURE 4-12: EXAMPLE OF BIT PLACEMENT IN A X4 LINK WITH ONE BLOCK PER LANE 236
FIGURE 4-13: LAYOUT OF FRAMING TOKENS 239
FIGURE 4-14: TLP AND DLLP LAYOUT 241
FIGURE 4-15: PACKET TRANSMISSION IN A X8 LINK 241
FIGURE 4-16: NULLIFIED TLP LAYOUT IN A X8 LINK WITH OTHER PACKETS 242

FIGURE 4-17: SKP ORDERED SET OF LENGTH 66-BIT IN A X8 LINK 242
FIGURE 4-18: LFSR WITH SCRAMBLING POLYNOMIAL IN 8.0 GT/S AND ABOVE DATA RATE 250
FIGURE 4-19: ALTERNATE IMPLEMENTATION OF THE LFSR FOR DESCRAMBLING..... 252
FIGURE 4-20: 8.0 GT/S EQUALIZATION FLOW..... 260
FIGURE 4-21: 16.0 GT/S EQUALIZATION FLOW..... 261
FIGURE 4-22: ELECTRICAL IDLE EXIT ORDERED SET FOR 8.0 GT/S AND ABOVE DATA RATES... 273
FIGURE 4-23: MAIN STATE DIAGRAM FOR LINK TRAINING AND STATUS STATE MACHINE 288
FIGURE 4-24: DETECT SUBSTATE MACHINE 290
FIGURE 4-25: POLLING SUBSTATE MACHINE 301
FIGURE 4-26: CONFIGURATION SUBSTATE MACHINE..... 317
FIGURE 4-27: RECOVERY SUBSTATE MACHINE..... 344
FIGURE 4-28: L0S SUBSTATE MACHINE 350
FIGURE 4-29: L1 SUBSTATE MACHINE..... 352
FIGURE 4-30: L2 SUBSTATE MACHINE..... 354
FIGURE 4-31: LOOPBACK SUBSTATE MACHINE..... 359
FIGURE 4-32: RECEIVER NUMBER ASSIGNMENT 376
FIGURE 4-33: SUPPORTED TOPOLOGIES 390
FIGURE 4-34: RETIMER CLKREQ# CONNECTION TOPOLOGY 420
FIGURE 5-1: LINK POWER MANAGEMENT STATE FLOW DIAGRAM 428
FIGURE 5-2: ENTRY INTO THE L1 LINK STATE 437
FIGURE 5-3: EXIT FROM L1 LINK STATE INITIATED BY UPSTREAM COMPONENT..... 440
FIGURE 5-4: CONCEPTUAL DIAGRAMS SHOWING TWO EXAMPLE CASES OF WAKE# ROUTING. 443
FIGURE 5-5: A CONCEPTUAL PME CONTROL STATE MACHINE 447
FIGURE 5-6: L1 TRANSITION SEQUENCE ENDING WITH A REJECTION (L0S ENABLED)..... 460
FIGURE 5-7: L1 SUCCESSFUL TRANSITION SEQUENCE 461
FIGURE 5-8: EXAMPLE OF L1 EXIT LATENCY COMPUTATION 463
FIGURE 5-9: STATE DIAGRAM FOR L1 PM SUBSTATES 468
FIGURE 5-10: DOWNSTREAM PORT WITH A SINGLE PLL 469
FIGURE 5-11: MULTIPLE DOWNSTREAM PORTS WITH A SHARED PLL 470
FIGURE 5-12: EXAMPLE: L1.1 WAVEFORMS ILLUSTRATING UPSTREAM PORT INITIATED EXIT ... 472
FIGURE 5-13: EXAMPLE: L1.1 WAVEFORMS ILLUSTRATING DOWNSTREAM PORT INITIATED EXIT 473
FIGURE 5-14: L1.2 SUBSTATES 474
FIGURE 5-15: EXAMPLE: ILLUSTRATION OF BOUNDARY CONDITION DUE TO DIFFERENT SAMPLING OF CLKREQ#..... 475
FIGURE 5-16: EXAMPLE: L1.2 WAVEFORMS ILLUSTRATING UPSTREAM PORT INITIATED EXIT ... 477
FIGURE 5-17: EXAMPLE: L1.2 WAVEFORMS ILLUSTRATING DOWNSTREAM PORT INITIATED EXIT 477
FIGURE 5-18: FUNCTION POWER MANAGEMENT STATE TRANSITIONS 480
FIGURE 5-19: NON-BRIDGE FUNCTION POWER MANAGEMENT DIAGRAM 481
FIGURE 5-20: PCI EXPRESS BRIDGE POWER MANAGEMENT DIAGRAM 486
FIGURE 6-1: ERROR CLASSIFICATION..... 501
FIGURE 6-2: FLOWCHART SHOWING SEQUENCE OF DEVICE ERROR SIGNALING AND LOGGING OPERATIONS 518
FIGURE 6-3: PSEUDO LOGIC DIAGRAM FOR ERROR MESSAGE CONTROLS 519
FIGURE 6-4: TC FILTERING EXAMPLE..... 538

FIGURE 6-5: TC TO VC MAPPING EXAMPLE 538
FIGURE 6-6: AN EXAMPLE OF TRAFFIC FLOW ILLUSTRATING INGRESS AND EGRESS 540
FIGURE 6-7: AN EXAMPLE OF DIFFERENTIATED TRAFFIC FLOW THROUGH A SWITCH 540
FIGURE 6-8: SWITCH ARBITRATION STRUCTURE 541
FIGURE 6-9: VC ID AND PRIORITY ORDER – AN EXAMPLE 543
FIGURE 6-10: MULTI-FUNCTION ARBITRATION MODEL 546
FIGURE 6-11: ROOT COMPLEX REPRESENTED AS A SINGLE COMPONENT 579
FIGURE 6-12: ROOT COMPLEX REPRESENTED AS MULTIPLE COMPONENTS 580
FIGURE 6-13: EXAMPLE SYSTEM TOPOLOGY WITH ARI DEVICES 593
FIGURE 6-14: SEGMENTATION OF THE MULTICAST ADDRESS RANGE 595
FIGURE 6-15: LATENCY FIELDS FORMAT FOR LTR MESSAGES 614
FIGURE 6-16: CLKREQ# AND CLOCK POWER MANAGEMENT 618
FIGURE 6-17: USE OF LTR AND CLOCK POWER MANAGEMENT 619
FIGURE 6-18: CODES AND EQUIVALENT WAKE# PATTERNS 621
FIGURE 6-19: EXAMPLE PLATFORM TOPOLOGY SHOWING A LINK WHERE OBFF IS CARRIED BY MESSAGES 622
FIGURE 6-20: PASID TLP PREFIX: 625
FIGURE 6-21: SAMPLE SYSTEM BLOCK DIAGRAM 629
FIGURE 6-22: LN PROTOCOL BASIC OPERATION 630
FIGURE 6-23: EXAMPLE SYSTEM TOPOLOGIES USING PTM 635
FIGURE 6-24: PRECISION TIME MEASUREMENT LINK PROTOCOL 636
FIGURE 6-25: PRECISION TIME MEASUREMENT EXAMPLE 637
FIGURE 6-26: PTM REQUESTER OPERATION 640
FIGURE 6-27: PTM TIMESTAMP CAPTURE EXAMPLE 643
FIGURE 6-28: EXAMPLE ILLUSTRATING APPLICATION OF ENHANCED ALLOCATION 647
FIGURE 6-29: EMERGENCY POWER REDUCTION STATE: EXAMPLE ADD-IN CARD 651
FIGURE 6-30: FPB HIGH LEVEL DIAGRAM AND EXAMPLE TOPOLOGY 656
FIGURE 6-31: EXAMPLE ILLUSTRATING “FLATTENING” OF A SWITCH 657
FIGURE 6-32: VECTOR MECHANISM FOR ADDRESS RANGE DECODING 657
FIGURE 6-33: RELATIONSHIP BETWEEN FPB AND NON-FPB DECODE MECHANISMS 658
FIGURE 6-34: ROUTING IDS (RIDS) AND SUPPORTED GRANULARITIES 660
FIGURE 6-35: ADDRESSES IN MEMORY BELOW 4 GB AND EFFECT OF GRANULARITY 662
FIGURE 6-36: VPD FORMAT 667
FIGURE 6-37: EXAMPLE NPEM CONFIGURATION USING A DOWNSTREAM PORT 672
FIGURE 6-38: EXAMPLE NPEM CONFIGURATION USING AN UPSTREAM PORT 672
FIGURE 6-39: NPEM COMMAND FLOW 673
FIGURE 7-1: PCI EXPRESS ROOT COMPLEX DEVICE MAPPING 676
FIGURE 7-2: PCI EXPRESS SWITCH DEVICE MAPPING 676
FIGURE 7-3: PCI EXPRESS CONFIGURATION SPACE LAYOUT 677
FIGURE 7-4: COMMON CONFIGURATION SPACE HEADER 688
FIGURE 7-5: COMMAND REGISTER 688
FIGURE 7-6: STATUS REGISTER 693
FIGURE 7-7: CLASS CODE REGISTER 696
FIGURE 7-8: HEADER TYPE REGISTER 697
FIGURE 7-9: BIST REGISTER 698
FIGURE 7-10: TYPE 0 CONFIGURATION SPACE HEADER 700

FIGURE 7-11: BASE ADDRESS REGISTER FOR MEMORY 701
FIGURE 7-12: BASE REGISTER FOR I/O 701
FIGURE 7-13: EXPANSION ROM BASE ADDRESS REGISTER LAYOUT 706
FIGURE 7-14: TYPE 1 CONFIGURATION SPACE HEADER 708
FIGURE 7-15: SECONDARY STATUS REGISTER 711
FIGURE 7-16: BRIDGE CONTROL REGISTER 715
FIGURE 7-17: POWER MANAGEMENT CAPABILITY STRUCTURE 717
FIGURE 7-18: POWER MANAGEMENT CAPABILITIES REGISTER 718
FIGURE 7-19: POWER MANAGEMENT CONTROL/STATUS REGISTER 721
FIGURE 7-20: PCI EXPRESS CAPABILITY STRUCTURE 725
FIGURE 7-21: PCI EXPRESS CAPABILITY LIST REGISTER 726
FIGURE 7-22: PCI EXPRESS CAPABILITIES REGISTER 726
FIGURE 7-23: DEVICE CAPABILITIES REGISTER 729
FIGURE 7-24: DEVICE CONTROL REGISTER 733
FIGURE 7-25: DEVICE STATUS REGISTER 740
FIGURE 7-26: LINK CAPABILITIES REGISTER 742
FIGURE 7-27: LINK CONTROL REGISTER 748
FIGURE 7-28: LINK STATUS REGISTER 756
FIGURE 7-29: SLOT CAPABILITIES REGISTER 759
FIGURE 7-30: SLOT CONTROL REGISTER 762
FIGURE 7-31: SLOT STATUS REGISTER 765
FIGURE 7-32: ROOT CONTROL REGISTER 767
FIGURE 7-33: ROOT CAPABILITIES REGISTER 768
FIGURE 7-34: ROOT STATUS REGISTER 769
FIGURE 7-35: DEVICE CAPABILITIES 2 REGISTER 770
FIGURE 7-36: DEVICE CONTROL 2 REGISTER 777
FIGURE 7-37: LINK CAPABILITIES 2 REGISTER 782
FIGURE 7-38: LINK CONTROL 2 REGISTER 786
FIGURE 7-39: LINK STATUS 2 REGISTER 791
FIGURE 7-40: PCI EXPRESS EXTENDED CONFIGURATION SPACE LAYOUT 796
FIGURE 7-41: PCI EXPRESS EXTENDED CAPABILITY HEADER 797
FIGURE 7-42: MSI CAPABILITY STRUCTURE FOR 32-BIT MESSAGE ADDRESS 798
FIGURE 7-43: MSI CAPABILITY STRUCTURE FOR 64-BIT MESSAGE ADDRESS 798
FIGURE 7-44: MSI CAPABILITY STRUCTURE FOR 32-BIT MESSAGE ADDRESS AND PVM 799
FIGURE 7-45: MSI CAPABILITY STRUCTURE FOR 64-BIT MESSAGE ADDRESS AND PVM 799
FIGURE 7-46: MSI CAPABILITY HEADER 800
FIGURE 7-47: MESSAGE CONTROL REGISTER FOR MSI 800
FIGURE 7-48: MESSAGE ADDRESS REGISTER FOR MSI 803
FIGURE 7-49: MESSAGE UPPER ADDRESS REGISTER FOR MSI 803
FIGURE 7-50: MESSAGE DATA REGISTER FOR MSI 804
FIGURE 7-51: EXTENDED MESSAGE DATA REGISTER FOR MSI (OPTIONAL) 804
FIGURE 7-52: MASK BITS REGISTER FOR MSI 805
FIGURE 7-53: PENDING BITS REGISTER FOR MSI 806
FIGURE 7-54: MSI-X CAPABILITY STRUCTURE 807
FIGURE 7-55: MSI-X TABLE STRUCTURE 807
FIGURE 7-56: MSI-X PBA STRUCTURE 807

FIGURE 7-57: MSI-X CAPABFPBILITY HEADER 809
FIGURE 7-58: MESSAGE CONTROL REGISTER FOR MSI-X 810
FIGURE 7-59: TABLE OFFSET/TABLE BIR REGISTER FOR MSI-X 810
FIGURE 7-60: PBA OFFSET/PBA BIR REGISTER FOR MSI-X 811
FIGURE 7-61: MESSAGE ADDRESS REGISTER FOR MSI-X TABLE ENTRIES 812
FIGURE 7-62: MESSAGE UPPER ADDRESS REGISTER FOR MSI-X TABLE ENTRIES 813
FIGURE 7-63: MESSAGE DATA REGISTER FOR MSI-X TABLE ENTRIES 813
FIGURE 7-64: VECTOR CONTROL REGISTER FOR MSI-X TABLE ENTRIES 814
FIGURE 7-65: PENDING BITS REGISTER FOR MSI-X PBA ENTRIES 814
FIGURE 7-66: SECONDARY PCI EXPRESS EXTENDED CAPABILITY STRUCTURE 815
FIGURE 7-67: SECONDARY PCI EXPRESS EXTENDED CAPABILITY HEADER 816
FIGURE 7-68: LINK CONTROL 3 REGISTER 816
FIGURE 7-69: LANE ERROR STATUS REGISTER 818
FIGURE 7-70: LANE EQUALIZATION CONTROL REGISTER 818
FIGURE 7-71: LANE EQUALIZATION CONTROL REGISTER ENTRY 819
FIGURE 7-72: DATA LINK FEATURE EXTENDED CAPABILITY 822
FIGURE 7-73: DATA LINK FEATURE EXTENDED CAPABILITY HEADER 822
FIGURE 7-74: DATA LINK FEATURE CAPABILITIES REGISTER 823
FIGURE 7-75: DATA LINK FEATURE STATUS REGISTER 824
FIGURE 7-76: PHYSICAL LAYER 16.0 GT/S EXTENDED CAPABILITY 825
FIGURE 7-77: 16.0 GT/S STATUS REGISTER 826
FIGURE 7-78: 16.0 GT/S LOCAL DATA PARITY MISMATCH STATUS REGISTER 828
FIGURE 7-79: 16.0 GT/S FIRST RETIMER DATA PARITY MISMATCH STATUS REGISTER 828
FIGURE 7-80: 16.0 GT/S SECOND RETIMER DATA PARITY MISMATCH STATUS REGISTER 829
FIGURE 7-81: HIGH LEVEL STRUCTURE OF 16.0 GT/S LANE EQUALIZATION CONTROL REGISTER 830
FIGURE 7-82: 16.0 GT/S LANE ((MAXIMUM LINK WIDTH – 1):0) EQUALIZATION CONTROL REGISTER ENTRY 830
FIGURE 7-83: MARGINING EXTENDED CAPABILITY 832
FIGURE 7-84: PHYSICAL LAYER 16.0 GT/S MARGINING EXTENDED CAPABILITY HEADER 833
FIGURE 7-85: MARGINING PORT CAPABILITIES REGISTER 833
FIGURE 7-86: MARGINING PORT STATUS REGISTER 834
FIGURE 7-87: LANE N: MARGINING LANE CONTROL REGISTER ENTRY 835
FIGURE 7-88: LANE N: MARGINING LANE STATUS REGISTER ENTRY 836
FIGURE 7-89: ACS EXTENDED CAPABILITY 837
FIGURE 7-90: ACS EXTENDED CAPABILITY HEADER 837
FIGURE 7-91: ACS CAPABILITY REGISTER 838
FIGURE 7-92: ACS CONTROL REGISTER 839
FIGURE 7-93: EGRESS CONTROL VECTOR REGISTER 841
FIGURE 7-94: PCI EXPRESS POWER BUDGETING CAPABILITY STRUCTURE 842
FIGURE 7-95: POWER BUDGETING EXTENDED CAPABILITY HEADER 843
FIGURE 7-96: POWER BUDGETING DATA REGISTER 844
FIGURE 7-97: POWER BUDGET CAPABILITY REGISTER 846
FIGURE 7-98: LTR EXTENDED CAPABILITY STRUCTURE 847
FIGURE 7-99: LTR EXTENDED CAPABILITY HEADER 847
FIGURE 7-100: MAX SNOOP LATENCY REGISTER 848

FIGURE 7-101: MAX NO-SNOOP LATENCY REGISTER 849
FIGURE 7-102: L1 PM SUBSTATES CAPABILITY 850
FIGURE 7-103: L1 PM SUBSTATES EXTENDED CAPABILITY HEADER 850
FIGURE 7-104: L1 PM SUBSTATES CAPABILITIES REGISTER 851
FIGURE 7-105: L1 PM SUBSTATES CONTROL 1 REGISTER 853
FIGURE 7-106: L1 PM SUBSTATES CONTROL 2 REGISTER 855
FIGURE 7-107: PCI EXPRESS ADVANCED ERROR REPORTING EXTENDED CAPABILITY STRUCTURE 857
FIGURE 7-108: ADVANCED ERROR REPORTING EXTENDED CAPABILITY HEADER 857
FIGURE 7-109: UNCORRECTABLE ERROR STATUS REGISTER 859
FIGURE 7-110: UNCORRECTABLE ERROR MASK REGISTER 861
FIGURE 7-111: UNCORRECTABLE ERROR SEVERITY REGISTER 863
FIGURE 7-112: CORRECTABLE ERROR STATUS REGISTER 865
FIGURE 7-113: CORRECTABLE ERROR MASK REGISTER 866
FIGURE 7-114: ADVANCED ERROR CAPABILITIES AND CONTROL REGISTER 867
FIGURE 7-115: HEADER LOG REGISTER 868
FIGURE 7-116: ROOT ERROR COMMAND REGISTER 869
FIGURE 7-117: ROOT ERROR STATUS REGISTER 871
FIGURE 7-118: ERROR SOURCE IDENTIFICATION REGISTER 872
FIGURE 7-119: TLP PREFIX LOG REGISTER 874
FIGURE 7-120: FIRST DW OF ENHANCED ALLOCATION CAPABILITY 874
FIGURE 7-121: FIRST DW OF EACH ENTRY FOR ENHANCED ALLOCATION CAPABILITY 875
FIGURE 7-122: FORMAT OF ENTRY FOR ENHANCED ALLOCATION CAPABILITY 878
FIGURE 7-123: EXAMPLE ENTRY WITH 64B BASE AND 64B MAXOFFSET 880
FIGURE 7-124: EXAMPLE ENTRY WITH 64B BASE AND 32B MAXOFFSET 880
FIGURE 7-125: EXAMPLE ENTRY WITH 32B BASE AND 64B MAXOFFSET 881
FIGURE 7-126: EXAMPLE ENTRY WITH 32B BASE AND 32B MAXOFFSET 881
FIGURE 7-127: RESIZABLE BAR CAPABILITY 883
FIGURE 7-128: RESIZABLE BAR EXTENDED CAPABILITY HEADER 883
FIGURE 7-129: RESIZABLE BAR CAPABILITY REGISTER 884
FIGURE 7-130: RESIZABLE BAR CONTROL REGISTER 885
FIGURE 7-131: ARI CAPABILITY 887
FIGURE 7-132: ARI CAPABILITY HEADER 888
FIGURE 7-133: ARI CAPABILITY REGISTER 888
FIGURE 7-134: ARI CONTROL REGISTER 889
FIGURE 7-135: PASID EXTENDED CAPABILITY STRUCTURE 890
FIGURE 7-136: PASID EXTENDED CAPABILITY HEADER 890
FIGURE 7-137: PASID CAPABILITY REGISTER 891
FIGURE 7-138: PASID CONTROL REGISTER 892
FIGURE 7-139: FRS EXTENDED CAPABILITY 893
FIGURE 7-140: FRS QUEUING EXTENDED CAPABILITY HEADER 893
FIGURE 7-141: FRS QUEUING CAPABILITY REGISTER 894
FIGURE 7-142: FRS QUEUEING STATUS REGISTER 895
FIGURE 7-143: FRS QUEUEING CONTROL REGISTER 895
FIGURE 7-144: FRS MESSAGE QUEUE REGISTER 896
FIGURE 7-145: FPB CAPABILITY STRUCTURE 897

FIGURE 7-146: FPB CAPABILITY HEADER 897
FIGURE 7-147: FPB CAPABILITIES REGISTER 898
FIGURE 7-148: FPB RID VECTOR CONTROL 1 REGISTER 900
FIGURE 7-149: FPB RID VECTOR CONTROL 2 REGISTER 901
FIGURE 7-150: FPB MEM LOW VECTOR CONTROL REGISTER 902
FIGURE 7-151: FPB MEM HIGH VECTOR CONTROL 1 REGISTER 904
FIGURE 7-152: FPB MEM HIGH VECTOR CONTROL 2 REGISTER 905
FIGURE 7-153: FPB VECTOR ACCESS CONTROL REGISTER 906
FIGURE 7-154: FPB VECTOR ACCESS DATA REGISTER 908
FIGURE 7-155: PCI EXPRESS VIRTUAL CHANNEL CAPABILITY STRUCTURE 909
FIGURE 7-156: VIRTUAL CHANNEL EXTENDED CAPABILITY HEADER 910
FIGURE 7-157: PORT VC CAPABILITY REGISTER 1 911
FIGURE 7-158: PORT VC CAPABILITY REGISTER 2 912
FIGURE 7-159: PORT VC CONTROL REGISTER 913
FIGURE 7-160: PORT VC STATUS REGISTER 914
FIGURE 7-161: VC RESOURCE CAPABILITY REGISTER 915
FIGURE 7-162: VC RESOURCE CONTROL REGISTER 917
FIGURE 7-163: VC RESOURCE STATUS REGISTER 919
FIGURE 7-164: EXAMPLE VC ARBITRATION TABLE WITH 32 PHASES 921
FIGURE 7-165: EXAMPLE PORT ARBITRATION TABLE WITH 128 PHASES AND 2-BIT TABLE ENTRIES 922
FIGURE 7-166: PCI EXPRESS MFVC CAPABILITY STRUCTURE 924
FIGURE 7-167: MFVC EXTENDED CAPABILITY HEADER 925
FIGURE 7-168: PORT VC CAPABILITY REGISTER 1 926
FIGURE 7-169: PORT VC CAPABILITY REGISTER 2 927
FIGURE 7-170: PORT VC CONTROL REGISTER 928
FIGURE 7-171: PORT VC STATUS REGISTER 929
FIGURE 7-172: VC RESOURCE CAPABILITY REGISTER 929
FIGURE 7-173: VC RESOURCE CONTROL REGISTER 931
FIGURE 7-174: VC RESOURCE STATUS REGISTER 933
FIGURE 7-175: PCI EXPRESS DEVICE SERIAL NUMBER CAPABILITY STRUCTURE 936
FIGURE 7-176: DEVICE SERIAL NUMBER EXTENDED CAPABILITY HEADER 936
FIGURE 7-177: SERIAL NUMBER REGISTER 937
FIGURE 7-178: VENDOR-SPECIFIC CAPABILITY 938
FIGURE 7-179: PCI EXPRESS VSEC STRUCTURE 939
FIGURE 7-180: VENDOR-SPECIFIC EXTENDED CAPABILITY HEADER 939
FIGURE 7-181: VENDOR-SPECIFIC HEADER 940
FIGURE 7-182: DESIGNATED VENDOR-SPECIFIC EXTENDED CAPABILITY 941
FIGURE 7-183: DESIGNATED VENDOR-SPECIFIC EXTENDED CAPABILITY HEADER 941
FIGURE 7-184: DESIGNATED VENDOR-SPECIFIC HEADER 1 942
FIGURE 7-185: DESIGNATED VENDOR-SPECIFIC HEADER 2 943
FIGURE 7-186: ROOT COMPLEX FEATURES CAPABILITY STRUCTURE 943
FIGURE 7-187: RCRB HEADER EXTENDED CAPABILITY HEADER 944
FIGURE 7-188: VENDOR ID AND DEVICE ID 944
FIGURE 7-189: RCRB CAPABILITIES 945
FIGURE 7-190: RCRB CONTROL 945

FIGURE 7-191: PCI EXPRESS ROOT COMPLEX LINK DECLARATION CAPABILITY 947
FIGURE 7-192: ROOT COMPLEX LINK DECLARATION EXTENDED CAPABILITY HEADER 947
FIGURE 7-193: ELEMENT SELF DESCRIPTION REGISTER 948
FIGURE 7-194: LINK ENTRY 949
FIGURE 7-195: LINK DESCRIPTION REGISTER 949
FIGURE 7-196: LINK ADDRESS FOR LINK TYPE 0 951
FIGURE 7-197: LINK ADDRESS FOR LINK TYPE 1 952
FIGURE 7-198: ROOT COMPLEX INTERNAL LINK CONTROL CAPABILITY 952
FIGURE 7-199: ROOT INTERNAL LINK CONTROL EXTENDED CAPABILITY HEADER 953
FIGURE 7-200: ROOT COMPLEX LINK CAPABILITIES REGISTER 954
FIGURE 7-201: ROOT COMPLEX LINK CONTROL REGISTER 957
FIGURE 7-202: ROOT COMPLEX LINK STATUS REGISTER 959
FIGURE 7-203: ROOT COMPLEX EVENT COLLECTOR ENDPOINT ASSOCIATION CAPABILITY 960
FIGURE 7-204: ROOT COMPLEX EVENT COLLECTOR ENDPOINT ASSOCIATION EXTENDED CAPABILITY HEADER 961
FIGURE 7-205: MULTICAST EXTENDED CAPABILITY STRUCTURE 962
FIGURE 7-206: MULTICAST EXTENDED CAPABILITY HEADER 962
FIGURE 7-207: MULTICAST CAPABILITY REGISTER 963
FIGURE 7-208: MULTICAST CONTROL REGISTER 964
FIGURE 7-209: MC_BASE_ADDRESS REGISTER 965
FIGURE 7-210: MC_RECEIVE REGISTER 965
FIGURE 7-211: MC_BLOCK_ALL REGISTER 966
FIGURE 7-212: MC_BLOCK_UNTRANSLATED REGISTER 966
FIGURE 7-213: MC_OVERLAY_BAR 967
FIGURE 7-214: DYNAMIC POWER ALLOCATION CAPABILITY STRUCTURE 968
FIGURE 7-215: DPA EXTENDED CAPABILITY HEADER 968
FIGURE 7-216: DPA CAPABILITY REGISTER 969
FIGURE 7-217: DPA LATENCY INDICATOR REGISTER 970
FIGURE 7-218: DPA STATUS REGISTER 970
FIGURE 7-219: DPA CONTROL REGISTER 971
FIGURE 7-220: DPA POWER ALLOCATION ARRAY 971
FIGURE 7-221: TPH EXTENDED CAPABILITY STRUCTURE 972
FIGURE 7-222: TPH REQUESTER EXTENDED CAPABILITY HEADER 972
FIGURE 7-223: TPH REQUESTER CAPABILITY REGISTER 973
FIGURE 7-224: TPH REQUESTER CONTROL REGISTER 974
FIGURE 7-225: TPH ST TABLE 975
FIGURE 7-226: LNR EXTENDED CAPABILITY 976
FIGURE 7-227: LNR EXTENDED CAPABILITY HEADER 976
FIGURE 7-228: LNR CAPABILITY REGISTER 977
FIGURE 7-229: LNR CONTROL REGISTER 977
FIGURE 7-230: DPC EXTENDED CAPABILITY 979
FIGURE 7-231: DPC EXTENDED CAPABILITY HEADER 979
FIGURE 7-232: DPC CAPABILITY REGISTER 980
FIGURE 7-233: DPC CONTROL REGISTER 981
FIGURE 7-234: DPC STATUS REGISTER 983
FIGURE 7-235: DPC ERROR SOURCE ID REGISTER 985

FIGURE 7-236: RP PIO STATUS REGISTER 985
FIGURE 7-237: RP PIO MASK REGISTER 986
FIGURE 7-238: RP PIO SEVERITY REGISTER 987
FIGURE 7-239: RP PIO SYSERROR REGISTER 988
FIGURE 7-240: RP PIO EXCEPTION REGISTER 989
FIGURE 7-241: RP PIO HEADER LOG REGISTER 990
FIGURE 7-242: RP PIO IMPSPEC LOG REGISTER 991
FIGURE 7-243: RP PIO TLP PREFIX LOG REGISTER 991
FIGURE 7-244: PTM EXTENDED CAPABILITY STRUCTURE 992
FIGURE 7-245: PTM EXTENDED CAPABILITY HEADER 992
FIGURE 7-246: PTM CAPABILITY REGISTER 993
FIGURE 7-247: PTM CONTROL REGISTER 994
FIGURE 7-248: READINESS TIME REPORTING EXTENDED CAPABILITY 996
FIGURE 7-249: READINESS TIME ENCODING 997
FIGURE 7-250: READINESS TIME REPORTING EXTENDED CAPABILITY HEADER 997
FIGURE 7-251: READINESS TIME REPORTING 1 REGISTER 998
FIGURE 7-252: READINESS TIME REPORTING 2 REGISTER 999
FIGURE 7-253: HIERARCHY ID EXTENDED CAPABILITY 1001
FIGURE 7-254: HIERARCHY ID EXTENDED CAPABILITY HEADER 1001
FIGURE 7-255: HIERARCHY ID STATUS REGISTER 1002
FIGURE 7-256: HIERARCHY ID DATA REGISTER 1004
FIGURE 7-257: HIERARCHY ID GUID 1 REGISTER 1005
FIGURE 7-258: HIERARCHY ID GUID 2 REGISTER 1005
FIGURE 7-259: HIERARCHY ID GUID 3 REGISTER 1006
FIGURE 7-260: HIERARCHY ID GUID 4 REGISTER 1007
FIGURE 7-261: HIERARCHY ID GUID 5 REGISTER 1007
FIGURE 7-262: VPD CAPABILITY STRUCTURE 1008
FIGURE 7-263: NPEM EXTENDED CAPABILITY 1010
FIGURE 7-264: NPEM EXTENDED CAPABILITY HEADER 1010
FIGURE 7-265: NPEM CAPABILITY REGISTER 1011
FIGURE 7-266: NPEM CAPABILITY REGISTER 1011
FIGURE 7-267: NPEM CONTROL REGISTER 1013
FIGURE 7-268: NPEM STATUS REGISTER 1015
FIGURE 8-1: TX TEST BOARD FOR NON-EMBEDDED REFCLK 1017
FIGURE 8-2: TX TEST BOARD FOR EMBEDDED REFCLK 1018
FIGURE 8-3: SINGLE-ENDED AND DIFFERENTIAL LEVELS 1019
FIGURE 8-4: TX EQUALIZATION FIR REPRESENTATION 1021
FIGURE 8-5: DEFINITION OF TX VOLTAGE LEVELS AND EQUALIZATION RATIOS 1022
FIGURE 8-6: WAVEFORM MEASUREMENT POINTS FOR PRE-SHOOT 1023
FIGURE 8-7: WAVEFORM MEASUREMENT POINTS FOR DE-EMPHASIS 1023
FIGURE 8-8: $V_{TX\text{-}DIFF\text{-}PP}$ AND $V_{TX\text{-}DIFF\text{-}PP\text{-}LOW}$ MEASUREMENT 1025
FIGURE 8-9: MEASURING $V_{TX\text{-}EIEOS\text{-}FS}$ AND $V_{TX\text{-}EIEOS\text{-}RS}$ AT 8.0 GT/S 1027
FIGURE 8-10: COMPLIANCE PATTERN AND RESULTING PACKAGE LOSS TEST WAVEFORM 1028
FIGURE 8-11: 2.5 AND 5.0 GT/S TRANSMITTER MARGINING VOLTAGE LEVELS AND CODES 1029
FIGURE 8-12: FIRST ORDER CC BEHAVIORAL CDR TRANSFER FUNCTIONS 1032

FIGURE 8-13: 2ND ORDER BEHAVIORAL SRIS CDR TRANSFER FUNCTIONS FOR 2.5 GT/S AND 5.0 GT/S 1032
FIGURE 8-14: BEHAVIORAL SRIS CDR FUNCTIONS FOR 8.0 AND 16.0 GT/S 1033
FIGURE 8-15: RELATION BETWEEN DATA EDGE PDFS AND RECOVERED DATA CLOCK 1035
FIGURE 8-16: DERIVATION OF T_{TX-UTJ} AND $T_{TX-UDJDD}$ 1035
FIGURE 8-17: PWJ RELATIVE TO CONSECUTIVE EDGES 1 UI APART 1036
FIGURE 8-18: DEFINITION OF $T_{TX-UPW-DJDD}$ AND $T_{TX-UPW-TJ}$ DATA RATE DEPENDENT TRANSMITTER PARAMETERS 1037
FIGURE 8-19: TX, RX DIFFERENTIAL RETURN LOSS MASK 1040
FIGURE 8-20: TX, RX COMMON MODE RETURN LOSS MASK 1041
FIGURE 8-21: RX TESTBOARD TOPOLOGY FOR 16.0 GT/S 1044
FIGURE 8-22: CALIBRATION CHANNEL IL MASK EXCLUDING RX PACKAGE 1045
FIGURE 8-23: EXAMPLE 16 GT/S CALIBRATION CHANNEL 1047
FIGURE 8-24: STACKUP FOR EXAMPLE 16 GT/S CALIBRATION CHANNEL 1048
FIGURE 8-25: CEM CONNECTOR DRILL HOLE PAD STACK 1048
FIGURE 8-26: PAD STACK FOR SMA DRILL HOLES 1049
FIGURE 8-27: TRANSFER FUNCTION FOR 8.0 GT/S BEHAVIORAL CTLE 1051
FIGURE 8-28: LOSS CURVES FOR 8.0 GT/S BEHAVIORAL CTLE 1051
FIGURE 8-29: LOSS CURVES FOR 16.0 GT/S BEHAVIORAL CTLE 1052
FIGURE 8-30: VARIABLES DEFINITION AND DIAGRAM FOR 1-TAP DFE 1053
FIGURE 8-31: DIAGRAM FOR 2-TAP DFE 1053
FIGURE 8-32: LAYOUT FOR CALIBRATING THE STRESSED JITTER EYE AT 8.0 GT/S 1055
FIGURE 8-33: LAYOUT FOR CALIBRATING THE STRESSED JITTER EYE AT 16.0 GT/S 1056
FIGURE 8-34: SJ MASK FOR RECEIVERS OPERATING IN IR MODE AT 8.0 GT/S 1059
FIGURE 8-35: SJ MASK FOR RECEIVERS OPERATING IN IR MODE AT 16.0 GT/S 1059
FIGURE 8-36: SJ MASKS FOR RECEIVERS OPERATING IN CC MODE AT 8.0 GT/S AND 16.0 GT/S 1060
FIGURE 8-37: LAYOUT FOR JITTER TESTING COMMON REFCLK RX AT 16.0 GT/S 1061
FIGURE 8-38: LAYOUT FOR JITTER TESTING FOR INDEPENDENT REFCLK RX AT 16.0 GT/S 1061
FIGURE 8-39: EXIT FROM IDLE VOLTAGE AND TIME MARGINS 1064
FIGURE 8-40: ALLOWED RANGES FOR MAXIMUM TIMING AND VOLTAGE MARGINS 1065
FIGURE 8-41 : FLOW DIAGRAM FOR CHANNEL TOLERANCING AT 2.5 AND 5.0 GT/S 1070
FIGURE 8-42: FLOW DIAGRAM FOR CHANNEL TOLERANCING AT 8.0 AND 16.0 GT/S 1071
FIGURE 8-43: TX/RX BEHAVIORAL PACKAGE MODELS 1072
FIGURE 8-44: BEHAVIORAL TX AND RX S-PORT DESIGNATION 1072
FIGURE 8-45: SDD21 PLOTS FOR ROOT AND NON-ROOT PACKAGES 1073
FIGURE 8-46: DERIVATION OF 8.0 GT/S JITTER PARAMETERS FOR TABLE 8-14 1075
FIGURE 8-47: EH, EW MASK 1077
FIGURE 8-48: REFCLK TEST SETUP 1080
FIGURE 8-49: SINGLE-ENDED MEASUREMENT POINTS FOR ABSOLUTE CROSS POINT AND SWING 1081
FIGURE 8-50: SINGLE-ENDED MEASUREMENT POINTS FOR DELTA CROSS POINT 1082
FIGURE 8-51: SINGLE-ENDED MEASUREMENT POINTS FOR RISE AND FALL TIME MATCHING 1082
FIGURE 8-52: DIFFERENTIAL MEASUREMENT POINTS FOR DUTY CYCLE AND PERIOD 1082
FIGURE 8-53: DIFFERENTIAL MEASUREMENT POINTS FOR RISE AND FALL TIME 1082
FIGURE 8-54: DIFFERENTIAL MEASUREMENT POINTS FOR RINGBACK 1083

FIGURE 8-55: LIMITS FOR PHASE JITTER FROM THE REFERENCE 1084
FIGURE 8-56: 5 MHZ PLL TRANSFER FUNCTION EXAMPLE 1085
FIGURE 8-57: COMMON REFCLK RX ARCHITECTURE 1086
FIGURE 9-1: GENERIC PLATFORM CONFIGURATION 1090
FIGURE 9-2: GENERIC PLATFORM CONFIGURATION WITH A VI AND MULTIPLE SI 1091
FIGURE 9-3: GENERIC PLATFORM CONFIGURATION WITH SR-IOV AND IOV ENABLERS 1093
FIGURE 9-4: EXAMPLE MULTI-FUNCTION DEVICE 1095
FIGURE 9-5: EXAMPLE SR-IOV SINGLE PF CAPABLE DEVICE 1096
FIGURE 9-6: EXAMPLE SR-IOV MULTI-PF CAPABLE DEVICE 1098
FIGURE 9-7: EXAMPLE SR-IOV DEVICE WITH MULTIPLE BUS NUMBERS 1100
FIGURE 9-8: EXAMPLE SR-IOV DEVICE WITH A MIXTURE OF FUNCTION TYPES 1101
FIGURE 9-9: I/O VIRTUALIZATION INTEROPERABILITY 1102
FIGURE 9-10: BAR SPACE EXAMPLE FOR SINGLE BAR DEVICE 1104
FIGURE 9-11: INITIAL VF MIGRATION STATE ARRAY 1109
FIGURE 9-12: VF MIGRATION STATE DIAGRAM 1110
FIGURE 9-13: SR-IOV EXTENDED CAPABILITY 1113
FIGURE 9-14: TYPE 0 CONFIGURATION SPACE HEADER 1131
FIGURE 9-15: LINK CONTROL REGISTER 1137
FIGURE 9-16: VF RESIZABLE BAR CAPABILITY 1145
FIGURE 9-17: VF RESIZABLE BAR EXTENDED CAPABILITY HEADER 1146
FIGURE 9-18: VF RESIZABLE BAR CONTROL REGISTER 1147
FIGURE 9-19: MSI-X CAPABILITY 1159
FIGURE 10-1: EXAMPLE ILLUSTRATING A PLATFORM WITH TA, ATPT, AND ATC ELEMENTS . 1164
FIGURE 10-2: EXAMPLE ATS TRANSLATION REQUEST/COMPLETION EXCHANGE 1165
FIGURE 10-3: EXAMPLE MULTI-FUNCTION DEVICE WITH ATC PER FUNCTION 1168
FIGURE 10-4: INVALIDATION PROTOCOL WITH A SINGLE INVALIDATION REQUEST AND COMPLETION 1169
FIGURE 10-5: SINGLE INVALIDATE REQUEST WITH MULTIPLE INVALIDATE COMPLETIONS 1170
FIGURE 10-6: MEMORY REQUEST HEADER WITH 64-BIT ADDRESS 1173
FIGURE 10-7: MEMORY REQUEST HEADER WITH 32-BIT ADDRESS 1173
FIGURE 10-8: 64-BIT TRANSLATION REQUEST HEADER 1174
FIGURE 10-9: 32-BIT TRANSLATION REQUEST HEADER 1175
FIGURE 10-10: TRANSLATION COMPLETION WITH NO DATA 1177
FIGURE 10-11: SUCCESSFUL TRANSLATION COMPLETION 1178
FIGURE 10-12: TRANSLATION COMPLETION DATA ENTRY 1179
FIGURE 10-13: INVALIDATE REQUEST MESSAGE 1187
FIGURE 10-14: INVALIDATE REQUEST MESSAGE BODY 1188
FIGURE 10-15: INVALIDATE COMPLETION MESSAGE FORMAT 1189
FIGURE 10-16: PAGE REQUEST MESSAGE 1196
FIGURE 10-17: STOP MARKER MESSAGE 1199
FIGURE 10-18: PRG RESPONSE MESSAGE 1201
FIGURE 10-19: ATS EXTENDED CAPABILITY STRUCTURE 1203
FIGURE 10-20: ATS EXTENDED CAPABILITY HEADER 1203
FIGURE 10-21: ATS CAPABILITY REGISTER 1204
FIGURE 10-22: ATS CONTROL REGISTER 1204
FIGURE 10-23: PAGE REQUEST EXTENDED CAPABILITY STRUCTURE 1205

FIGURE 10-24: PAGE REQUEST EXTENDED CAPABILITY HEADER 1206
FIGURE 10-25: PAGE REQUEST CONTROL REGISTER 1206
FIGURE 10-26: PAGE REQUEST STATUS REGISTER 1207
FIGURE A-1: AN EXAMPLE SHOWING ENDPOINT-TO-ROOT-COMPLEX AND PEER-TO-PEER COMMUNICATION MODELS 1213
FIGURE A-2: TWO BASIC BANDWIDTH RESOURCING PROBLEMS: OVER-SUBSCRIPTION AND CONGESTION 1214
FIGURE A-3: A SIMPLIFIED EXAMPLE ILLUSTRATING PCI EXPRESS ISOCHRONOUS PARAMETERS 1219
FIGURE C-1: SCRAMBLING SPECTRUM AT 2.5 GT/S FOR DATA VALUE OF 0 1238
FIGURE E-1: REFERENCE TOPOLOGY FOR IDO USE 1245
FIGURE G-1: DEVICE AND PROCESSOR CONNECTED USING A PMUX LINK 1253
FIGURE G-2: PMUX LINK 1254
FIGURE G-3: PMUX PACKET FLOW THROUGH THE LAYERS 1255
FIGURE G-4: PMUX PACKET 1261
FIGURE G-5: TLP AND PMUX PACKET FRAMING (8B/10B ENCODING) 1262
FIGURE G-6: TLP AND PMUX PACKET FRAMING (128B/130B ENCODING) 1264
FIGURE G-7: PMUX EXTENDED CAPABILITY 1268
FIGURE G-8: PMUX EXTENDED CAPABILITY HEADER 1268
FIGURE G-9: PMUX CAPABILITY REGISTER 1269
FIGURE G-10: PMUX CONTROL REGISTER 1271
FIGURE G-11: PMUX STATUS REGISTER 1272
FIGURE G-12: PMUX PROTOCOL ARRAY ENTRY 1275

Tables

TABLE 2-1: TRANSACTION TYPES FOR DIFFERENT ADDRESS SPACES 70
TABLE 2-2: FMT[2:0] FIELD VALUES 75
TABLE 2-3: FMT[2:0] AND TYPE[4:0] FIELD ENCODINGS 75
TABLE 2-4: LENGTH[9:0] FIELD ENCODING 77
TABLE 2-5: ADDRESS FIELD MAPPING 82
TABLE 2-6: HEADER FIELD LOCATIONS FOR NON-ARI ID ROUTING 83
TABLE 2-7: HEADER FIELD LOCATIONS FOR ARI ID ROUTING 84
TABLE 2-8: BYTE ENABLES LOCATION AND CORRESPONDENCE 86
TABLE 2-9: ORDERING ATTRIBUTES 95
TABLE 2-10: CACHE COHERENCY MANAGEMENT ATTRIBUTE 96
TABLE 2-11: DEFINITION OF TC FIELD ENCODINGS 96
TABLE 2-12: LENGTH FIELD VALUES FOR ATOMICOP REQUESTS 97
TABLE 2-13: TPH TLP PREFIX BIT MAPPING 101
TABLE 2-14: LOCATION OF PH[1:0] IN TLP HEADER 102
TABLE 2-15: PROCESSING HINT ENCODING 102
TABLE 2-16: LOCATION OF ST[7:0] IN TLP HEADERS 103
TABLE 2-17: MESSAGE ROUTING 105
TABLE 2-18: INTX MECHANISM MESSAGES 107
TABLE 2-19: BRIDGE MAPPING FOR INTX VIRTUAL WIRES 109
TABLE 2-20: POWER MANAGEMENT MESSAGES 111
TABLE 2-21: ERROR SIGNALING MESSAGES 112
TABLE 2-22: UNLOCK MESSAGE 112
TABLE 2-23: SET_SLOT_POWER_LIMIT MESSAGE 113
TABLE 2-24: VENDOR_DEFINED MESSAGES 114
TABLE 2-25: NOTIFICATION REASON (NR) FIELD ENCODINGS 117
TABLE 2-26: LN MESSAGES 117
TABLE 2-27: DRS MESSAGE 118
TABLE 2-28: FRS MESSAGE 120
TABLE 2-29: HIERARCHY ID MESSAGE 121
TABLE 2-30: IGNORED MESSAGES 122
TABLE 2-31: LTR MESSAGE 122
TABLE 2-32: OBFF MESSAGE 123
TABLE 2-33: PRECISION TIME MEASUREMENT MESSAGES 124
TABLE 2-34: COMPLETION STATUS FIELD VALUES 126
TABLE 2-35: LOCAL TLP PREFIX TYPES 129
TABLE 2-36: END-END TLP PREFIX TYPES 130
TABLE 2-37: CALCULATING BYTE COUNT FROM LENGTH AND BYTE ENABLES 145
TABLE 2-38: CALCULATING LOWER ADDRESS FROM 1ST DW BE 146
TABLE 2-39: ORDERING RULES SUMMARY 151
TABLE 2-40: TC TO VC MAPPING EXAMPLE 158
TABLE 2-41: FLOW CONTROL CREDIT TYPES 162
TABLE 2-42: TLP FLOW CONTROL CREDIT CONSUMPTION 162
TABLE 2-43: MINIMUM INITIAL FLOW CONTROL ADVERTISEMENTS 164

TABLE 2-44: [FIELD SIZE] VALUES 166
TABLE 2-45: MAXIMUM UPDATEFC TRANSMISSION LATENCY GUIDELINES FOR 2.5 GT/S (SYMBOL TIMES) 172
TABLE 2-46: MAXIMUM UPDATEFC TRANSMISSION LATENCY GUIDELINES FOR 5.0 GT/S (SYMBOL TIMES) 172
TABLE 2-47: MAXIMUM UPDATEFC TRANSMISSION LATENCY GUIDELINES FOR 8.0 GT/S (SYMBOL TIMES) 173
TABLE 2-48: MAXIMUM UPDATEFC TRANSMISSION LATENCY GUIDELINES FOR 16.0 GT/S (SYMBOL TIMES) 173
TABLE 2-49: MAPPING OF BITS INTO ECRC FIELD 175
TABLE 3-1: DATA LINK FEATURE SUPPORTED BIT DEFINITION 193
TABLE 3-2: SCALED FLOW CONTROL SCALING FACTORS 198
TABLE 3-3: DLLP TYPE ENCODINGS 199
TABLE 3-4: HDRSCALE AND DATASCALE ENCODINGS 200
TABLE 3-5: MAPPING OF BITS INTO CRC FIELD 203
TABLE 3-6: MAPPING OF BITS INTO LCRC FIELD 207
TABLE 3-7: MAXIMUM ACK LATENCY LIMITS FOR 2.5 GT/S (SYMBOL TIMES) 222
TABLE 3-8: MAXIMUM ACK LATENCY LIMITS FOR 5.0 GT/S (SYMBOL TIMES) 222
TABLE 3-9: MAXIMIM ACK LATENCY LIMITS FOR 8.0 GT/S (SYMBOL TIMES) 222
TABLE 3-10: MAXIMUM ACK LATENCY LIMITS FOR 16.0 GT/S (SYMBOL TIMES) 223
TABLE 4-1: SPECIAL SYMBOLS 228
TABLE 4-2: FRAMING TOKEN ENCODING 238
TABLE 4-3: TRANSMITTER PRESET ENCODING 263
TABLE 4-4: RECEIVER PRESET HINT ENCODING FOR 8.0 GT/S 263
TABLE 4-5: TS1 ORDERED SET 266
TABLE 4-6: TS2 ORDERED SET 269
TABLE 4-7: ELECTRICAL IDLE ORDERED SET (EIOS) FOR 2.5 GT/S AND 5.0 GT/S DATA RATES 272
TABLE 4-8: ELECTRICAL IDLE ORDERED SET (EIOS) FOR 8.0 GT/S AND ABOVE DATA RATES .. 272
TABLE 4-9: ELECTRICAL IDLE EXIT ORDERED SET (EIEOS) FOR 5.0 GT/S DATA RATE 272
TABLE 4-10: ELECTRICAL IDLE EXIT ORDERED SET (EIEOS) FOR 8.0 GT/S DATA RATES 272
TABLE 4-11: ELECTRICAL IDLE EXIT ORDERED SET (EIEOS) FOR 16.0 GT/S DATA RATE 272
TABLE 4-12: ELECTRICAL IDLE INFERENCE CONDITIONS 275
TABLE 4-13: FTS FOR 8.0 GT/S AND ABOVE DATA RATES 278
TABLE 4-14: SDS ORDERED SET (FOR 8.0 GT/S AND ABOVE DATA RATE) 279
TABLE 4-15: LINK STATUS MAPPED TO THE LTSSM 287
TABLE 4-16: STANDARD SKP ORDERED SET WITH 128B/130B ENCODING 364
TABLE 4-17: CONTROL SKP ORDERED SET WITH 128B/130B ENCODING 364
TABLE 4-18: MARGIN COMMAND RELATED FIELDS IN THE CONTROL SKP ORDERED SET 375
TABLE 4-19: MARGIN COMMANDS AND CORRESPONDING RESPONSES 377
TABLE 4-20: MAXIMUM RETIMER EXIT LATENCY 396
TABLE 4-21: INFERRING ELECTRICAL IDLE 397
TABLE 4-22. RETIMER LATENCY LIMIT NOT SRIS (SYMBOL TIMES) 416
TABLE 4-23: RETIMER LATENCY LIMIT SRIS (SYMBOL TIMES) 417
TABLE 5-1: SUMMARY OF PCI EXPRESS LINK POWER MANAGEMENT STATES 430
TABLE 5-2: RELATION BETWEEN POWER MANAGEMENT STATES OF LINK AND COMPONENTS .. 436
TABLE 5-3: ENCODING OF THE ASPM SUPPORT FIELD 464

TABLE 5-4: DESCRIPTION OF THE SLOT CLOCK CONFIGURATION BIT 464
TABLE 5-5: DESCRIPTION OF THE COMMON CLOCK CONFIGURATION BIT 464
TABLE 5-6: ENCODING OF THE L0S EXIT LATENCY FIELD 465
TABLE 5-7: ENCODING OF THE L1 EXIT LATENCY FIELD 465
TABLE 5-8: ENCODING OF THE ENDPOINT L0S ACCEPTABLE LATENCY FIELD 465
TABLE 5-9: ENCODING OF THE ENDPOINT L1 ACCEPTABLE LATENCY FIELD 465
TABLE 5-10: ENCODING OF THE ASPM CONTROL FIELD 466
TABLE 5-11: TIMING PARAMETERS 478
TABLE 5-12: POWER MANAGEMENT SYSTEM MESSAGES AND DLLPS 479
TABLE 5-13: D0 POWER MANAGEMENT POLICIES 482
TABLE 5-14: D1 POWER MANAGEMENT POLICIES 482
TABLE 5-15: D2 POWER MANAGEMENT POLICIES 483
TABLE 5-16: $D3_{HOT}$ POWER MANAGEMENT POLICIES 484
TABLE 5-17: $D3_{COLD}$ POWER MANAGEMENT POLICIES 485
TABLE 5-18: PCI FUNCTION STATE TRANSITION DELAYS 485
TABLE 6-1: ERROR MESSAGES 504
TABLE 6-2: GENERAL PCI EXPRESS ERROR LIST 520
TABLE 6-3: PHYSICAL LAYER ERROR LIST 520
TABLE 6-4: DATA LINK LAYER ERROR LIST 521
TABLE 6-5: TRANSACTION LAYER ERROR LIST 521
TABLE 6-6: ELEMENTS OF HOT-PLUG 562
TABLE 6-7: ATTENTION INDICATOR STATES 563
TABLE 6-8: POWER INDICATOR STATES 564
TABLE 6-9: ACS P2P REQUEST REDIRECT AND ACS P2P EGRESS CONTROL INTERACTIONS 588
TABLE 6-10: ECRC RULES FOR MC_OVERLAY 601
TABLE 6-11: PROCESSING HINT MAPPING 610
TABLE 6-12: ST MODES OF OPERATION 612
TABLE 6-13: PASID TLP PREFIX 625
TABLE 6-14: EMERGENCY POWER REDUCTION SUPPORTED VALUES 649
TABLE 6-15: SYSTEM GUID AUTHORITY ID ENCODING 653
TABLE 6-16: SMALL RESOURCE DATA TYPE TAG BIT DEFINITIONS 666
TABLE 6-17: LARGE RESOURCE DATA TYPE TAG BIT DEFINITIONS 666
TABLE 6-18: RESOURCE DATA TYPE FLAGS FOR A TYPICAL VPD 667
TABLE 6-19: EXAMPLE OF ADD-IN SERIAL CARD NUMBER 668
TABLE 6-20: VPD LARGE AND SMALL RESOURCE DATA TAGS 668
TABLE 6-21: VPD READ-ONLY FIELDS 669
TABLE 6-22: VPD READ/WRITE FIELDS 670
TABLE 6-23: VPD EXAMPLE 670
TABLE 6-24: NPEM STATES 674
TABLE 7-1: ENHANCED CONFIGURATION ADDRESS MAPPING 679
TABLE 7-2: REGISTER AND REGISTER BIT-FIELD TYPES 685
TABLE 7-3: COMMAND REGISTER 689
TABLE 7-4: STATUS REGISTER 693
TABLE 7-5: HEADER TYPE REGISTER 696
TABLE 7-6: HEADER TYPE REGISTER 697
TABLE 7-7: BIST REGISTER 698

TABLE 7-8: MEMORY BASE ADDRESS REGISTER BITS 2:1 ENCODING 702
TABLE 7-9: I/O ADDRESSING CAPABILITY 710
TABLE 7-10: SECONDARY STATUS REGISTER 711
TABLE 7-11: BRIDGE CONTROL REGISTER 715
TABLE 7-12: POWER MANAGEMENT CAPABILITIES REGISTER 718
TABLE 7-13: POWER MANAGEMENT CONTROL/STATUS REGISTER REQUIREMENTS 721
TABLE 7-14: DATA REGISTER 723
TABLE 7-15: POWER CONSUMPTION/DISSIPATION REPORTING 723
TABLE 7-16: PCI EXPRESS CAPABILITY LIST REGISTER 726
TABLE 7-17: PCI EXPRESS CAPABILITIES REGISTER 727
TABLE 7-18: DEVICE CAPABILITIES REGISTER 729
TABLE 7-19: DEVICE CONTROL REGISTER 733
TABLE 7-20: DEVICE STATUS REGISTER 740
TABLE 7-21: LINK CAPABILITIES REGISTER 743
TABLE 7-22: LINK CONTROL REGISTER 749
TABLE 7-23: LINK STATUS REGISTER 757
TABLE 7-24: SLOT CAPABILITIES REGISTER 760
TABLE 7-25: SLOT CONTROL REGISTER 762
TABLE 7-26: SLOT STATUS REGISTER 765
TABLE 7-27: ROOT CONTROL REGISTER 767
TABLE 7-28: ROOT CAPABILITIES REGISTER 769
TABLE 7-29: ROOT STATUS REGISTER 769
TABLE 7-30: DEVICE CAPABILITIES 2 REGISTER 771
TABLE 7-31: DEVICE CONTROL 2 REGISTER 778
TABLE 7-32: LINK CAPABILITIES 2 REGISTER 783
TABLE 7-33: LINK CONTROL 2 REGISTER 787
TABLE 7-34: LINK STATUS 2 REGISTER 792
TABLE 7-35: PCI EXPRESS EXTENDED CAPABILITY HEADER 797
TABLE 7-36: MSI CAPABILITY HEADER 800
TABLE 7-37: MESSAGE CONTROL REGISTER FOR MSI 801
TABLE 7-38: MESSAGE ADDRESS REGISTER FOR MSI 803
TABLE 7-39: MESSAGE UPPER ADDRESS REGISTER FOR MSI 803
TABLE 7-40: MESSAGE DATA REGISTER FOR MSI 804
TABLE 7-41: EXTENDED MESSAGE DATA REGISTER FOR MSI 805
TABLE 7-42: MASK BITS REGISTER FOR MSI 805
TABLE 7-43: PENDING BITS REGISTER FOR MSI 806
TABLE 7-44: MSI-X CAPABILITY HEADER 809
TABLE 7-45: MESSAGE CONTROL REGISTER FOR MSI-X 810
TABLE 7-46: TABLE OFFSET/TABLE BIR REGISTER FOR MSI-X 811
TABLE 7-47: PBA OFFSET/PBA BIR REGISTER FOR MSI-X 812
TABLE 7-48: MESSAGE ADDRESS REGISTER FOR MSI-X TABLE ENTRIES 812
TABLE 7-49: MESSAGE UPPER ADDRESS REGISTER FOR MSI-X TABLE ENTRIES 813
TABLE 7-50: MESSAGE DATA REGISTER FOR MSI-X TABLE ENTRIES 813
TABLE 7-51: VECTOR CONTROL REGISTER FOR MSI-X TABLE ENTRIES 814
TABLE 7-52: PENDING BITS REGISTER FOR MSI-X PBA ENTRIES 815
TABLE 7-53: SECONDARY PCI EXPRESS EXTENDED CAPABILITY HEADER 816

TABLE 7-54: LINK CONTROL 3 REGISTER 817
TABLE 7-55: LANE ERROR STATUS REGISTER 818
TABLE 7-56: LANE EQUALIZATION CONTROL REGISTER ENTRY 819
TABLE 7-57: DATA LINK FEATURE EXTENDED CAPABILITY HEADER 822
TABLE 7-58: DATA LINK FEATURE CAPABILITIES REGISTER 823
TABLE 7-59: DATA LINK FEATURE STATUS REGISTER 824
TABLE 7-60: PHYSICAL LAYER 16.0 GT/S EXTENDED CAPABILITY HEADER 825
TABLE 7-61: 16.0 GT/S CAPABILITIES REGISTER 826
TABLE 7-62: 16.0 GT/S CONTROL REGISTER 826
TABLE 7-63: 16.0 GT/S STATUS REGISTER 827
TABLE 7-64: 16.0 GT/S LOCAL DATA PARITY MISMATCH STATUS REGISTER 828
TABLE 7-65: 16.0 GT/S FIRST RETIMER DATA PARITY MISMATCH STATUS REGISTER 829
TABLE 7-66: 16.0 GT/S SECOND RETIMER DATA PARITY MISMATCH STATUS REGISTER 829
TABLE 7-67: 16.0 GT/S LANE ((MAXIMUM LINK WIDTH – 1):0) EQUALIZATION CONTROL REGISTER ENTRY 830
TABLE 7-68: PHYSICAL LAYER 16.0 GT/S MARGINING EXTENDED CAPABILITY HEADER 833
TABLE 7-69: MARGINING PORT CAPABILITIES REGISTER 834
TABLE 7-70: MARGINING PORT STATUS REGISTER 834
TABLE 7-71: LANE N: MARGINING LANE STATUS REGISTER ENTRY 836
TABLE 7-72: ACS EXTENDED CAPABILITY HEADER 837
TABLE 7-73: ACS CAPABILITY REGISTER 838
TABLE 7-74: ACS CONTROL REGISTER 839
TABLE 7-75: EGRESS CONTROL VECTOR 841
TABLE 7-76: POWER BUDGETING EXTENDED CAPABILITY HEADER 843
TABLE 7-77: POWER BUDGETING DATA REGISTER 844
TABLE 7-78: POWER BUDGET CAPABILITY REGISTER 847
TABLE 7-79: LTR EXTENDED CAPABILITY HEADER 848
TABLE 7-80: MAX SNOOP LATENCY REGISTER 849
TABLE 7-81: MAX NO-SNOOP LATENCY REGISTER 849
TABLE 7-82: L1 PM SUBSTATES EXTENDED CAPABILITY HEADER 851
TABLE 7-83: L1 PM SUBSTATES CAPABILITIES REGISTER 851
TABLE 7-84: L1 PM SUBSTATES CONTROL 1 REGISTER 853
TABLE 7-85: L1 PM SUBSTATES CONTROL 2 REGISTER 855
TABLE 7-86: ADVANCED ERROR REPORTING EXTENDED CAPABILITY HEADER 858
TABLE 7-87: UNCORRECTABLE ERROR STATUS REGISTER 859
TABLE 7-88: UNCORRECTABLE ERROR MASK REGISTER 861
TABLE 7-89: UNCORRECTABLE ERROR SEVERITY REGISTER 863
TABLE 7-90: CORRECTABLE ERROR STATUS REGISTER 865
TABLE 7-91: CORRECTABLE ERROR MASK REGISTER 866
TABLE 7-92: ADVANCED ERROR CAPABILITIES AND CONTROL REGISTER 867
TABLE 7-93: HEADER LOG REGISTER 869
TABLE 7-94: ROOT ERROR COMMAND REGISTER 869
TABLE 7-95: ROOT ERROR STATUS REGISTER 871
TABLE 7-96: ERROR SOURCE IDENTIFICATION REGISTER 873
TABLE 7-97: TLP PREFIX LOG REGISTER 874
TABLE 7-98: FIRST DW OF ENHANCED ALLOCATION CAPABILITY 874

TABLE 7-99: FIRST DW OF EACH ENTRY FOR ENHANCED ALLOCATION CAPABILITY 875
TABLE 7-100: ENHANCED ALLOCATION ENTRY FIELD VALUE DEFINITIONS FOR BOTH THE PRIMARY PROPERTIES AND SECONDARY PROPERTIES FIELDS..... 879
TABLE 7-101: RESIZABLE BAR EXTENDED CAPABILITY HEADER..... 883
TABLE 7-102: RESIZABLE BAR CAPABILITY REGISTER..... 884
TABLE 7-103: RESIZABLE BAR CONTROL REGISTER..... 885
TABLE 7-104: ARI CAPABILITY HEADER..... 888
TABLE 7-105: ARI CAPABILITY REGISTER 888
TABLE 7-106: ARI CONTROL REGISTER 889
TABLE 7-107: PASID EXTENDED CAPABILITY HEADER..... 890
TABLE 7-108: PASID CAPABILITY REGISTER..... 891
TABLE 7-109: PASID CONTROL REGISTER..... 892
TABLE 7-110: FRS QUEUING EXTENDED CAPABILITY HEADER 893
TABLE 7-111: FRS QUEUING CAPABILITY REGISTER 894
TABLE 7-112: FRS QUEUING STATUS REGISTER 895
TABLE 7-113: FRS QUEUING CONTROL REGISTER 896
TABLE 7-114: FRS MESSAGE QUEUE REGISTER..... 896
TABLE 7-115: FPB CAPABILITY HEADER 897
TABLE 7-116: FPB CAPABILITIES REGISTER..... 898
TABLE 7-117: FPB RID VECTOR CONTROL 1 REGISTER..... 900
TABLE 7-118: FPB RID VECTOR CONTROL 2 REGISTER..... 902
TABLE 7-119: FPB MEM LOW VECTOR CONTROL REGISTER 902
TABLE 7-120: FPB MEM HIGH VECTOR CONTROL 1 REGISTER..... 904
TABLE 7-121: FPB MEM HIGH VECTOR CONTROL 2 REGISTER..... 906
TABLE 7-122: FPB VECTOR ACCESS CONTROL REGISTER..... 907
TABLE 7-123: FPB VECTOR ACCESS DATA REGISTER..... 908
TABLE 7-124: VIRTUAL CHANNEL EXTENDED CAPABILITY HEADER 910
TABLE 7-125: PORT VC CAPABILITY REGISTER 1..... 911
TABLE 7-126: PORT VC CAPABILITY REGISTER 2..... 913
TABLE 7-127: PORT VC CONTROL REGISTER 914
TABLE 7-128: PORT VC STATUS REGISTER 915
TABLE 7-129: VC RESOURCE CAPABILITY REGISTER..... 916
TABLE 7-130: VC RESOURCE CONTROL REGISTER..... 917
TABLE 7-131: VC RESOURCE STATUS REGISTER..... 920
TABLE 7-132: DEFINITION OF THE 4-BIT ENTRIES IN THE VC ARBITRATION TABLE 921
TABLE 7-133: LENGTH OF THE VC ARBITRATION TABLE 921
TABLE 7-134: LENGTH OF PORT ARBITRATION TABLE 923
TABLE 7-135: MFVC EXTENDED CAPABILITY HEADER 925
TABLE 7-136: PORT VC CAPABILITY REGISTER 1..... 926
TABLE 7-137: PORT VC CAPABILITY REGISTER 2..... 927
TABLE 7-138: PORT VC CONTROL REGISTER 928
TABLE 7-139: PORT VC STATUS REGISTER 929
TABLE 7-140: VC RESOURCE CAPABILITY REGISTER..... 930
TABLE 7-141: VC RESOURCE CONTROL REGISTER..... 931
TABLE 7-142: VC RESOURCE STATUS REGISTER..... 933
TABLE 7-143: LENGTH OF FUNCTION ARBITRATION TABLE 935

TABLE 7-144: DEVICE SERIAL NUMBER EXTENDED CAPABILITY HEADER 936
TABLE 7-145: SERIAL NUMBER REGISTER 937
TABLE 7-146: VENDOR-SPECIFIC CAPABILITY 938
TABLE 7-147: VENDOR-SPECIFIC EXTENDED CAPABILITY HEADER 939
TABLE 7-148: VENDOR-SPECIFIC HEADER 940
TABLE 7-149: DESIGNATED VENDOR-SPECIFIC EXTENDED CAPABILITY HEADER 942
TABLE 7-150: DESIGNATED VENDOR-SPECIFIC HEADER 1 942
TABLE 7-151: DESIGNATED VENDOR-SPECIFIC HEADER 2 943
TABLE 7-152: RCRB HEADER EXTENDED CAPABILITY HEADER 944
TABLE 7-153: VENDOR ID AND DEVICE ID 944
TABLE 7-154: RCRB CAPABILITIES 945
TABLE 7-155: RCRB CONTROL 945
TABLE 7-156: ROOT COMPLEX LINK DECLARATION EXTENDED CAPABILITY HEADER 947
TABLE 7-157: ELEMENT SELF DESCRIPTION REGISTER 948
TABLE 7-158: LINK DESCRIPTION REGISTER 950
TABLE 7-159: LINK ADDRESS FOR LINK TYPE 1 952
TABLE 7-160: ROOT COMPLEX INTERNAL LINK CONTROL EXTENDED CAPABILITY HEADER 953
TABLE 7-161: ROOT COMPLEX LINK CAPABILITIES REGISTER 954
TABLE 7-162: ROOT COMPLEX LINK CONTROL REGISTER 958
TABLE 7-163: ROOT COMPLEX LINK STATUS REGISTER 959
TABLE 7-164: ROOT COMPLEX EVENT COLLECTOR ENDPOINT ASSOCIATION EXTENDED CAPABILITY HEADER 961
TABLE 7-165: MULTICAST EXTENDED CAPABILITY HEADER 963
TABLE 7-166: MULTICAST CAPABILITY REGISTER 963
TABLE 7-167: MULTICAST CONTROL REGISTER 964
TABLE 7-168: MC_BASE ADDRESS REGISTER 965
TABLE 7-169: MC_RECEIVE REGISTER 965
TABLE 7-170: MC_BLOCK_ALL REGISTER 966
TABLE 7-171: MC_BLOCK_UNTRANSLATED REGISTER 967
TABLE 7-172: MC_OVERLAY BAR 967
TABLE 7-173: DPA EXTENDED CAPABILITY HEADER 968
TABLE 7-174: DPA CAPABILITY REGISTER 969
TABLE 7-175: DPA LATENCY INDICATOR REGISTER 970
TABLE 7-176: DPA STATUS REGISTER 970
TABLE 7-177: DPA CONTROL REGISTER 971
TABLE 7-178: SUBSTATE POWER ALLOCATION REGISTER (0 TO SUBSTATE_MAX) 971
TABLE 7-179: TPH REQUESTER EXTENDED CAPABILITY HEADER 972
TABLE 7-180: TPH REQUESTER CAPABILITY REGISTER 973
TABLE 7-181: TPH REQUESTER CONTROL REGISTER 974
TABLE 7-182: TPH ST TABLE 975
TABLE 7-183: LNR EXTENDED CAPABILITY HEADER 976
TABLE 7-184: LNR CAPABILITY REGISTER 977
TABLE 7-185: LNR CONTROL REGISTER 978
TABLE 7-186: DPC EXTENDED CAPABILITY HEADER 979
TABLE 7-187: DPC CAPABILITY REGISTER 980
TABLE 7-188: DPC CONTROL REGISTER 982

TABLE 7-189: DPC STATUS REGISTER 983
TABLE 7-190: DPC ERROR SOURCE ID REGISTER 985
TABLE 7-191: RP PIO STATUS REGISTER 986
TABLE 7-192: RP PIO MASK REGISTER 987
TABLE 7-193: RP PIO SEVERITY REGISTER 988
TABLE 7-194: RP PIO SYSERROR REGISTER 988
TABLE 7-195: RP PIO EXCEPTION REGISTER 990
TABLE 7-196: RP PIO HEADER LOG REGISTER 990
TABLE 7-197: RP PIO IMPSPEC LOG REGISTER 991
TABLE 7-198: RP PIO TLP PREFIX LOG REGISTER 992
TABLE 7-199: PTM EXTENDED CAPABILITY HEADER 993
TABLE 7-200: PTM CAPABILITY REGISTER 993
TABLE 7-201: PTM CONTROL REGISTER 994
TABLE 7-202: READINESS TIME REPORTING EXTENDED CAPABILITY HEADER 997
TABLE 7-203: READINESS TIME REPORTING 1 REGISTER 998
TABLE 7-204: READINESS TIME REPORTING 2 REGISTER 999
TABLE 7-205: HIERARCHY ID EXTENDED CAPABILITY HEADER 1001
TABLE 7-206: HIERARCHY ID STATUS REGISTER 1002
TABLE 7-207: HIERARCHY ID DATA REGISTER 1004
TABLE 7-208: HIERARCHY ID GUID 1 REGISTER 1005
TABLE 7-209: HIERARCHY ID GUID 2 REGISTER 1006
TABLE 7-210: HIERARCHY ID GUID 3 REGISTER 1006
TABLE 7-211: HIERARCHY ID GUID 4 REGISTER 1007
TABLE 7-212: HIERARCHY ID GUID 5 REGISTER 1008
TABLE 7-213: VPD REGISTER FIELDS 1009
TABLE 7-214: NPEM EXTENDED CAPABILITY HEADER 1010
TABLE 7-215: NPEM CONTROL REGISTER 1013
TABLE 7-216: NPEM STATUS REGISTER 1015
TABLE 8-1. TX PRESET RATIOS AND CORRESPONDING COEFFICIENT VALUES 1022
TABLE 8-2. PRESET MEASUREMENT CROSS REFERENCE TABLE 1024
TABLE 8-3: TRANSMIT EQUALIZATION COEFFICIENT SPACE TRIANGULAR MATRIX EXAMPLE. 1026
TABLE 8-4. CASES THAT THE REFERENCE PACKAGES AND PS21TX PARAMETER ARE NORMATIVE 1028
TABLE 8-5: RECOMMENDED DE-EMBEDDING CUTOFF FREQUENCY 1030
TABLE 8-6: TX MEASUREMENT AND POST PROCESSING FOR DIFFERENT REFCLKS 1031
TABLE 8-7: DATA RATE DEPENDENT TRANSMITTER PARAMETERS 1038
TABLE 8-8: DATA RATE INDEPENDENT TX PARAMETERS 1042
TABLE 8-9: CALIBRATION CHANNEL IL LIMITS 1045
TABLE 8-10: STRESSED JITTER EYE PARAMETERS 1057
TABLE 8-11: COMMON RECEIVER PARAMETERS 1062
TABLE 8-12: LANE MARGINING TIMING 1066
TABLE 8-13: PACKAGE MODEL CAPACITANCE VALUES 1072
TABLE 8-14: JITTER/VOLTAGE PARAMETERS FOR CHANNEL TOLERANCING 1076
TABLE 8-15: CHANNEL TOLERANCING EYE MASK VALUES 1078
TABLE 8-16: EIEOS SIGNALING PARAMETERS 1079
TABLE 8-17: REFCLK DC SPECIFICATIONS AND AC TIMING REQUIREMENTS 1080

TABLE 8-18: DATA RATE INDEPENDENT REFCLK PARAMETERS 1083
TABLE 8-19: COMMON REFCLK PLL AND CDR CHARACTERISTICS FOR 2.5 GT/S..... 1087
TABLE 8-20: COMMON REFCLK PLL AND CDR CHARACTERISTICS FOR 5.0 GT/S..... 1087
TABLE 8-21: COMMON REFCLK PLL AND CDR CHARACTERISTICS FOR 8.0 AND 16.0 GT/S 1088
TABLE 8-22: JITTER LIMITS FOR CC ARCHITECTURE..... 1088
TABLE 8-23: FORM FACTOR CLOCKING ARCHITECTURE REQUIREMENTS..... 1089
TABLE 8-24: COMMON CLOCK ARCHITECTURE DETAILS..... 1089
TABLE 8-25: FORM FACTOR CLOCKING ARCHITECTURE REQUIREMENTS EXAMPLE..... 1089
TABLE 8-26: COMMON CLOCK ARCHITECTURE DETAILS EXAMPLE 1089
TABLE 9-1: VF ROUTING ID ALGORITHM..... 1105
TABLE 9-2: SR-IOV VF MIGRATION STATE TABLE 1111
TABLE 9-3: SR-IOV EXTENDED CAPABILITY HEADER 1113
TABLE 9-4: SR-IOV CAPABILITIES 1114
TABLE 9-5: SR-IOV CONTROL 1117
TABLE 9-6: SR-IOV STATUS 1121
TABLE 9-7: BAR OFFSETS 1127
TABLE 9-8: VF MIGRATION STATE ARRAY OFFSET..... 1128
TABLE 9-9: VF MIGRATION STATE ENTRY 1129
TABLE 9-10: VF MIGRATION STATE DESCRIPTIONS 1129
TABLE 9-11: SR-PCIM INITIATED VF MIGRATION STATE TRANSITIONS 1129
TABLE 9-12: MR-PCIM INITIATED VF MIGRATION STATE TRANSITIONS 1130
TABLE 9-13: COMMAND REGISTER..... 1132
TABLE 9-14: STATUS REGISTER..... 1133
TABLE 9-15: DEVICE CAPABILITIES REGISTER 1135
TABLE 9-16: DEVICE CONTROL REGISTER..... 1136
TABLE 9-17: DEVICE STATUS REGISTER..... 1136
TABLE 9-18: LINK CONTROL REGISTER 1137
TABLE 9-19: DEVICE CAPABILITIES 2 REGISTER..... 1138
TABLE 9-20: DEVICE CONTROL 2 REGISTER 1139
TABLE 9-21. LINK STATUS 2 REGISTER..... 1140
TABLE 9-22: SR-IOV USAGE OF PCI STANDARD CAPABILITIES..... 1140
TABLE 9-23: SR-IOV USAGE OF PCI EXPRESS EXTENDED CAPABILITIES 1141
TABLE 9-24: VF RESIZABLE BAR EXTENDED CAPABILITY HEADER..... 1146
TABLE 9-25: VF RESIZABLE BAR CONTROL REGISTER..... 1147
TABLE 9-26: ACS CAPABILITY REGISTER 1148
TABLE 9-27: ARI CAPABILITY REGISTER 1149
TABLE 9-28: ATS CAPABILITY REGISTER..... 1150
TABLE 9-29: ATS CONTROL REGISTER 1150
TABLE 9-30: MULTICAST CAPABILITY REGISTER 1150
TABLE 9-31: MULTICAST CONTROL REGISTER 1151
TABLE 9-32: MULTICAST BASE ADDRESS REGISTER 1151
TABLE 9-33: UNCORRECTABLE ERROR STATUS REGISTER 1154
TABLE 9-34: UNCORRECTABLE ERROR MASK REGISTER..... 1154
TABLE 9-35: UNCORRECTABLE ERROR SEVERITY REGISTER..... 1155
TABLE 9-36: CORRECTABLE ERROR STATUS REGISTER..... 1156
TABLE 9-37: CORRECTABLE ERROR MASK REGISTER 1156

TABLE 9-38: ADVANCED ERROR CAPABILITIES AND CONTROL REGISTER 1156
TABLE 9-39: HEADER LOG REGISTER 1157
TABLE 9-40: MSI CAPABILITY: MESSAGE CONTROL 1159
TABLE 9-41: POWER MANAGEMENT CONTROL/STATUS (PMCSR) 1161
TABLE 9-42: POWER MANAGEMENT DATA REGISTER 1162
TABLE 10-1: ADDRESS TYPE (AT) FIELD ENCODINGS 1174
TABLE 10-2: TRANSLATION COMPLETION WITH NO DATA STATUS CODES 1177
TABLE 10-3: TRANSLATION COMPLETION DATA FIELDS 1179
TABLE 10-4: EXAMPLES OF TRANSLATION SIZE USING S FIELD 1181
TABLE 10-5: PAGE REQUEST MESSAGE DATA FIELDS 1196
TABLE 10-6: PRG RESPONSE MESSAGE DATA FIELDS 1201
TABLE 10-7: RESPONSE CODES 1202
TABLE 10-8: ATS EXTENDED CAPABILITY HEADER 1203
TABLE 10-9: ATS CAPABILITY REGISTER 1204
TABLE 10-10: ATS CONTROL REGISTER 1205
TABLE 10-11: PAGE REQUEST EXTENDED CAPABILITY HEADER 1206
TABLE 10-12: PAGE REQUEST CONTROL REGISTER 1207
TABLE 10-13: PAGE REQUEST STATUS REGISTER 1208
TABLE A-1: ISOCHRONOUS BANDWIDTH RANGES AND GRANULARITIES 1216
TABLE B-1: 8B/10B DATA SYMBOL CODES 1224
TABLE B-2: 8B/10B SPECIAL CHARACTER SYMBOL CODES 1232
TABLE F-1: MESSAGE CODE USAGE 1251
TABLE F-2: PCI-SIG-DEFINED VDM SUBTYPE USAGE 1252
TABLE G-1: PCI EXPRESS ATTRIBUTE IMPACT ON PROTOCOL MULTIPLEXING 1256
TABLE G-2: PMUX ATTRIBUTE IMPACT ON PCI EXPRESS 1259
TABLE G-3: PMUX PACKET LAYOUT (8B/10B ENCODING) 1263
TABLE G-4: PMUX PACKET LAYOUT (128B/130B ENCODING) 1265
TABLE G-5: SYMBOL 1 BITS [6:3] 1266
TABLE G-6: PMUX EXTENDED CAPABILITY HEADER 1268
TABLE G-7: PMUX CAPABILITY REGISTER 1269
TABLE G-8: PMUX CONTROL REGISTER 1271
TABLE G-9: PMUX STATUS REGISTER 1273
TABLE G-10: PMUX PROTOCOL ARRAY ENTRY 1275
TABLE H-1: MAXIMUM UPDATEFC TRANSMISSION LATENCY GUIDELINES FOR 2.5 GT/S MODE OPERATION BY LINK WIDTH AND MAX PAYLOAD (SYMBOL TIMES) 1278
TABLE H-2: MAXIMUM UPDATEFC TRANSMISSION LATENCY GUIDELINES FOR 5.0 GT/S MODE OPERATION BY LINK WIDTH AND MAX PAYLOAD (SYMBOL TIMES) 1278
TABLE H-3: MAXIMUM UPDATEFC TRANSMISSION LATENCY GUIDELINES FOR 8.0 GT/S OPERATION BY LINK WIDTH AND MAX PAYLOAD (SYMBOL TIMES) 1279
TABLE H-4: MAXIMUM ACK LATENCY LIMIT AND ACKFACTOR FOR 2.5 GT/S (SYMBOL TIMES) 1280
TABLE H-5: MAXIMUM ACK TRANSMISSION LATENCY LIMIT AND ACKFACTOR FOR 5.0 GT/S (SYMBOL TIMES) 1281
TABLE H-6: MAXIMUM ACK TRANSMISSION LATENCY LIMIT AND ACKFACTOR FOR 8.0 GT/S (SYMBOL TIMES) 1281

This page left intentionally blank

Objective of the Specification

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

The goal is to enable such devices from different vendors to inter-operate in an open architecture. The specification is intended as an enhancement to the PCI™ architecture spanning multiple market segments; clients (desktops and mobile), servers (standard and enterprise), and embedded and communication devices. The specification allows system OEMs and peripheral developers adequate room for product versatility and market differentiation without the burden of carrying obsolete interfaces or losing compatibility.

Document Organization

The PCI Express Specification is organized as a base specification and a set of companion documents. The *PCI Express Base Specification* and the *PCI Express Card Electromechanical Specification* are among those that have been published. As the PCI Express definition evolves, other companion documents will be published.

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

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

Documentation Conventions

Capitalization

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

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

Numbers and Number Bases

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

Binary numbers larger than four digits are written with a space dividing each group of four digits, as in 1000 0101 0010b.

All other numbers are decimal.

Implementation Notes

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

Terms and Acronyms

8b/10b	The data encoding scheme[1] used in the PCI Express Physical Layer for 5.0 GT/s and below.
10-Bit Tags	A Tag's capability that provides a total of 10 bits for the Tag field. See Tag.
Access Control Services, ACS	A set of capabilities and control registers used to implement access control over routing within a PCI Express component.
ACS Violation	An error that applies to a Posted or Non-Posted Request when the Completer detects an access control violation.
Adapter	Used generically to refer to an add-in card or module.
Advanced Error Reporting, AER	Advanced Error Reporting (see Section 7.8.4).
Advertise (Credits)	Used in the context of Flow Control, the act of a Receiver sending information regarding its Flow Control Credit availability.
Alternative Routing-ID, ARI	Alternative Routing-ID Interpretation. Applicable to Requester IDs and Completer IDs as well as Routing IDs.
ARI Device	A Device associated with an Upstream Port, whose Functions each contain an ARI Capability structure.
ARI Downstream Port	A Switch Downstream Port or Root Port that supports ARI Forwarding.
ARI Forwarding	Functionality that enables the Downstream Port immediately above an ARI Device to access the Devices extended Functions. Enabling ARI Forwarding ensures the logic that determines when to turn a Type 1 Configuration Request into a Type 0 Configuration Request no longer enforces a restriction on the traditional Device Number field being 0.
Asserted	The active logical state of a conceptual or actual signal.
Async Removal	Removal of an adapter or cable from a slot without lock-step synchronization with the operating system (i.e., in an asynchronous manner without button presses, etc.).

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

Atomic Operation, AtomicOp	One of three architected Atomic Operations where a single PCI Express transaction targeting a location in Memory Space reads the location's value, potentially writes a new value to the location, and returns the original value. This read-modify-write sequence to the location is performed atomically. AtomicOps include FetchAdd, Swap, and CAS.
Attribute	Transaction handling preferences indicated by specified Packet header bits and fields (e.g., non-snoop).
Base Address Register, BAR	Base Address Registers exist within Configuration Space and are used to determine the amount of system memory space needed by a Function and to provide the base address for a mapping to Function memory space. A Base Address Register may map to memory space or I/O space.
Beacon	An optional 30 kHz to 500 MHz in-band signal used to exit the L2 Link Power Management state. One of two defined mechanisms for waking up a Link in L2 (see Wakeup).
Bridge	One of several defined System Elements. A Function that virtually or actually connects a PCI/PCI-X segment or PCI Express Port with an internal component interconnect or with another PCI/PCI-X bus segment or PCI Express Port. A virtual *Bridge* in a Root Complex or Switch must use the software configuration interface described in this specification.
by-1, x1	A Link or Port with one Physical Lane.
by-8, x8	A Link or Port with eight Physical Lanes.
by-N, xN	A Link or Port with "N" Physical Lanes.
Compare and Swap, CAS	An AtomicOp where the value of a target location is compared to a specified value and, if they match, another specified value is written back to the location. Regardless, the original value of the location is returned.
Character	An 8-bit quantity treated as an atomic entity; a byte.
Clear	A bit is Clear when its value is 0b.
cold reset	A Fundamental Reset following the application of main power.
Completer	The Function that terminates or "completes" a given Request, and generates a Completion if appropriate. Generally the Function targeted by the Request serves as the Completer. For cases when an uncorrectable error prevents the Request from reaching its targeted Function, the Function that detects and handles the error serves as the Completer.
Completer Abort, CA	1. A status that applies to a posted or non-posted Request that the Completer is permanently unable to complete successfully, due to a violation of the Completer's programming model or to an unrecoverable error associated with the Completer. 2. A status indication returned with a Completion for a non-posted Request that suffered a Completer Abort at the Completer.
Completer ID	The combination of a Completer's Bus Number, Device Number, and Function Number that uniquely identifies the Completer of the Request within a Hierarchy. With an ARI Completer ID, bits traditionally used for the Device Number field are used instead to expand the Function Number field, and the Device Number is implied to be 0.

Completion	A Packet used to terminate, or to partially terminate, a transaction sequence. A *Completion* always corresponds to a preceding Request, and, in some cases, includes data.
component	A physical device (a single package).
Configuration Software	The component of system software responsible for accessing Configuration Space and configuring the PCI/PCIe bus.
Configuration Space	One of the four address spaces within the PCI Express architecture. Packets with a *Configuration Space* address are used to configure Functions.
Configuration-Ready	A Function is "Configuration-Ready" when it is guaranteed that the Function will respond to a valid Configuration Request targeting the Function with a Completion indicating Successful Completion status.
Conventional PCI	Behaviors or features originally defined in the *PCI Local Bus Specification.* The PCI Express Base 4.0 and subsequent specifications incorporate the relevant requirements from the *PCI Local Bus Specification*.
Conventional Reset	A Hot, Warm, or Cold reset. Distinct from Function Level Reset (FLR).
Data Link Layer	The intermediate Layer that is between the Transaction Layer and the Physical Layer.
Data Link Layer Packet, DLLP	A Packet generated in the Data Link Layer to support Link management functions.
data payload	Information following the header in some packets that is destined for consumption by the targeted Function receiving the Packet (for example, Write Requests or Read Completions).
deasserted	The inactive logical state of a conceptual or actual signal.
Design for Testability DFT	Design for Testability.
Device	A collection of one or more Functions within a single Hierarchy identified by common Bus Number and Device Number. An SR-IOV Device may have additional Functions accessed via additional Bus Numbers and/or Device Numbers configured through one or more SR-IOV Capability structures.
device	1. A physical or logical entity that performs a specific type of I/O. 2. A component on either end of a PCI Express Link. 3. A common imprecise synonym for Function, particularly when a device has a single Function.
Device Readiness Status, DRS	A mechanism for indicating that a Device is Configuration-Ready (see Section 6.23.1)
Downstream	1. The relative position of an interconnect/System Element (Port/component) that is farther from the Root Complex. The Ports on a Switch that are not the Upstream Port are *Downstream* Ports. All Ports on a Root Complex are *Downstream* Ports. The *Downstream* component on a Link is the component farther from the Root Complex. 2. A direction of information flow where the information is flowing away from the Root Complex.
Downstream Path	The flow of data through a Retimer from the Upstream Pseudo Port Receiver to the Downstream Pseudo Port Transmitter.

Downstream Port Containment, DPC	The automatic disabling of the Link below a Downstream Port following an uncorrectable error, which prevents TLPs subsequent to the error from propagating Upstream or Downstream.
DWORD, DW	Four bytes. Used in the context of a data payload, the 4 bytes of data must be on a naturally aligned 4-byte boundary (the least significant 2 bits of the byte address are 00b).
Egress Port	The transmitting Port; that is, the Port that sends outgoing traffic.
Electrical Idle	A Link state used in a variety of defined cases, with specific requirements defined for the Transmitter and Receiver.
End-End TLP Prefix	A TLP Prefix that is carried along with a TLP from source to destination. See Section 2.2.10.2.
Endpoint	One of several defined System Elements. A Function that has a Type 00h Configuration Space header.
error detection	Mechanisms that determine that an error exists, either by the first agent to discover the error (e.g., Malformed TLP) or by the recipient of a signaled error (e.g., receiver of a poisoned TLP).
error logging	A detector setting one or more bits in architected registers based on the detection of an error. The detector might be the original discoverer of an error or a recipient of a signaled error.
error reporting	In a broad context, the general notification of errors. In the context of the Device Control register, sending an error Message. In the context of the Root Error Command register, signaling an interrupt as a result of receiving an error Message.
error signaling	One agent notifying another agent of an error either by (1) sending an error Message, (2) sending a Completion with UR/CA Status, or (3) poisoning a TLP.
Extension Device	A component whose purpose is to extend the physical length of a Link.
Extended Function	Within an ARI Device, a Function whose Function Number is greater than 7. Extended Functions are accessible only after ARI-aware software has enabled ARI Forwarding in the Downstream Port immediately above the ARI Device.
FetchAdd	Fetch and Add. An AtomicOp where the value of a target location is incremented by a specified value using two's complement arithmetic ignoring any carry or overflow, and the result is written back to the location. The original value of the location is returned.
Flow Control	The method for communicating receive buffer status from a Receiver to a Transmitter to prevent receive buffer overflow and allow Transmitter compliance with ordering rules.
Flow Control Packet, FCP	A DLLP used to send Flow Control information from the Transaction Layer in one component to the Transaction Layer in another component.
Function	Within a Device, an addressable entity in Configuration Space associated with a single Function Number. Used to refer to one Function of a Multi-Function Device, or to the only Function in a single-Function Device. Specifically included are special types of Functions defined in the I/O Virtualization specifications, notably Physical Functions and Virtual Functions.

Function Group	Within an ARI Device, a configurable set of Functions that are associated with a single Function Group Number. Function Groups can optionally serve as the basis for VC arbitration or access control between multiple Functions within the ARI Device.
Function Level Reset, FLR	A mechanism for resetting a specific Endpoint Function (see Section 6.6.2).
Function Readiness Status, FRS	A mechanism for indicating that a Function is Configuration-Ready (see Section 6.23.2)
Fundamental Reset	A hardware mechanism for setting or returning all Port states to the initial conditions specified in this document (see Section 6.6).
header	A set of fields that appear at or near the front of a Packet that contain the information required to determine the characteristics and purpose of the Packet.
Hierarchy	A tree structured PCI Express I/O interconnect topology, wherein the Configuration Space addresses (IDs) used for routing and Requester/Completer identification are unique. A system may contain multiple Hierarchies.
hierarchy domain	The part of a Hierarchy originating from a single Root Port.
Host Bridge	Part of a Root Complex that connects a host CPU or CPUs to a Hierarchy.
Hot Reset	A reset propagated in-band across a Link using a Physical Layer mechanism.
in-band signaling	A method for signaling events and conditions using the Link between two components, as opposed to the use of separate physical (sideband) signals. All mechanisms defined in this document can be implemented using *in-band signaling*, although in some form factors sideband signaling may be used instead.
Ingress Port	Receiving Port; that is, the Port that accepts incoming traffic.
Internal Error	An error associated with a PCI Express interface that occurs within a component and which may not be attributable to a packet or event on the PCI Express interface itself or on behalf of transactions initiated on PCI Express.
I/O Space	One of the four address spaces of the PCI Express architecture.
isochronous	Data associated with time-sensitive applications, such as audio or video applications.
invariant	A field of a TLP header or TLP Prefix that contains a value that cannot legally be modified as the TLP flows through the PCI Express fabric.
Lane	A set of differential signal pairs, one pair for transmission and one pair for reception. A by-N Link is composed of N *Lanes*.
Layer	A unit of distinction applied to this specification to help clarify the behavior of key elements. The use of the term *Layer* does not imply a specific implementation.
Link	The collection of two Ports and their interconnecting Lanes. A *Link* is a dual-simplex communications path between two components.
Link Segment	The collection of a Port and a Pseudo Port or two Pseudo Ports and their interconnecting Lanes. A Link Segment is a dual simplex communications path between a Component and a Retimer or between two Retimers (two Pseudo Ports).
Lightweight Notification, LN	A lightweight protocol that supports notifications to Endpoints via a hardware mechanism when cachelines of interest are updated.

LN Completer, LNC	A service subsystem in the host that receives LN Read/Write Requests, and sends LN Messages when registered cachelines are updated.
LN Completion	A Completion whose TLP Header has the LN bit Set.
LN Message	An architected Message used for notifications with LN protocol.
LN Read	A Memory Read Request whose TLP Header has the LN bit Set.
LN Requester, LNR	A client subsystem in an Endpoint that sends LN Read/Write Requests and receives LN Messages.
LN Write	A Memory Write Request whose TLP Header has the LN bit Set.
Local TLP Prefix	A TLP Prefix that is carried along with a TLP on a single Link. See Section 2.2.10.1.
Logical Bus	The logical connection among a collection of Devices that have the same Bus Number in Configuration Space.
Logical Idle	A period of one or more Symbol Times when no information (TLPs, DLLPs, or any special Symbol) is being transmitted or received. Unlike Electrical Idle, during *Logical Idle* the Idle data Symbol is being transmitted and received.
LTR	Abbreviation for Latency Tolerance ReportingMalformed Packet A TLP that violates specific TLP formation rules as defined in this specification.
Memory Space	One of the four address spaces of the PCI Express architecture.
Message	A TLP used to communicate information outside of the Memory, I/O, and Configuration Spaces.
Message Signaled Interrupt (MSI/MSI-X)	Two similar but separate mechanisms that enable a Function to request service by writing a system-specified DWORD of data to a system-specified address using a Memory Write Request. Compared to MSI, MSI-X supports a larger maximum number of vectors and independent message address and data for each vector.
Message Space	One of the four address spaces of the PCI Express architecture.
Multicast, MC	A feature and associated mechanisms that enable a single Posted Request TLP sent by a source to be distributed to multiple targets.
Multicast Group, MCG	A set of Endpoints that are the target of Multicast TLPs in a particular address range.
Multicast Hit	The determination by a Receiver that a TLP will be handled as a Multicast TLP.
Multicast TLP	A TLP that is potentially distributed to multiple targets, as controlled by Multicast Capability structures in the components through which the TLP travels.
Multicast Window	A region of Memory Space where Posted Request TLPs that target it will be handled as Multicast TLPs.
Multi-Function Device, MFD	A Device that has multiple Functions.
Multi-Root I/O Virtualization (MR-IOV)	A Function that supports the MR-IOV capability. See the *Multi-Root I/O Virtualization and Sharing Specification* for additional information.

naturally aligned	A data payload with a starting address equal to an integer multiple of a power of two, usually a specific power of two. For example, 64-byte *naturally aligned* means the least significant 6 bits of the byte address are 00 0000b.
NPEM	Native PCIe Enclosure Management
OBFF	Optimized Buffer Flush/Fill
Operating System	Throughout this specification, the terms operating system and system software refer to the combination of power management services, device drivers, user-mode services, and/or kernel mode services.
P2P	Peer-to-peer.
Path	The flow of data through a Retimer, in either the Upstream Path or the Downstream Path.
Packet	A fundamental unit of information transfer consisting of an optional TLP Prefix, followed by a header and, in some cases, followed by a data payload.
Parts per Million, ppm	Applied to frequency, the difference, in millionths of a Hertz, between a stated ideal frequency, and the measured long-term average of a frequency.
PCIe®	PCI Express®
PCI Bridge	See Type 1 Function.
PCI Software Model	The software model necessary to initialize, discover, configure, and use a PCI-compatible device, as specified in the *PCI Local Bus Specification, Revision 3.0*, the *PCI-X Addendum to the PCI Local Bus Specification, Revision 2.0*, and the *PCI Firmware Specification*.
Phantom Function Number, PFN	An unclaimed Function Number that may be used to expand the number of outstanding transaction identifiers by logically combining the *PFN* with the Tag identifier to create a unique transaction identifier.
Physical Function (PF)	A PCI Function that supports the SR-IOV capabilities defined in this specification. A PF contains the SR-IOV capability structure.
Physical Lane	See Lane.
Physical Layer	The Layer that directly interacts with the communication medium between two components.
Port	1. Logically, an interface between a component and a PCI Express Link. 2. Physically, a group of Transmitters and Receivers located on the same chip that define a Link.
Power Management	Software or Hardware mechanisms used to minimize system power consumption, manage system thermal limits, and maximize system battery life. Power management involves tradeoffs among system speed, noise, battery life, and AC power consumption.
PMUX Channel	A multiplexed channel on a PMUX Link that is configured to transport a specific multiplexed protocol. See Appendix G.
PMUX Link	A Link where Protocol Multiplexing is supported and enabled. See Appendix G.
PMUX Packet	A non-PCI Express Packet transported over a PCI Express Link. See Appendix G.

Precision Time Measurement (PTM)	Optional capability for communicating precise timing information between components.
Process Address Space ID (PASID)	The Process Address Space ID, in conjunction with the Requester ID, uniquely identifies the address space associated with a transaction.
Pseudo Port	1. Logically, an interface between a Retimer and a PCI Express Link Segment. 2. Physically, a group of Transmitters and Receivers located on the same Retimer chip that define a Link Segment.
Quality of Service, QoS	Attributes affecting the bandwidth, latency, jitter, relative priority, etc., for differentiated classes of traffic.
QWORD, QW	Eight bytes. Used in the context of a data payload, the 8 bytes of data must be on a naturally aligned 8-byte boundary (the least significant 3 bits of the address are 000b).
RCiEP	Root Complex Integrated Endpoint.
Receiver, Rx	The component that receives Packet information across a Link.
Receiving Port	In the context of a specific TLP or DLLP, the Port that receives the Packet on a given Link.
Re-driver	A non-protocol aware, software transparent, Extension Device.
repeater	An imprecise term for Extension Device.
Reported Error	An error subject to the logging and signaling requirements architecturally defined in this document
Request	A Packet used to initiate a transaction sequence. A *Request* includes operation code and, in some cases, address and length, data, or other information.
Requester	The Function that first introduces a transaction sequence into the PCI Express domain.
Requester ID	The combination of a Requester's Bus Number, Device Number, and Function Number that uniquely identifies the Requester within a Hierarchy. With an ARI Requester ID, bits traditionally used for the Device Number field are used instead to expand the Function Number field, and the Device Number is implied to be 0.
Reserved	The contents, states, or information are not defined at this time. Using any *Reserved* area (for example, packet header bit-fields, configuration register bits) is not permitted. Reserved register fields must be read only and must return 0 (all 0's for multi-bit fields) when read. Reserved encodings for register and packet fields must not be used. Any implementation dependence on a *Reserved* field value or encoding will result in an implementation that is not PCI Express-compliant. The functionality of such an implementation cannot be guaranteed in this or any future revision of this specification.
Refclk	An abbreviation for Reference Clock.
Retimer	A Physical Layer protocol aware, software transparent, Extension Device that forms two separate electrical Link Segments.
Root Complex (RC)	A defined System Element that includes at least one Host Bridge, Root Port, or Root Complex Integrated Endpoint.
Root Complex Component	A logical aggregation of Root Ports, Root Complex Register Blocks, Root Complex Integrated Endpoints, and Root Complex Event Collectors.

Root Port (RP)	A PCI Express Port on a Root Complex that maps a portion of a Hierarchy through an associated virtual PCI-PCI Bridge.
Routing Element	A term referring to a Root Complex, Switch, or Bridge in regard to its ability to route, multicast, or block TLPs.
Routing ID	Either the Requester ID or Completer ID that identifies a PCI ExpressFunction.
RP PIO	Root Port Programmed I/O. See Section 6.2.10.3.
Set	A bit is Set when its value is 1b.
sideband signaling	A method for signaling events and conditions using physical signals separate from the signals forming the Link between two components. All mechanisms defined in this document can be implemented using in-band signaling, although in some form factors sideband signaling may be used instead.
single–Function Device (SFD)	A device that has a single Function
Single Root I/O Virtualization (SR-IOV)	A function that supports the SR-IOV capability defined in this specification.
Single Root PCI Manager (SR-PCIM)	Software responsible for configuration and management the SR-IOV capability and PF/VF as well as dealing with associated error handling. Multiple implementation options exist; therefore, SR-PCIM implementation is outside the scope of this specification.
SR-IOV Device	A Device containing one or more Functions that have an SR-IOV Capability structure.
SSD	Solid State Drive
Swap	Unconditional Swap. An AtomicOp where a specified value is written to a target location, and the original value of the location is returned.
Switch	A defined System Element that connects two or more Ports to allow Packets to be routed from one Port to another. To configuration software, a *Switch* appears as a collection of virtual PCI-to-PCI Bridges.
Symbol	A 10-bit quantity when using 8b/10b encoding. An 8-bit quantity when using 128b/130b encoding.
Symbol Time	The period of time required to place a Symbol on a Lane (10 times the Unit Interval when using 8b/10b encoding and 8 times the Unit Interval when using 128b/130b encoding).
System Element	A defined Device or collection of devices that operate according to distinct sets of rules. The following *System Elements* are defined: Root Complex, Endpoint, Switch, and Bridge.
System Image (SI)	A software component running on a virtual system to which specific Functions, PF, and VF can be assigned. Specification of the behavior and architecture of an SI is outside the scope of this specification. Examples of SIs include guest operating systems and shared/non-shared protected domain device drivers.
System Software	Includes System Firmware (BIOS, UEFI), Operating System, VMM, management software, platform vendor's add-on to the Operating System.
Tag	A number assigned to a given Non-Posted Request to distinguish Completions for that Request from other Requests.

TLP Prefix	Additional information that may be optionally prepended to a TLP. TLP Prefixes are either Local or End-End. A TLP can have multiple TLP Prefixes. See Section 2.2.10.
TPH	Abbreviation for TLP Processing Hints
Transaction Descriptor	An element of a Packet header that, in addition to Address, Length, and Type, describes the properties of the Transaction.
Transaction ID	A component of the Transaction Descriptor including Requester ID and Tag.
Transaction Layer	The Layer that operates at the level of transactions (for example, read, write).
Transaction Layer Packet, TLP	A Packet generated in the Transaction Layer to convey a Request or Completion.
transaction sequence	A single Request and zero or more Completions associated with carrying out a single logical transfer by a Requester.
Transceiver	The physical Transmitter and Receiver pair on a single chip.
Transmitter, Tx	The component sending Packet information across a Link.
Transmitting Port	In the context of a specific TLP or DLLP, the Port that transmits the Packet on a given Link.
Type 0 Function	Function with a Type 0 Configuration Space Header (see Section 7.5.1.2).
Type 1 Function	Function with a Type 1 Configuration Space Header (see Section 7.5.1.3).
Unconditional Swap, Swap	An AtomicOp where a specified value is written to a target location, and the original value of the location is returned.
Unit Interval, UI	Given a data stream of a repeating pattern of alternating 1 and 0 values, the *Unit Interval* is the value measured by averaging the time interval between voltage transitions, over a time interval long enough to make all intentional frequency modulation of the source clock negligible.
Unsupported Request, UR	1. A status that applies to a posted or non-posted Request that specifies some action or access to some space that is not supported by the Completer. 2. A status indication returned with a Completion for a non-posted Request that suffered an Unsupported Request at the Completer.
Upstream	1. The relative position of an interconnect/System Element (Port/component) that is closer to the Root Complex. The Port on a Switch that is closest topologically to the Root Complex is the *Upstream* Port. The Port on a component that contains only Endpoint or Bridge Functions is an *Upstream* Port. The *Upstream* component on a Link is the component closer to the Root Complex. 2. A direction of information flow where the information is flowing towards the Root Complex.
Upstream Path	The flow of data through a Retimer from the Downstream Pseudo Port Receiver to the Upstream Pseudo Port Transmitter.
variant	A field of a TLP header that contains a value that is subject to possible modification according to the rules of this specification as the TLP flows through the PCI Express fabric.
Virtual Function (VF)	A Function that is associated with a Physical Function. A VF shares one or more physical resources, such as a Link, with the Physical Function and other VFs that are associated with the same PF.

Virtualization Intermediary (VI)	A software component supporting one or more SIs–colloquially known as a hypervisor or virtual machine monitor. Specification of the behavior and architecture of the VI is outside the scope of this specification.
wakeup	An optional mechanism used by a component to request the reapplication of main power when in the L2 Link state. Two such mechanisms are defined: Beacon (using in-band signaling) and WAKE# (using sideband signaling).
warm reset	A Fundamental Reset without cycling main power.

Reference Documents

PCI Express Card Electromechanical Specification, Revision 4.0

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

PCI Express Mini Card Electromechanical Specification, Revision 2.1

PCI Express OCuLink Specification, Revision 1.0

PCI Express M.2 Specification, Revision 1.1

PCI Express External Cabling Specification, Revision 2.0

PCI Express ExpressModule Electromechanical Specification, Revision 1.0

PCI-X Addendum to the PCI Local Bus Specification, Revision 2.0

PCI Hot-Plug Specification, Revision 1.1

PCI Standard Hot-Plug Controller and Subsystem Specification, Revision 1.0

PCI Code and ID Assignment Specification, Revision 1.9 (or later)

PCI Firmware Specification, Revision 3.2

Advanced Configuration and Power Interface Specification, Revision 6.2

Unified Extensible Firmware Interface (UEFI) Specification Version 2.7 Errata A

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

Multi-Root I/O Virtualization and Sharing Specification, Revision 1.0

1

1 Introduction

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

1.1 A Third Generation I/O Interconnect

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

- **Supports multiple market segments and emerging applications:**
  - Unifying I/O architecture for desktop, mobile, workstation, server, communications platforms, and embedded devices
- **Ability to deliver low cost, high volume solutions:**
  - Cost at or below PCI cost structure at the system level
- **Support multiple platform interconnect usages:**
  - Chip-to-chip, board-to-board via connector or cabling
- **A variety of mechanical form factors:**
  - M.2, Card Electro-Mechanical (CEM), U.2, OCuLink
- **PCI-compatible software model:**
  - Ability to enumerate and configure PCI Express hardware using PCI system configuration software implementations with no modifications
  - Ability to boot existing operating systems with no modifications
  - Ability to support existing I/O device drivers with no modifications
  - Ability to configure/enable new PCI Express functionality by adopting the PCI configuration paradigm

- **Performance:**
  - Low-overhead, low-latency communications to maximize application payload bandwidth and Link efficiency
  - High-bandwidth per pin to minimize pin count per device and connector interface
  - Scalable performance via aggregated Lanes and signaling frequency
- **Advanced features:**
  - Comprehend different data types and ordering rules
  - Power management and budgeting
    - Ability to identify power management capabilities of a given Function
    - Ability to transition a Function into a specific power state
    - Ability to receive notification of the current power state of a Function
    - Ability to generate a request to wakeup from a power-off state of the main power supply
    - Ability to sequence device power-up to allow graceful platform policy in power budgeting
  - Ability to support differentiated services, i.e., different (QoS)
    - Ability to have dedicated Link resources per QoS data flow to improve fabric efficiency and effective application-level performance in the face of head-of-line blocking
    - Ability to configure fabric QoS arbitration policies within every component
    - Ability to tag end-to-end QoS with each packet
    - Ability to create end-to-end isochronous (time-based, injection rate control) solutions
  - Hot-Plug support
    - Ability to support existing PCI Hot-Plug solutions
    - Ability to support native Hot-Plug solutions (no sideband signals required)
    - Ability to support async removal
    - Ability to support a unified software model for all form factors
  - Data Integrity
    - Ability to support Link-level data integrity for all types of transaction and Data Link packets
    - Ability to support end-to-end data integrity for high availability solutions
  - Error handling
    - Ability to support PCI-level error handling

        - Ability to support advanced error reporting and handling to improve fault isolation and recovery solutions
    - Process Technology Independence
        - Ability to support different DC common mode voltages at Transmitter and Receiver
    - Ease of Testing
        - Ability to test electrical compliance via simple connection to test equipment

1.2 PCI Express Link

A Link represents a dual-simplex communications channel between two components. The fundamental PCI Express Link consists of two, low-voltage, differentially driven signal pairs: a Transmit pair and a Receive pair as shown in Figure 1-1. A PCI Express Link consists of a PCIe PHY as defined in Chapter 4.

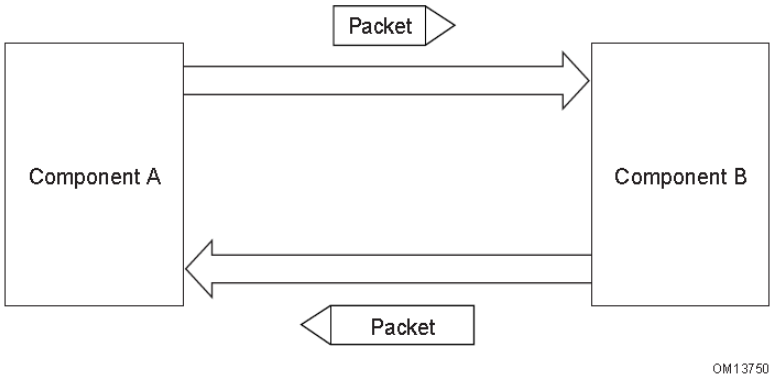


Figure 1-1: PCI Express Link

The primary Link attributes for PCI Express Link are:

- The basic Link – PCI Express Link consists of dual unidirectional differential Links, implemented as a Transmit pair and a Receive pair. A data clock is embedded using an encoding scheme (see Chapter 4) to achieve very high data rates.
- Signaling rate – Once initialized, each Link must only operate at one of the supported signaling levels. For the first generation of PCI Express technology, there is only one signaling rate defined, which provides an effective 2.5 Gigabits/second/Lane/direction of raw bandwidth. The second generation provides an effective 5.0 Gigabits/second/Lane/direction of raw bandwidth. The third generation provides an effective 8.0 Gigabits/second/Lane/direction of raw bandwidth. The fourth generation provides an effective 16.0 Gigabits/second/Lane/direction of raw bandwidth.
- Lanes – A Link must support at least one Lane – each Lane represents a set of differential signal pairs (one pair for transmission, one pair for reception). To scale bandwidth, a Link may aggregate multiple Lanes denoted by xN where N may be any of the supported Link widths. A x8 Link operating at the 2.5 GT/s data rate represents an aggregate bandwidth of

20 Gigabits/second of raw bandwidth in each direction. This specification describes operations for x1, x2, x4, x8, x12, x16, and x32 Lane widths.

- Initialization – During hardware initialization, each PCI Express Link is set up following a negotiation of Lane widths and frequency of operation by the two agents at each end of the Link. No firmware or operating system software is involved.
- Symmetry – Each Link must support a symmetric number of Lanes in each direction, i.e., a x16 Link indicates there are 16 differential signal pairs in each direction.

1.3 PCI Express Fabric Topology

A fabric is composed of point-to-point Links that interconnect a set of components – an example fabric topology is shown in Figure 1-2. This figure illustrates a single fabric instance referred to as a Hierarchy – composed of a Root Complex (RC), multiple Endpoints (I/O devices), a Switch, and a PCI Express to PCI/PCI-X Bridge, all interconnected via PCI Express Links.

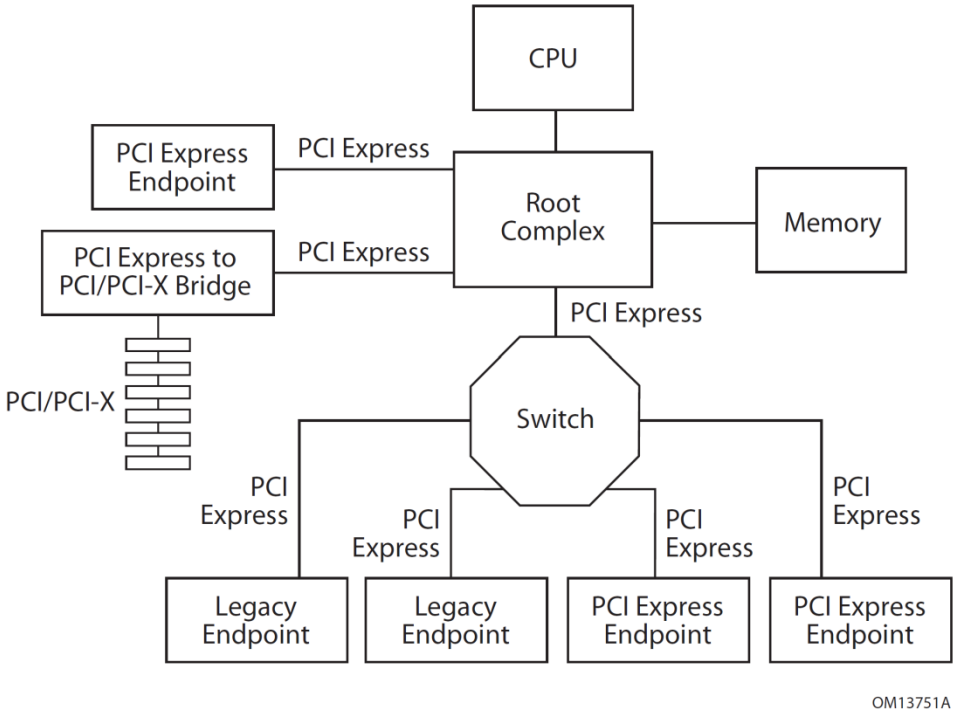


Figure 1-2: Example Topology

1.3.1 Root Complex

- An RC denotes the root of an I/O hierarchy that connects the CPU/memory subsystem to the I/O.
- As illustrated in Figure 1-2, an RC may support one or more PCI Express Ports. Each interface defines a separate hierarchy domain. Each hierarchy domain may be composed of a single Endpoint or a sub-hierarchy containing one or more Switch components and Endpoints.

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

  Unlike the rules for a Switch, an RC is generally permitted to split a packet into smaller packets when routing transactions peer-to-peer between hierarchy domains (except as noted below), e.g., split a single packet with a 256-byte payload into two packets of 128 bytes payload each. The resulting packets are subject to the normal packet formation rules contained in this specification (e.g., Max_Payload_Size, Read Completion Boundary (RCB), etc.). Component designers should note that splitting a packet into smaller packets may have negative performance consequences, especially for a transaction addressing a device behind a PCI Express to PCI/PCI-X bridge.

  Exception: An RC that supports peer-to-peer routing of Vendor_Defined Messages is not permitted to split a Vendor_Defined Message packet into smaller packets except at 128-byte boundaries (i.e., all resulting packets except the last must be an integral multiple of 128 bytes in length) in order to retain the ability to forward the Message across a PCI Express to PCI/PCI-X Bridge.

- An RC must support generation of configuration requests as a Requester.
- An RC is permitted to support the generation of I/O Requests as a Requester.

An RC is permitted to generate I/O Requests to either or both of locations 80h and 84h to a selected Root Port, without regard to that Root Port's PCI Bridge I/O decode configuration; it is recommended that this mechanism only be enabled when specifically needed.

- An RC must not support Lock semantics as a Completer.
- An RC is permitted to support generation of Locked Requests as a Requester.

1.3.2 Endpoints

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

1.3.2.1 *Legacy Endpoint Rules*

- A Legacy Endpoint must be a Function with a Type 00h Configuration Space header.
- A Legacy Endpoint must support Configuration Requests as a Completer.
- A Legacy Endpoint may support I/O Requests as a Completer.
  - A Legacy Endpoint is permitted to accept I/O Requests to either or both of locations 80h and 84h, without regard to that Endpoint's I/O decode configuration.
- A Legacy Endpoint may generate I/O Requests.

- A Legacy Endpoint may support Lock memory semantics as a Completer if that is required by the device's legacy software support requirements.
- A Legacy Endpoint must not issue a Locked Request.
- A Legacy Endpoint may implement Extended Configuration Space Capabilities, but such Capabilities may be ignored by software.
- A Legacy Endpoint operating as the Requester of a Memory Transaction is not required to be capable of generating addresses 4 GB or greater.
- A Legacy Endpoint is required to support MSI or MSI-X or both if an interrupt resource is requested. If MSI is implemented, a Legacy Endpoint is permitted to support either the 32-bit or 64-bit Message Address version of the MSI Capability structure.
- A Legacy Endpoint is permitted to support 32-bit addressing for Base Address registers that request memory resources.
- A Legacy Endpoint must appear within one of the hierarchy domains originated by the Root Complex.

1.3.2.2 *PCI Express Endpoint Rules*

- A PCI Express Endpoint must be a Function with a Type 00h Configuration Space header.
- A PCI Express Endpoint must support Configuration Requests as a Completer.
- A PCI Express Endpoint must not depend on operating system allocation of I/O resources claimed through BAR(s).
- A PCI Express Endpoint must not generate I/O Requests.
- A PCI Express Endpoint must not support Locked Requests as a Completer or generate them as a Requester. PCI Express-compliant software drivers and applications must be written to prevent the use of lock semantics when accessing a PCI Express Endpoint.
- A PCI Express Endpoint operating as the Requester of a Memory Transaction is required to be capable of generating addresses greater than 4 GB.
- A PCI Express Endpoint is required to support MSI or MSI-X or both if an interrupt resource is requested., If MSI is implemented, a PCI Express Endpoint must support the 64-bit Message Address version of the MSI Capability structure.
- A PCI Express Endpoint requesting memory resources through a BAR must set the BAR's Prefetchable bit unless the range contains locations with read side-effects or locations in which the Function does not tolerate write merging. See Section 7.5.1.2.1 for additional guidance on having the Prefetchable bit Set.
- For a PCI Express Endpoint, 64-bit addressing must be supported for all BARs that have the Prefetchable bit Set. 32-bit addressing is permitted for all BARs that do not have the Prefetchable bit Set.
- The minimum memory address range requested by a BAR is 128 bytes.
- A PCI Express Endpoint must appear within one of the hierarchy domains originated by the Root Complex.

1.3.2.3 *Root Complex Integrated Endpoint Rules*

- A Root Complex Integrated Endpoint (RCiEP) is implemented on internal logic of Root Complexes that contains the Root Ports.
- An RCiEP must be a Function with a Type 00h Configuration Space header.
- An RCiEP must support Configuration Requests as a Completer.
- An RCiEP must not require I/O resources claimed through BAR(s).
- An RCiEP must not generate I/O Requests.
- An RCiEP must not support Locked Requests as a Completer or generate them as a Requester. PCI Express-compliant software drivers and applications must be written to prevent the use of lock semantics when accessing an RCiEP.
- An RCiEP operating as the Requester of a Memory Transaction is required to be capable of generating addresses equal to or greater than the Host is capable of handling as a Completer.
- An RCiEP is required to support MSI or MSI-X or both if an interrupt resource is requested. If MSI is implemented, an RCiEP is permitted to support either the 32-bit or 64-bit Message Address version of the MSI Capability structure.
- An RCiEP is permitted to support 32-bit addressing for Base Address registers that request memory resources.
- An RCiEP must not implement Link Capabilities, Link Status, Link Control, Link Capabilities 2, Link Status 2, and Link Control 2 registers in the PCI Express Extended Capability. See Section 7.5.2 for more details.
- An RCiEP must signal PME and error conditions through the same mechanisms used on PCI systems. If a Root Complex Event Collector is implemented, an RCiEP may optionally signal PME and error conditions through a Root Complex Event Collector. In this case, an RCiEP must be associated with no more than one Root Complex Event Collector.
- An RCiEP does not implement Active State Power Management.
- An RCiEP may not be hot-plugged independent of the Root Complex as a whole.
- An RCiEP must not appear in any of the hierarchy domains exposed by the Root Complex.
- An RCiEP must not appear in Switches.

1.3.3 Switch

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

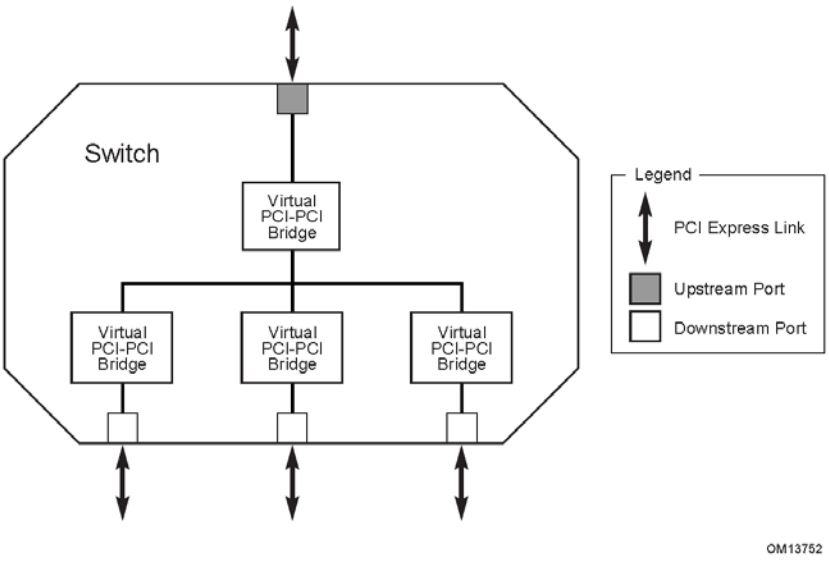


Figure 1-3: Logical Block Diagram of a Switch

- Switches appear to configuration software as two or more logical PCI-to-PCI Bridges.
- A Switch forwards transactions using PCI Bridge mechanisms; e.g., address-based routing except when engaged in a Multicast, as defined in Section 6.14.
- Except as noted in this document, a Switch must forward all types of Transaction Layer Packets (TLPs) between any set of Ports.
- Locked Requests must be supported as specified in Section 6.5. Switches are not required to support Downstream Ports as initiating Ports for Locked Requests.
- Each enabled Switch Port must comply with the Flow Control specification within this document.
- A Switch is not allowed to split a packet into smaller packets, e.g., a single packet with a 256-byte payload must not be divided into two packets of 128 bytes payload each.
- Arbitration between Ingress Ports (inbound Link) of a Switch may be implemented using round robin or weighted round robin when contention occurs on the same Virtual Channel. This is described in more detail later within the specification.
- Endpoints (represented by Type 00h Configuration Space headers) must not appear to configuration software on the Switch's internal bus as peers of the virtual PCI-to-PCI Bridges representing the Switch Downstream Ports.

1.3.4 Root Complex Event Collector

- A Root Complex Event Collector provides support for terminating error and PME messages from RCiEPs.
- A Root Complex Event Collector must follow all rules for an RCiEP.
- A Root Complex Event Collector is not required to decode any memory or I/O resources.

- A Root Complex Event Collector is identified by its Device/Port Type value (see section 7.5.3.2).
- A Root Complex Event Collector has the Base Class 08h, Sub-Class 07h and Programming Interface 00h.[2]
- A Root Complex Event Collector resides on the same Logical Bus as the RCiEPs it supports.
- Multiple Root Complex Event Collectors are permitted to reside on a single Logical Bus.
- A Root Complex Event Collector explicitly declares supported RCiEPs through the Root Complex Event Collector Endpoint Association Capability.
- Root Complex Event Collectors are optional.

1.3.5 PCI Express to PCI/PCI-X Bridge

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

1.4 Hardware/Software Model for Discovery, Configuration and Operation

The PCI/PCIe hardware/software model includes architectural constructs necessary to discover, configure, and use a Function, without needing Function-specific knowledge. Key elements include:

- A configuration model which provides system software the means to discover hardware Functions available in a system
- Mechanisms to perform basic resource allocation for addressable resources such as memory space and interrupts
- Enable/disable controls for Function response to received Requests, and initiation of Requests
- Well-defined ordering and flow control models to support the consistent and robust implementation of hardware/software interfaces.

The PCI Express configuration model supports two mechanisms:

- PCI-compatible configuration mechanism: The PCI-compatible mechanism supports 100% binary compatibility with Conventional PCI aware operating systems and their corresponding bus enumeration and configuration software.
- PCI Express enhanced configuration mechanism: The enhanced mechanism is provided to increase the size of available Configuration Space and to optimize access mechanisms.

Each PCI Express Link is mapped through a virtual PCI-to-PCI Bridge structure and has a logical PCI bus associated with it. The virtual PCI-to-PCI Bridge structure may be part of a PCI Express Root Complex Port, a Switch Upstream Port, or a Switch Downstream Port. A Root Port is a

[2] Since an earlier version of this specification used Sub-Class 06h for this purpose, an implementation is still permitted to use Sub-Class 06h, but this is strongly discouraged.

virtual PCI-to-PCI Bridge structure that originates a PCI Express hierarchy domain from a PCI Express Root Complex. Devices are mapped into Configuration Space such that each will respond to a particular Device Number.

1.5 PCI Express Layering Overview

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

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

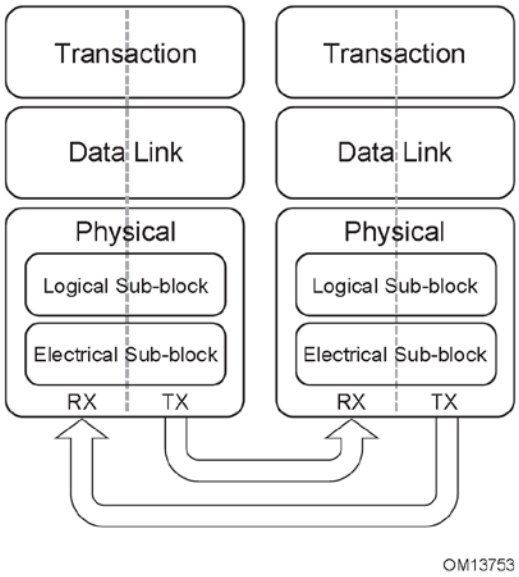


Figure 1-4: High-Level Layering Diagram

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

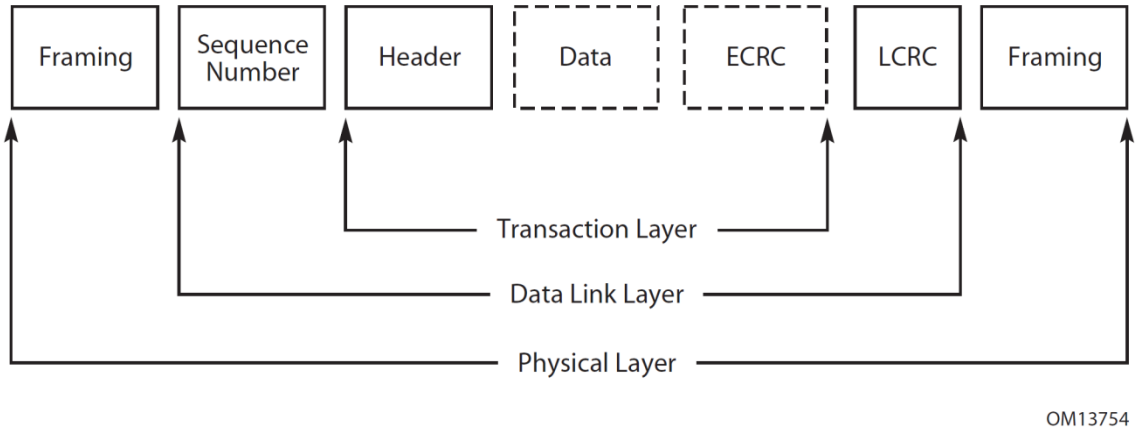


Figure 1-5: Packet Flow Through the Layers

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

1.5.1 Transaction Layer

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

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

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

1.5.2 Data Link Layer

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

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

retransmission of TLPs until information is correctly received, or the Link is determined to have failed.

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

1.5.3 Physical Layer

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

.

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

1.5.4 Layer Functions and Services

1.5.4.1 *Transaction Layer Services*

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

Initialization and configuration functions require the Transaction Layer to:

- Store Link configuration information generated by the processor or management device
- Store Link capabilities generated by Physical Layer hardware negotiation of width and operational frequency

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

- Generate TLPs from device core Requests
- Convert received Request TLPs into Requests for the device core
- Convert received Completion Packets into a payload, or status information, deliverable to the core
- Detect unsupported TLPs and invoke appropriate mechanisms for handling them
- If end-to-end data integrity is supported, generate the end-to-end data integrity CRC and update the TLP header accordingly.

Flow Control services:

- The Transaction Layer tracks Flow Control credits for TLPs across the Link.
- Transaction credit status is periodically transmitted to the remote Transaction Layer using transport services of the Data Link Layer.
- Remote Flow Control information is used to throttle TLP transmission.

Ordering rules:

- PCI/PCI-X compliant producer/consumer ordering model
- Extensions to support Relaxed Ordering
- Extensions to support ID-Based Ordering

Power management services:

- Software-controlled power management through mechanisms, as dictated by system software.
- Hardware-controlled autonomous power management minimizes power during full-on power states.

Virtual Channels and Traffic Class:

- The combination of Virtual Channel mechanism and Traffic Class identification is provided to support differentiated services and QoS support for certain classes of applications.
- Virtual Channels: Virtual Channels provide a means to support multiple independent logical data flows over given common physical resources of the Link. Conceptually this involves multiplexing different data flows onto a single physical Link.
- Traffic Class: The Traffic Class is a Transaction Layer Packet label that is transmitted unmodified end-to-end through the fabric. At every service point (e.g., Switch) within the fabric, Traffic Class labels are used to apply appropriate servicing policies. Each Traffic Class label defines a unique ordering domain - no ordering guarantees are provided for packets that contain different Traffic Class labels.

1.5.4.2 *Data Link Layer Services*

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

Initialization and power management services:

- 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 *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
- Modified Compliance pattern

1.5.4.4 *Inter-Layer Interfaces*

1.5.4.4.1 Transaction/Data Link Interface

The Transaction to Data Link interface provides:

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

The Data Link to Transaction interface provides:

- Byte or multi-byte data received from the PCI Express Link
- TLP framing information for the received byte

- Actual power state for the Link
- Link status information

1.5.4.4.2 Data Link/Physical Interface

The Data Link to Physical interface provides:

- Byte or multi-byte wide data to be sent across the Link
    - Data transfer handshake mechanism
    - TLP and DLLP boundary information for bytes
- Requested power state for the Link

The Physical to Data Link interface provides:

- Byte or multi-byte wide data received from the PCI Express Link
- TLP and DLLP framing information for data
- Indication of errors detected by the Physical Layer
- Actual power state for the Link
- Connection status information

2

2 Transaction Layer Specification

2.1 Transaction Layer Overview

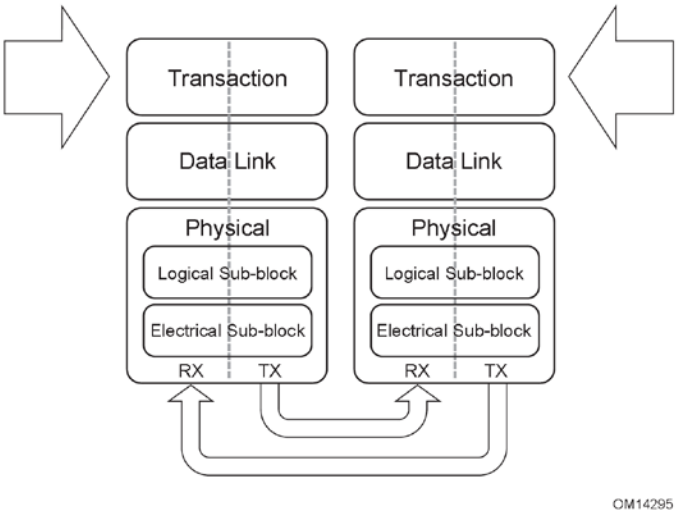


Figure 2-1: Layering Diagram Highlighting the Transaction Layer

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
- 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 2-12.

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 *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 PASID TLP Prefix containing the Process Address Space ID (PASID). See Section 6.20 for details.

2.1.1.2 *I/O Transactions*

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

- Read Request/Completion
- Write Request/Completion

I/O Transactions use a single address format:

- Short Address Format: 32-bit address

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 *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 Sections 2.2.8.5 and 2.2.8.7).

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

2.1.2 Packet Format Overview

Transactions consist of Requests and Completions, which are communicated using packets. Figure 2-2 shows a high level serialized view of a 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 shows a more detailed view of the TLP. The following sections of this chapter define the detailed structure of the packet headers and digest.

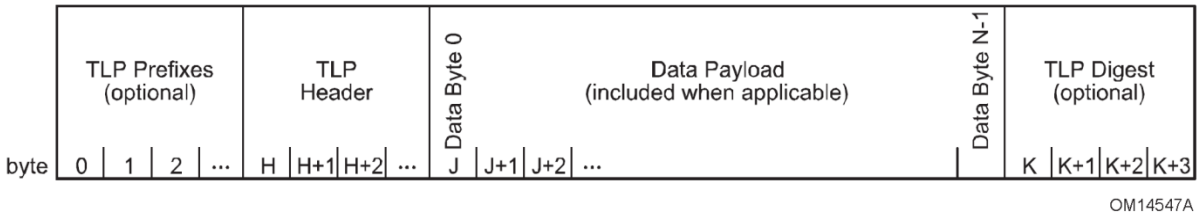


Figure 2-2: Serial View of a TLP

PCI Express conceptually transfers information as a serialized stream of bytes as shown in Figure 2-2. Note that at the byte level, information is transmitted/received over the interconnect with the left-most byte of the TLP as shown in Figure 2-2 being transmitted/received first (byte 0 if one or more optional TLP Prefixes are present else byte H). Refer to Section 4.2 for details on how individual bytes of the packet are encoded and transmitted over the physical media.

Detailed layouts of the TLP Prefix, TLP Header and TLP Digest (presented in generic form in Figure 2-3) are drawn with the lower numbered bytes on the left rather than on the right as has traditionally been depicted in other PCI specifications. The 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.

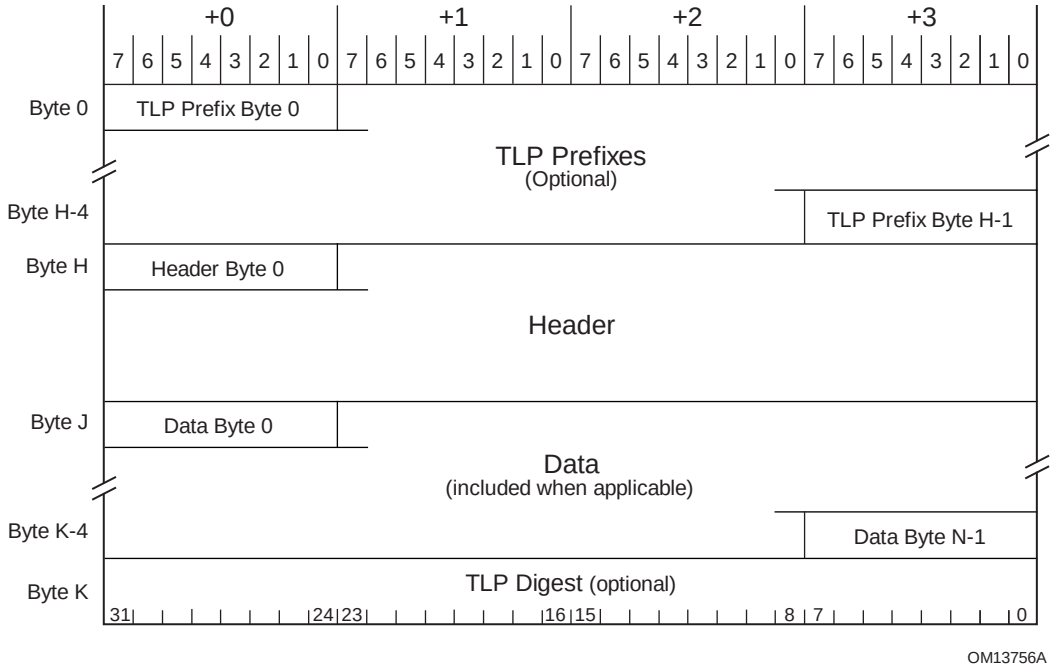


Figure 2-3: Generic TLP Format

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

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

- Format of the packet
- Type of the packet
- Length for any associated data
- Transaction Descriptor, including:
  - Transaction ID
  - Attributes
  - Traffic Class
- Address/routing information
- 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):

- Fmt[2:0] – Format of TLP (see Table 2-2) – bits 7:5 of byte 0
- Type[4:0] – Type of TLP – bits 4:0 of byte 0

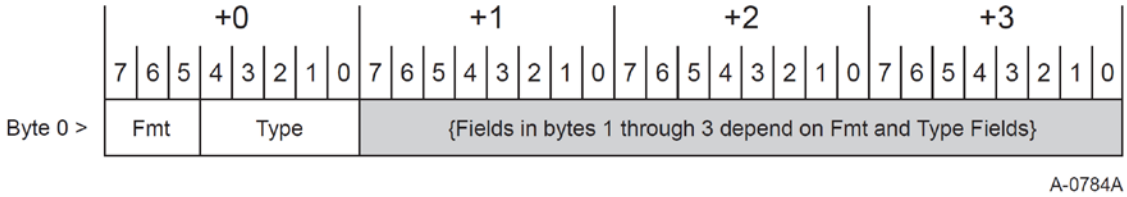


Figure 2-4: Fields Present in All TLPs

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 determines how the TLP is routed by a Switch. Different types of TLPs are discussed in more detail in the following sections.

- Permitted Fmt[2:0] and Type[4:0] field values are shown in Table 2-3.
    - All other encodings are Reserved (see Section 2.3).
- TC[2:0] – Traffic Class (see Section 2.2.6.6) – bits [6:4] of byte 1

- 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 TLP Prefix (if present) – bit 0 of byte 1 (see Section 2.2.7.1)
- Attr[1:0] – Attributes (see Section 2.2.6.3) – bits [5:4] of byte 2
- Attr[2] – Attribute (see Section 2.2.6.3) – 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) – bit 7 of byte 2
- Error Poisoned (EP) – indicates the TLP is poisoned (see Section 2.7) – bit 6 of byte 2
- Length[9:0] – Length of data payload in DW (see Table 2-4) – bits 1:0 of byte 2 concatenated with bits 7:0 of byte 3
    - TLP data must be 4-byte naturally aligned and in increments of 4-byte DW.
    - Reserved for TLPs that do not contain or refer to data payloads, including Cpl, CplLk, and Messages (except as specified)

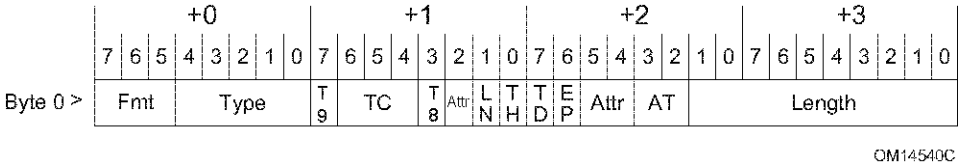


Figure 2-5: Fields Present in All TLP Headers

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).

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

[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.

TLP Type	Fmt [2:0][3] (b)	Type [4:0] (b)	Description
MRdLk	000 001	0 0001	Memory Read Request-Locked
MWr	010 011	0 0000	Memory Write Request
IORd	000	0 0010	I/O Read Request
IOWr	010	0 0010	I/O Write 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 Type[4]
TCfgWr	010	1 1011	Deprecated TLP Type[4]
Msg	001	1 $0r_2r_1r_0$	Message Request – The sub-field r[2:0] specifies the Message routing mechanism (see Table 2-17).
MsgD	011	1 $0r_2r_1r_0$	Message Request with data payload – The sub-field r[2:0] specifies the Message routing mechanism (see Table 2-17).
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	$0L_3L_2L_1L_0$	Local TLP Prefix – The sub-field L[3:0] specifies the Local TLP Prefix type (see Table 2-35).

[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.

TLP Type	Fmt [2:0][3] (b)	Type [4:0] (b)	Description
EPrfx	100	$1E_3E_2E_1E_0$	End-End TLP Prefix – The sub-field E[3:0] specifies the End-End TLP Prefix type (see Table 2-36).
			All encodings not shown above are Reserved (see Section 2.3).

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

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

2.2.2 TLPs with Data Payloads - Rules

- Length is specified as an integral number of DW
- Length[9:0] is Reserved for all Messages except those which explicitly refer to a data length
  - Refer to the Message Code tables in Section 2.2.8.
- 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_Size field of the Transmitter's Device Control register taken as an integral number of DW (see Section 7.5.3.4).
  - 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 an Upstream Port associated with a non-ARI Multi-Function Device (MFD) whose Max_Payload_Size settings are identical across all Functions, a transmitted TLP's data payload must not exceed the common Max_Payload_Size setting.
  - For an Upstream Port associated with a non-ARI Multi-Function Device whose Max_Payload_Size settings 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 the Max_Payload_Size setting from the Function that generated the transaction, or else the smallest Max_Payload_Size setting across all Functions.
    - Software should not set the Max_Payload_Size in different Functions to different values unless software is aware of the specific implementation.

- Note: Max_Payload_Size applies only to TLPs with data payloads; Memory Read Requests are not restricted in length by Max_Payload_Size. The size of the Memory Read Request is 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 the Max_Payload_Size field of the Receiver's Device Control register taken as an integral number of DW (see Section 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 (see Section 6.2).
  - 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 an Upstream Port associated with a non-ARI Multi-Function Device whose Max_Payload_Size settings are identical across all Functions, the Receiver is required to check the TLP's data payload size against the common Max_Payload_Size setting.
  - For an Upstream Port associated with a non-ARI Multi-Function Device whose Max_Payload_Size settings are not identical across all Functions, the Receiver is required to check the TLP's data payload against a Max_Payload_Size setting whose determination is implementation specific.
    - Receiver implementations are encouraged to use the Max_Payload_Size setting from the Function targeted by the transaction, or else the largest Max_Payload_Size setting across all Functions.
    - Software should not set the Max_Payload_Size in 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 (see Section 6.2).

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

- When a data payload is included in a TLP 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 in significance. 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 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.

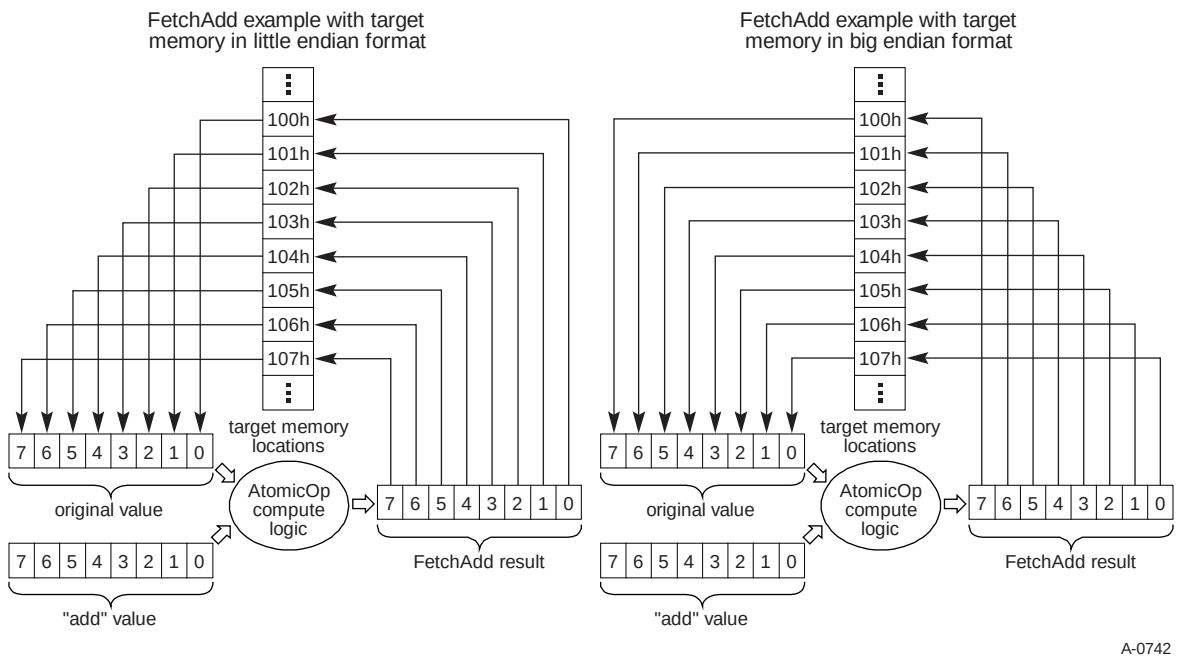


Figure 2-6: Examples of Completer Target Memory Access for FetchAdd

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.

IMPLEMENTATION NOTE

Maintaining Alignment in Data Payloads

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

2.2.3 TLP Digest Rules

- For any TLP, a value of 1b in the TD bit indicates the presence of the TLP Digest field including an end-to-end CRC (ECRC) value at the end of the TLP.
  - A TLP 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 (see Section 6.2).
- If an intermediate or ultimate PCI Express Receiver of the TLP does not support ECRC checking, the Receiver must ignore the TLP Digest[5].
  - 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.

2.2.4 Routing and Addressing Rules

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

[5] An exception is an Intermediate Receiver forwarding a Multicast TLP out an Egress Port with MC_Overlay enabled. See Section 6.14.5.

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) and a 32-bit format used with a 3 DW header (see Figure 2-8).

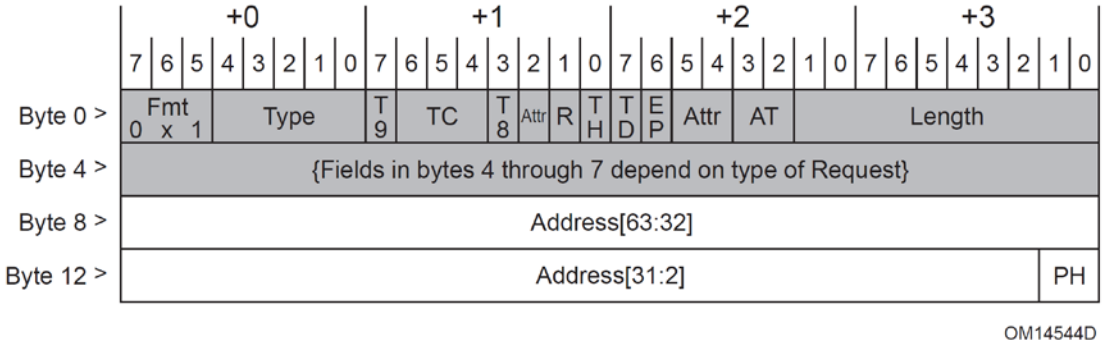


Figure 2-7: 64-bit Address Routing

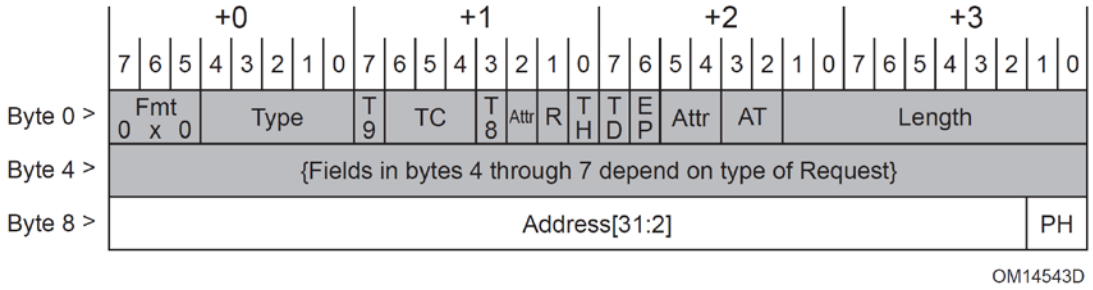


Figure 2-8: 32-bit Address Routing

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

Table 2-5: Address Field Mapping

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

Address Bits	32-bit Addressing	64-bit Addressing
23:16	Bits 7:0 of Byte 9	Bits 7:0 of Byte 13
15:8	Bits 7:0 of Byte 10	Bits 7:0 of Byte 14
7:2	Bits 7:2 of Byte 11	Bits 7:2 of Byte 15

- Memory Read, 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 *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.
  - For ARI Routing IDs, the Bus and (8-bit) Function Number to TLP header mapping is shown in Table 2-7.
- Two ID routing formats are specified, one used with a 4 DW header (see Figure 2-9) and one used with a 3 DW header (see Figure 2-10).
  - Header field locations are the same for both formats, and are given in Table 2-6 and Table 2-7.

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

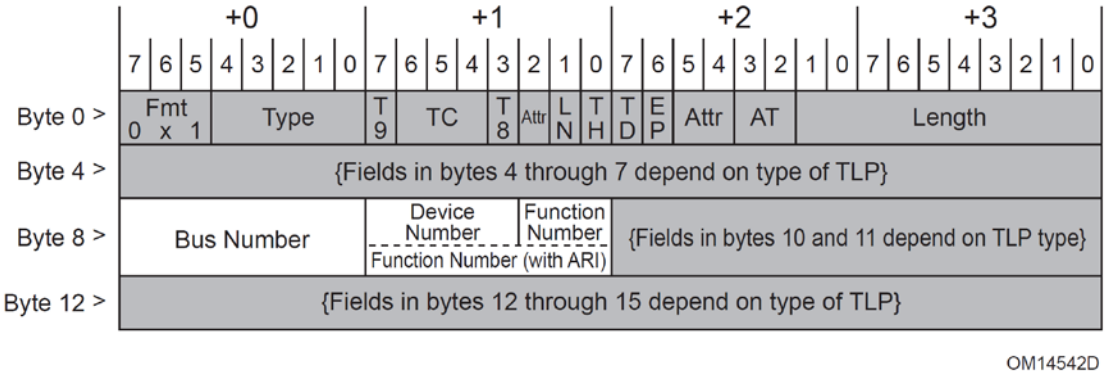


Figure 2-9: ID Routing with 4 DW Header

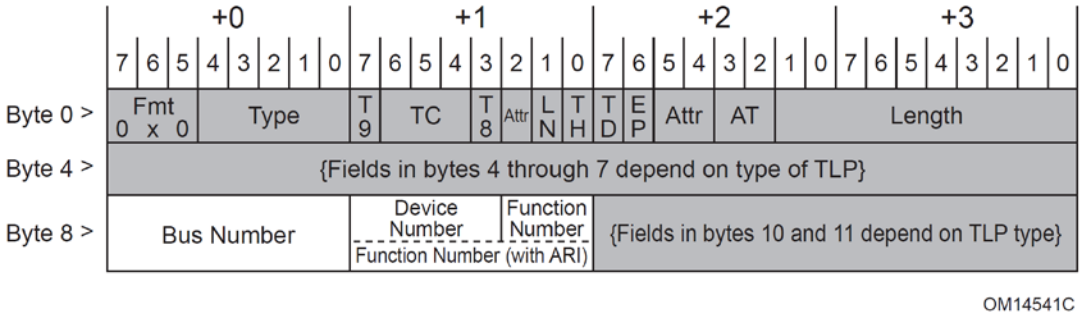


Figure 2-10: ID Routing with 3 DW Header

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 2-11). 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 2.2.7.1 for 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 2.2.7 for additional requirements.
  - If the Length field for this Request indicates a length of 1 DW, then the value for the 1st DW Byte Enables is implied to be 1111b and 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 1st DW Byte Enables and the Last DW Byte Enables is implied to be 1111b.

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

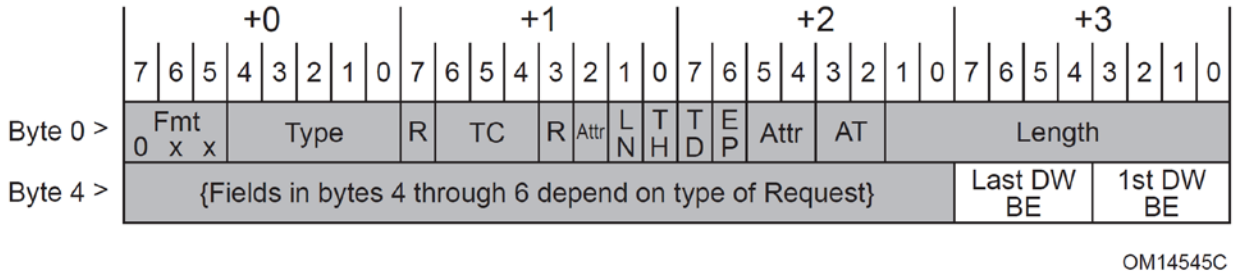


Figure 2-11: Location of Byte Enables in TLP Header

- The 1st DW BE[3:0] field contains Byte Enables for the first (or only) DW referenced by a Request.
  - If the Length field for a Request indicates a length of greater than 1 DW, this field must not equal 0000b.
- The Last DW BE[3:0] field contains Byte Enables for the last DW of a Request.
  - 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 1st 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:

    1st DW BE: 1100b, Last DW BE: 0011b
    1st DW BE: 1000b, Last DW BE: 0111b

- Table 2-8 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 Byte[6]
1st DW BE[0]	Bit 0 of Byte 7	Byte 0
1st DW BE[1]	Bit 1 of Byte 7	Byte 1
1st DW BE[2]	Bit 2 of Byte 7	Byte 2
1st DW BE[3]	Bit 3 of Byte 7	Byte 3
Last DW BE[0]	Bit 4 of Byte 7	Byte N-4
Last DW BE[1]	Bit 5 of Byte 7	Byte N-3
Last DW BE[2]	Bit 6 of Byte 7	Byte N-2
Last DW BE[3]	Bit 7 of Byte 7	Byte N-1

- 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.

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 (1st DW BE[3:0] field = 0000b), the corresponding Completion must specify a Length of 1 DW, and include a data payload of 1 DW

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.

[6] 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.

- 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 *Overview*

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

- Transaction ID – identifies outstanding Transactions
- Attributes field – specifies characteristics of the Transaction
- Traffic Class (TC) field – associates Transaction with type of required service

Figure 2-12 shows the fields of the Transaction Descriptor. Note that these fields are shown together to highlight their relationship as parts of a single logical entity. The fields are not contiguous in the packet header.

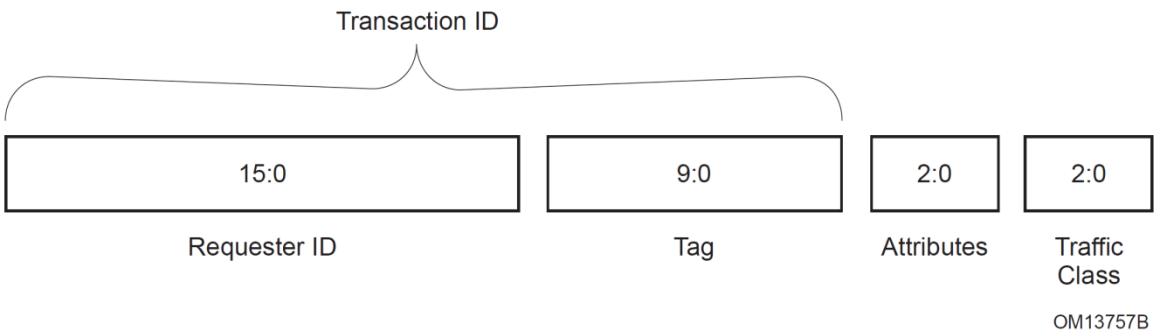


Figure 2-12: Transaction Descriptor

2.2.6.2 *Transaction Descriptor – Transaction ID Field*

The Transaction ID field consists of two major sub-fields: Requester ID and Tag as shown in Figure 2-13.

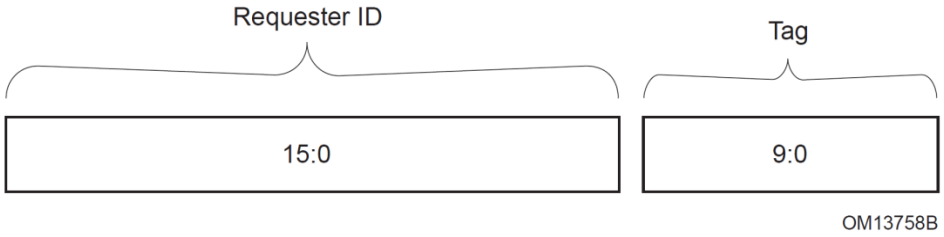


Figure 2-13: Transaction ID

10-Bit Tag capability, introduced in *PCI Express Base Specification, Revision 4.0,* increases the total Tag field size from 8 bits to 10 bits. The two additional Tag bits, Tag[8] and Tag[9], 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.
  - Functions[7] (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 7.5.3.15 and 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.

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

    - Each RP indicating support must handle such Requests received by its Ingress Port.
    - 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 Function shall 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 Endpoint[8] 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

[8] This includes PCI Express Endpoints, Legacy PCI Express Endpoints, and Root Complex Integrated Endpoints.

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 Completions[9], 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.
- 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 2.2.7.1 for details). For Posted Requests with the TH bit Clear, the Tag[7:0] field is undefined and may contain any value. (Refer to Table F-1 for exceptions to this rule for certain Vendor_Defined Messages.)

[9] If a Completion has a higher precedence error, that error should be reported instead.

- 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 Numbers[10] 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 ID[11] 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 Number[12] 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 (see Section 7.1) causing the Request generation.
- 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.

---

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

[11] An ARI Requester ID does not contain a Device Number field. See Section 2.2.4.2.

[12] With ARI Devices, only the Bus Number can change.

- 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 (see Section 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 (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.

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 Functions[13] 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 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.

---

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

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.

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 *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 2-15 and Figure 2-16).

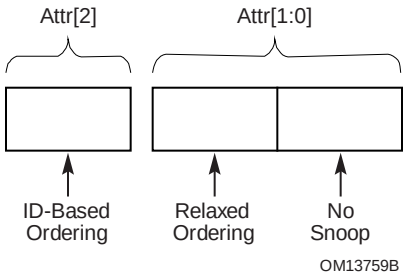


Figure 2-14: Attributes Field of Transaction Descriptor

2.2.6.4 *Relaxed Ordering and ID-Based Ordering Attributes*

Table 2-9 defines the states of the Relaxed Ordering and ID-Based Ordering attribute fields. These attributes are discussed in Section 2.4. Note that Relaxed Ordering and ID-Based Ordering attributes are not adjacent in location (see Figure 2-5).

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 *No Snoop Attribute*

Table 2-10 defines the states of the No Snoop attribute field. Note that the No Snoop attribute does not alter Transaction ordering.

Table 2-10: Cache Coherency Management Attribute

No Snoop 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).

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 covers the details of the VC mechanism.

Table 2-11 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 1st 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 1st 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 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 2-15 and Figure 2-16).
- 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. 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 (see Section 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 (see Section 6.2).

- Requests must not specify an Address/Length combination which causes a Memory Space access to cross a 4-KB boundary.
  - Receivers may optionally check for violations of this rule. If a Receiver implementing this check determines that a TLP violates this rule, the TLP is a Malformed TLP.
    - If checked, this is a reported error associated with the Receiving Port (see Section 6.2).
  - 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.

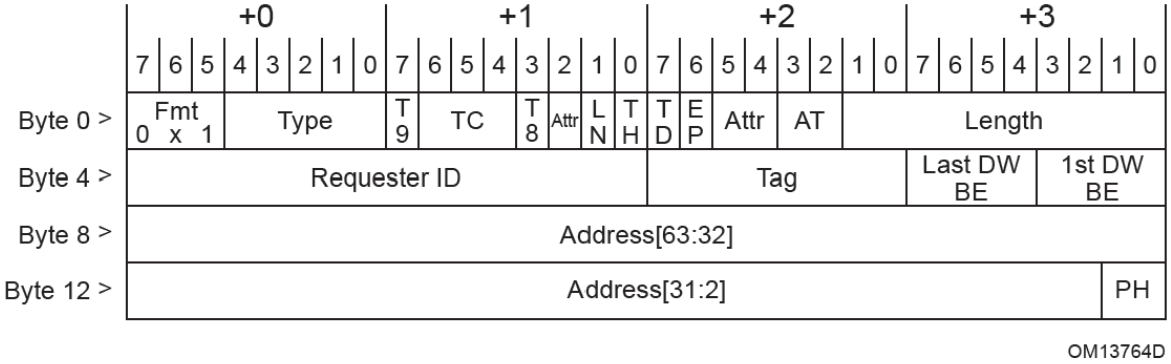


Figure 2-15: Request Header Format for 64-bit Addressing of Memory

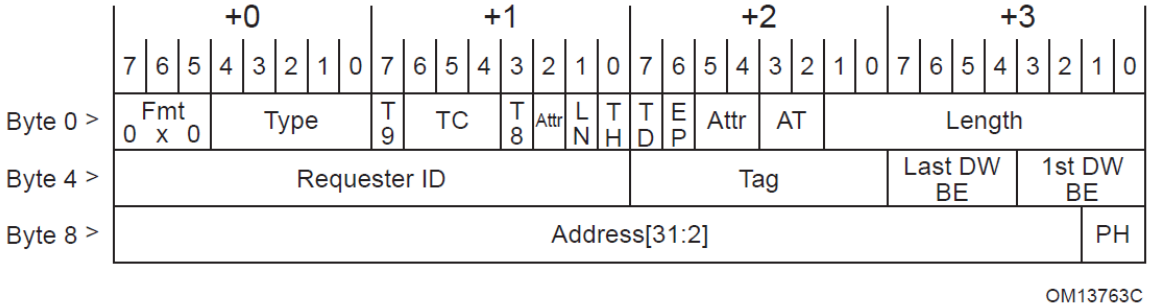


Figure 2-16: Request Header Format for 32-bit Addressing of Memory

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 2-17)
- 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.
  - 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 (see Section 6.2).

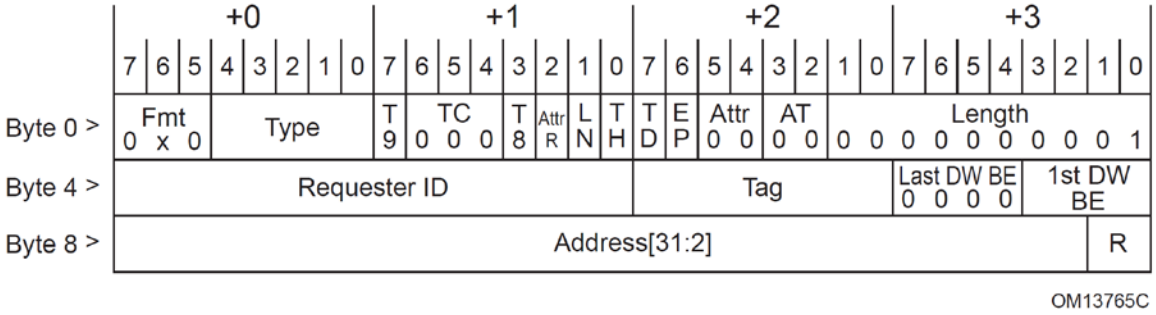


Figure 2-17: Request Header Format for I/O Transactions

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 2-18).
  - 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 (see Section 6.2).

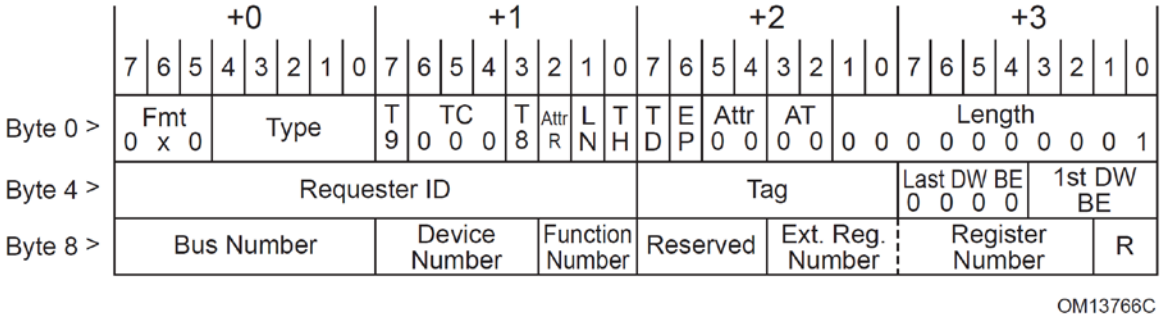


Figure 2-18: Request Header Format for Configuration Transactions

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 *TPH Rules*

- Two formats are specified for TPH. The Baseline TPH format (see Figure 2-20 and Figure 2-21) must be used for all Requests that provide TPH. The format with the optional TPH TLP Prefix extends the TPH fields (see Figure 2-19) to provide additional bits for the Steering Tag (ST) field.

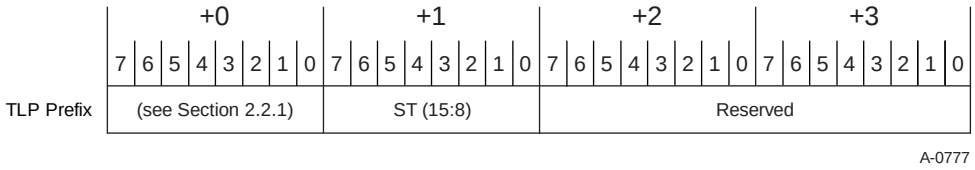


Figure 2-19: TPH TLP Prefix

- 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 (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
  - The TH bit is not applicable and is Reserved for all other Requests.
- The Processing Hints (PH) fields mapping is shown in Figure 2-20, Figure 2-21 and Table 2-14.

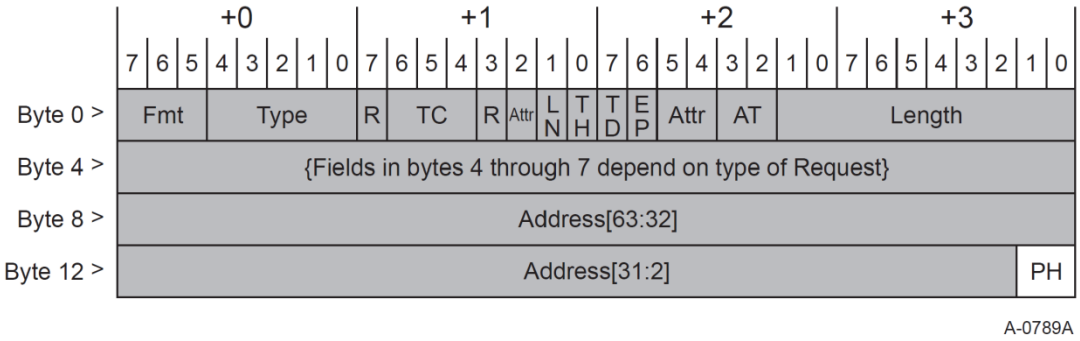


Figure 2-20: Location of PH[1:0] in a 4 DW Request Header

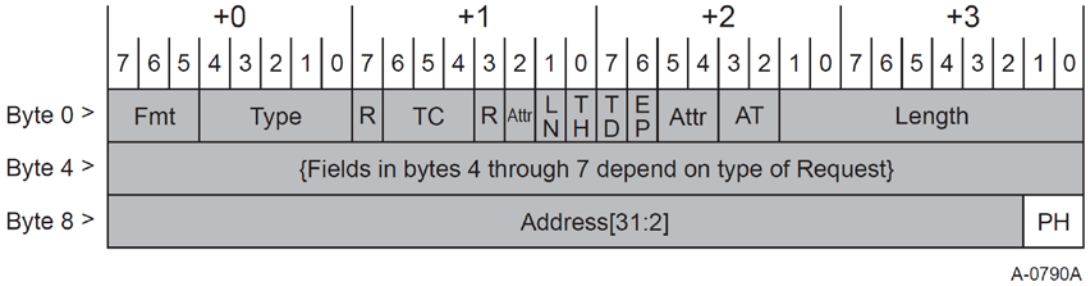


Figure 2-21: Location of PH[1:0] in a 3 DW Request Header

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.

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
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 2-22, Figure 2-23 and Table 2-16.

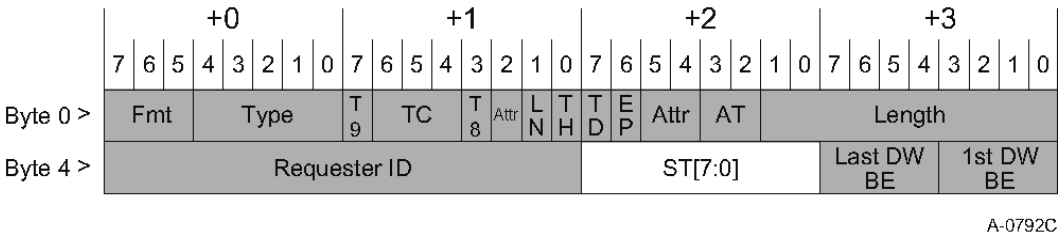


Figure 2-22: Location of ST[7:0] in the Memory Write Request Header

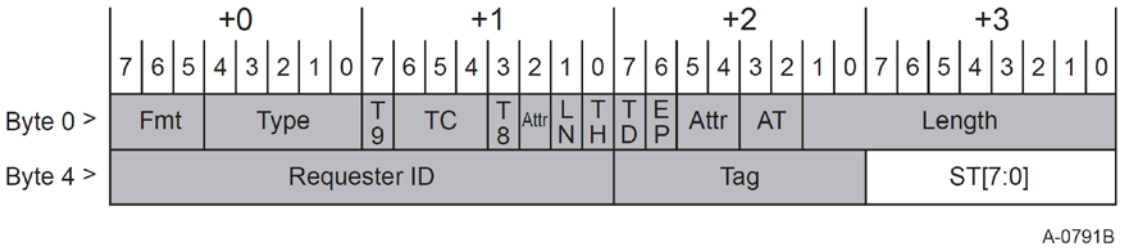


Figure 2-23: Location of ST[7:0] in Memory Read and AtomicOp Request Headers

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
- 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 2-34):
  - 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 the Msg 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.

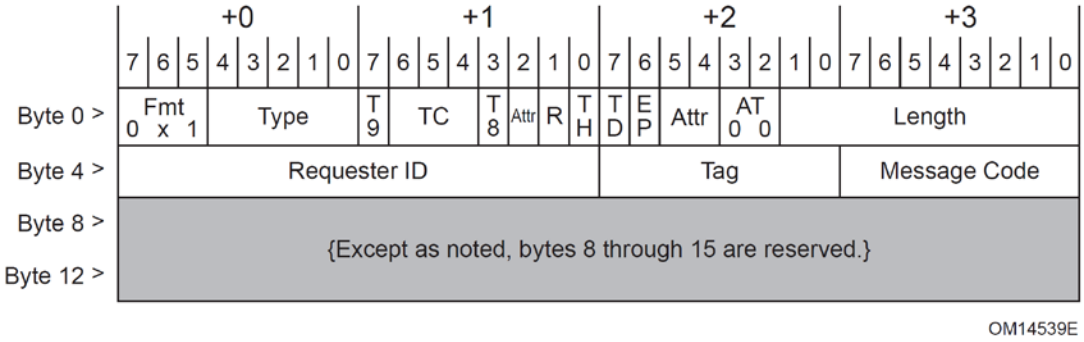


Figure 2-24: Message Request Header

In addition to address and ID routing, Messages support several other routing mechanisms. These mechanisms are referred to as "implicit" because no address or ID specifies the destination, 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
  - Permitted values are defined in the following sections for each Message

Table 2-17: Message Routing

r[2:0] (b)	Description	Bytes 8 to 15[14]
000	Routed to Root Complex	Reserved
001	Routed by Address[15]	Address
010	Routed by ID	See Section 2.2.4.2
011	Broadcast from Root Complex	Reserved
100	Local - Terminate at Receiver	Reserved
101	Gathered and routed to Root Complex[16]	Reserved
110 to111	Reserved - Terminate at Receiver	Reserved

[14] Except as noted, e.g., Vendor_Defined Messages.

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

[16] This routing type is used only for PME_TO_Ack, and is described in Section 5.6.4.2.1.

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 (see Section 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).
- 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 and ARI 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 (see Section 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 (see Section 6.2).

Table 2-18: INTx Mechanism Messages

Name	Code[7:0] (b)	Routing r[2:0] (b)	Support[17]				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	
Deassert_INTD	0010 0111	100	All:				Deassert INTD virtual wire
			r		tr		
			As Required:				
				t		t	

[17] Abbreviations:
RC = Root Complex
Sw = Switch (only used with “Link” routing)
Ep = Endpoint
Br = PCI Express (primary) to PCI/PCI-X (secondary) Bridge
r = Supports as Receiver
t = Supports as Transmitter

Note that Switches must support passing Messages on all legal routing paths. Only Messages specifying Local (100b) routing or a Reserved routing value are terminated locally at the Receiving Port on a Switch.

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

- The components at both ends of each Link must track the logical state of the four virtual wires using the Assert/Deassert Messages to represent the active and inactive transitions (respectively) of each corresponding virtual wire.
  - An Assert_INTx represents the active going transition of the INTx (x = A, B, C, or D) virtual wire
  - A Deassert_INTx represents the inactive going transition of the INTx (x = A, B, C, or D) virtual wire
- When the local logical state of an INTx virtual wire changes at an Upstream Port, the Port must communicate this change in state to the Downstream Port on the other side of the same Link using the appropriate Assert_INTx or Deassert_INTx Message.

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

- INTx Interrupt Signaling is disabled when the Interrupt Disable bit of the Command register (see Section 7.5.1.1.3) is Set.
  - Any INTx virtual wires that are active when the Interrupt Disable bit is set must be deasserted by transmitting the appropriate Deassert_INTx Message(s).
- Virtual and actual PCI to PCI Bridges must map the virtual wires tracked on the secondary side of the Bridge according to the Device Number of the device on the secondary side of the Bridge, as shown in Table 2-19.
- 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 be deasserted, and any associated system interrupt resource request(s) must be discarded.

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 Source[18]) **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

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

IMPLEMENTATION NOTE

System Interrupt Mapping

Note that system software (including BIOS and operating system) needs to comprehend the remapping of legacy interrupts (INTx mechanism) in the entire topology of the system (including hierarchically connected Switches and subordinate PCI Express/PCI Bridges) to establish proper correlation between PCI Express device interrupt and associated interrupt resources in the system interrupt controller. The remapping described by Table 2-19 is applied hierarchically at every Switch. In addition, PCI Express/PCI and PCI/PCI Bridges perform a similar mapping function.

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 *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 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 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 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 and ARI 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 (see Section 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
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 *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 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 and ARI 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 (see Section 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 to Section 6.2 for details on uses for these Messages.

2.2.8.4 *Locked Transactions Support*

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

- Table 2-22 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 (see Section 6.2).

Table 2-22: 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 *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 2-23 defines the Set_Slot_Power_Limit Message.
- The Set_Slot_Power_Limit Message includes a 1 DW data payload (TLP Type is MsgD).
- The Set_Slot_Power_Limit Message must use the default Traffic Class designator (TC0). Receivers must check for violations of this rule. If a Receiver determines that a TLP violates this rule, it must handle the TLP as a Malformed TLP.
    - This is a reported error associated with the Receiving Port (see Section 6.2).

Table 2-23: 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 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 be Set to all 0 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 (see Section 7.5.3.9) when the Data Link Layer reports DL_Up status.
- Any time when a Link transitions from a non-DL_Up status to a DL_Up status (see Section 2.9.2) and the Auto Slot Power Limit Disable bit is Cleared 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 adapteris 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.

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 2-24) use the header format shown in Figure 2-25.
  - The Requester ID is implementation specific. It is strongly recommended that the Requester ID field contain the value associated with the Requester.[19]
  - 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 2-24: 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.

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.

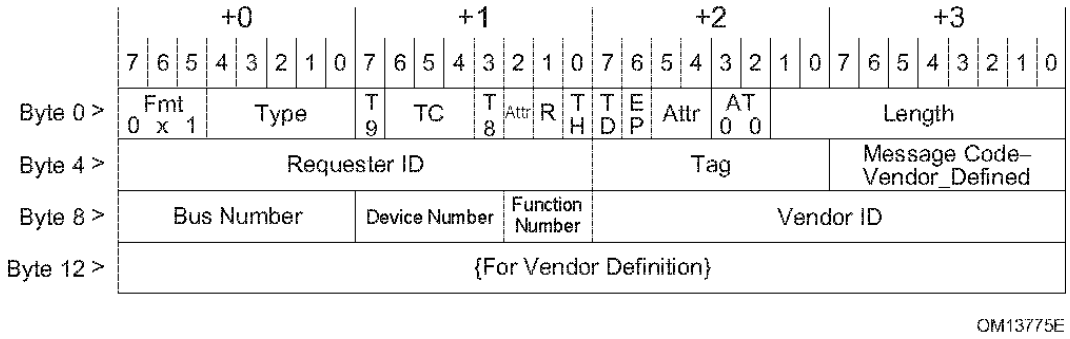


Figure 2-25: Header for Vendor-Defined Messages

[19] ACS Source Validation (see Section 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.

- 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)
- 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 which 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 to Section 6.2.

PCI Express to PCI/PCI-X Bridge Specification, Revision 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 *PCI Express to PCI/PCI-X Bridge Specification, Revision 1.0* for details. Refer to Section 2.2.6.2 for 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 2-26.
- 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 F for a list of PCI-SIG-Defined VDMs.

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0 >	Fmt 0x1			Type					T 9	TC			T 8	Attr	L N	T H	T D	E P	Attr 00		R				Length							
Byte 4 >	Requester ID																Tag								Message Code 0111 1111							
Byte 8 >	Destination ID if ID Routed; otherwise Reserved																Vendor ID 0000 0000 0000 0001															
Byte 12 >	Subtype								{For PCI-SIG-Defined VDM Definition}																							

Figure 2-26: Header for PCI-SIG-Defined VDMs

2.2.8.6.2 LN Messages

LN protocol (see 6.21) defines 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 2-26 and Figure 2-27 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 (IDO) bit, is not Reserved.
- Attr[1], the Relaxed Ordering (RO) bit, is not Reserved.
- Attr[0], the No Snoop 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 cacheline 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 2-25, indicating the specific reason that the LN Message was sent. These encodings apply to both the directed and broadcast versions of LN Messages.

Table 2-25: 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

Table 2-26: 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 2-27 below.

+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
Fmt 0 1 1			Type 1 0 r_2 r_1 r_0					R	TC 0 0 0			R	Attr	L N	T H	T D	E P	Attr 0 0		AT 0 0		Length 00 0000 0010b									
Requester ID																Reserved								Message Code 0111 1111b							
Destination ID (if directed) or Reserved (if broadcast)																Vendor ID 0000 0000 0000 0001b															
Subtype 0000 0000b								Reserved																							
Cacheline Address [63:32]																															
Cacheline Address [31:6]																								Reserved						NR	

Figure 2-27: 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-Defined VDM mechanism (see Section 2.2.8.6.1). The DRS Message is a PCI-SIG-Defined VDM (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 2-27 and Figure 2-28 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 (see Section 6.2).

Table 2-27: 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 2-28 below:

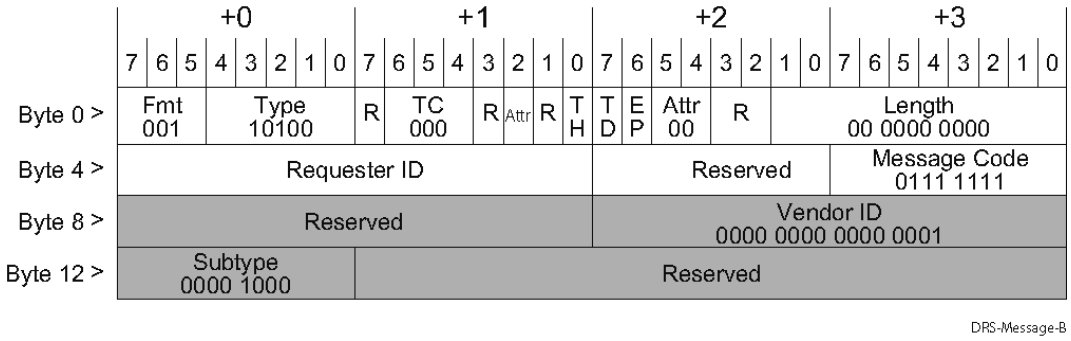


Figure 2-28: DRS Message

2.2.8.6.4 Function Readiness Status (FRS) Message

The Function Readiness Status (FRS) protocol (see Section 6.23.2) uses the PCI-SIG-Defined VDM mechanism (see Section 2.2.8.6.1). The FRS message is a PCI-SIG-Defined VDM (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:

Table 2-28 and Figure 2-29 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: $D3_{hot}$ to D0 Transition Completed** - A $D3_{hot}$ to D0 transition has completed, and the Function indicated by the Message Requester ID is now Configuration-Ready and has returned to the $D0_{uninitialized}$ or $D0_{active}$ state depending on the setting of the No_Soft_Reset bit (see Section 7.5.2.2)
  - **0011b: FLR Completed** - An FLR has completed, and the Function indicated by the Message Requester ID is now Configuration-Ready
  - **1000b: VF Enabled** – The Message Requester ID indicates a Physical Function (PF) - All Virtual 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.
  - 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 (see Section 6.2).

Table 2-28: 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 2-29 below:

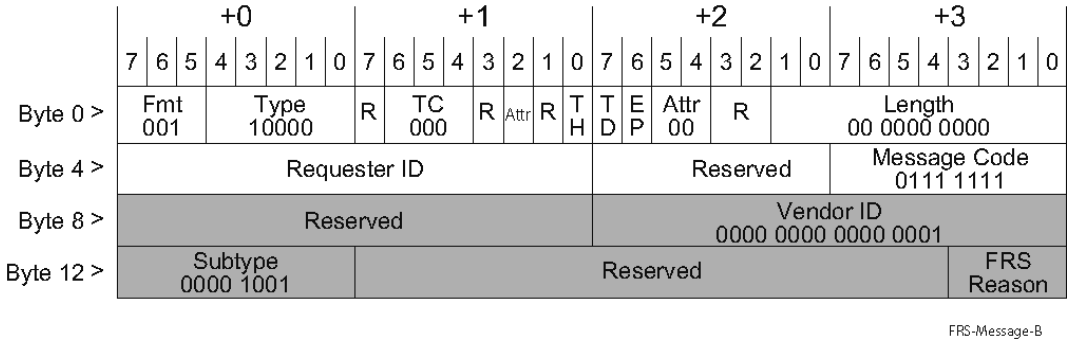


Figure 2-29: FRS Message

2.2.8.6.5 Hierarchy ID Message

Hierarchy ID uses the PCI-SIG-Defined VDM mechanism (see Section 2.2.8.6.1). The Hierarchy ID Message is a PCI-SIG-Defined VDM (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 2-29 and Figure 2-30 illustrate and define the Hierarchy ID Message.
- The TLP Type must be MsgD.
- 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.

1. If checked, this is a reported error associated with the Receiving Port (see Section 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 2-29: 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 2-30 below:

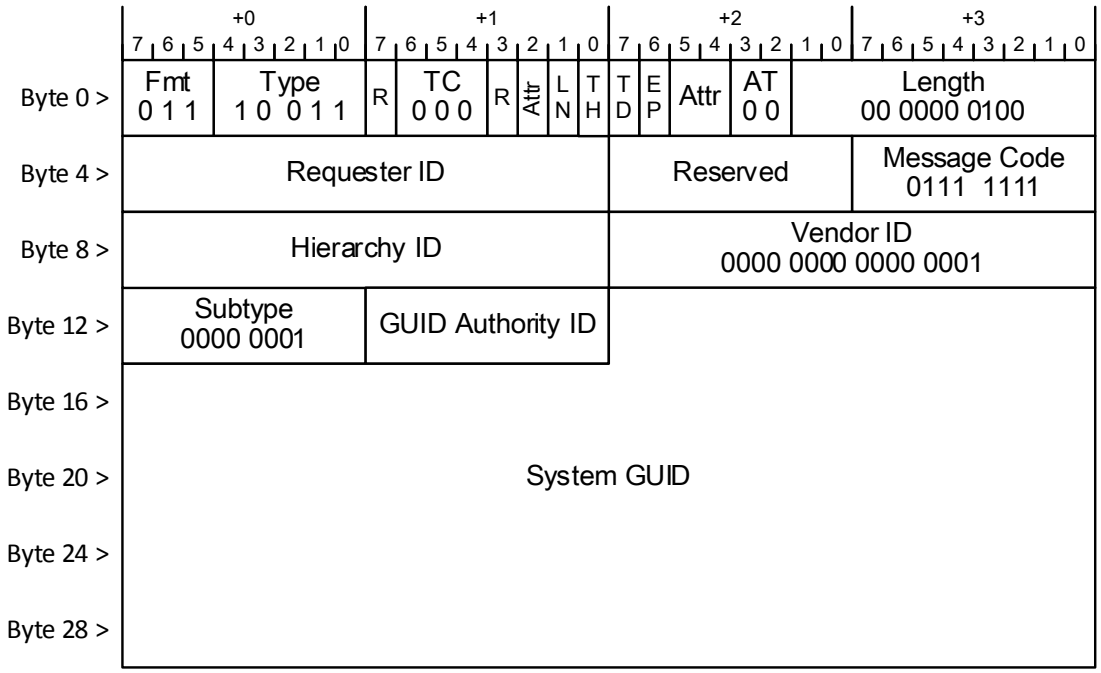


Figure 2-30: Hierarchy ID Message

2.2.8.7 *Ignored Messages*

The messages listed in Table 2-30 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 2-30 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 2-30: 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 *Latency Tolerance Reporting (LTR) Message*

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

- Table 2-31 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 (see Section 6.2).

Table 2-31: LTR Message

Name	Code[7:0] (b)	Routing r[2:0] (b)	Support[20]				Description/Comments
			RC	Ep	Sw	Br	
LTR	0001 0000	100	r	t	tr		Latency Tolerance Reporting

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

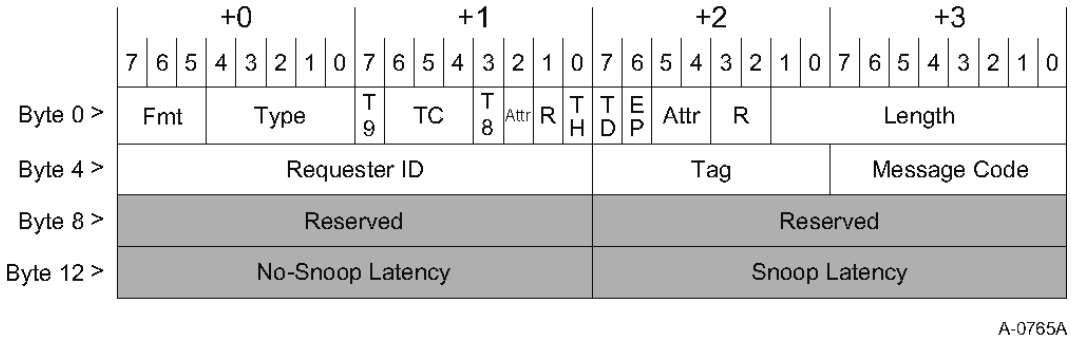


Figure 2-31: LTR Message

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 2-32 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 (see Section 6.2).

Table 2-32: OBFF Message

Name	Code[7:0] (b)	Routing r[2:0] (b)	Support[1]				Description/Comments
			RC	Ep	Sw	Br	
OBFF	0001 0010	100	t	r	tr		Optimized Buffer Flush/Fill

Note 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.

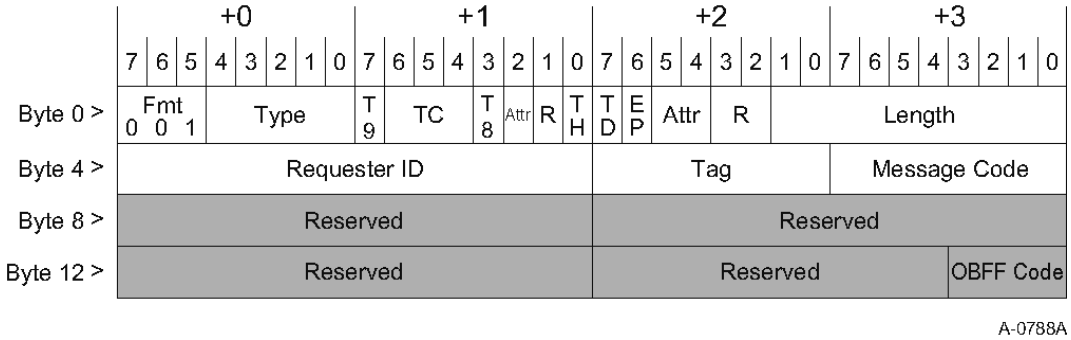


Figure 2-32: OBFF Message

2.2.8.10 *Precision Time Measurement (PTM) Messages*

Table 2-33 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 2-33 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 Delay field.
  - Figure 2-34 illustrates the format of the PTM ResponseD Message.
  - Refer to section 6.22.3.2 for 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 (see Section 6.2).

Table 2-33: 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
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

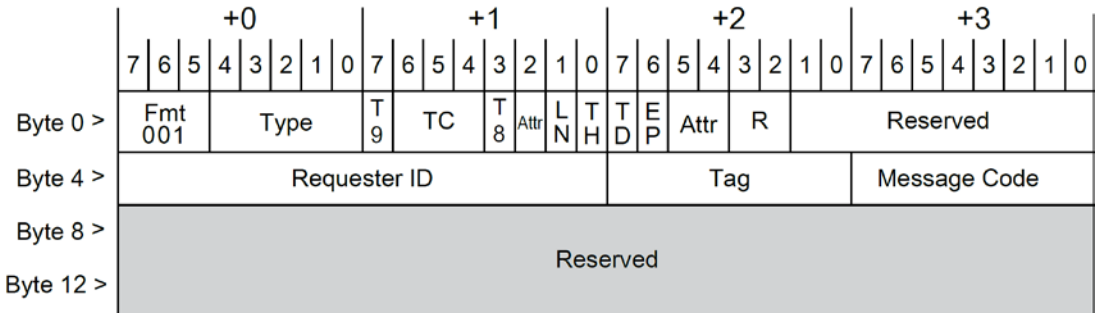


Figure 2-33: PTM Request/Response Message

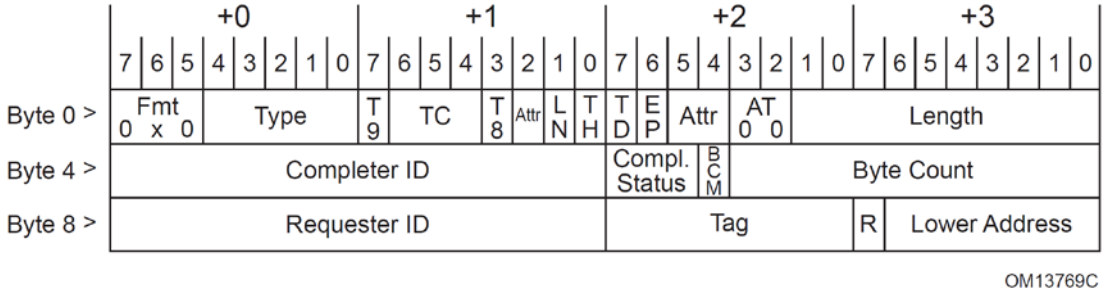


Figure 2-34: PTM ResponseD Message (4 DW header and 1 DW payload)

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 DW 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.
- In addition to the header fields included in all TLPs and the ID routing fields, Completions contain the following additional fields (see Figure 2-35):
  - Completer ID[15:0] – Identifies the Completer – described in detail below
  - Completion Status[2:0] – Indicates the status for a Completion (see Table 2-34)
    - Rules for determining the value in the Completion Status[2:0] field are in Section 2.3.1.
  - 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 2.3.1.1.
    - 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).
  - 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, second bullet, for details.

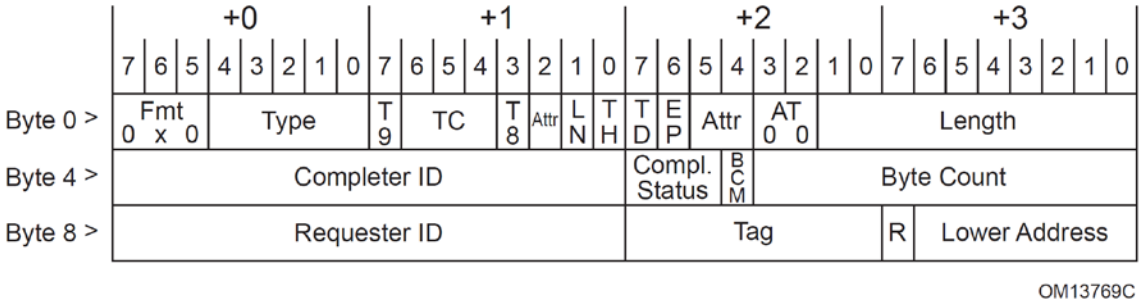


Figure 2-35: Completion Header Format

Table 2-34: Completion Status Field Values

Completion 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)
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 2-36 and Figure 2-37)

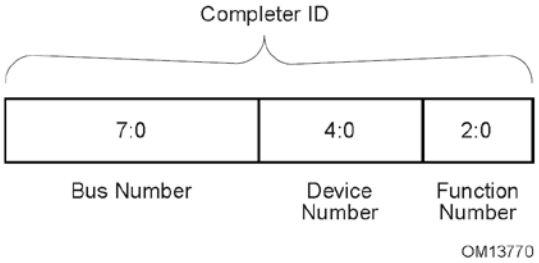


Figure 2-36: (Non-ARI) Completer ID

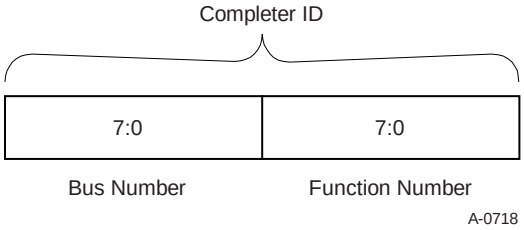


Figure 2-37: ARI Completer ID

- Functions must capture the Bus and Device Numbers[21] supplied with all Type 0 Configuration Write Requests completed by the Function, and supply these numbers in the Bus and Device Number fields of the Completer ID[22] 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 the UR status may be generated by a Multi-Function Device without associating the Completion with a specific Function within the device – in this case, the Function Number field[23] is Reserved.
  - Example: A Multi-Function Device receives a Read Request which does not target any resource associated with any of the Functions of the device – the device generates a Completion with UR status and sets a value of all 0's in the Function Number field of the Completer ID.
- Completion headers must supply the same values for the Requester ID, Tag, 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).
- 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.

---

[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.

[22] An ARI Completer ID does not contain a Device Number field. See Section 2.2.4.2.

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

- 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.
- A TLP 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 (see Section 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 (see Section 6.2).
- The size of each TLP Prefix is 1 DW. A TLP Prefix may be repeated to provide space for additional data.
- 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).
- 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).

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 2-35

Table 2-35: Local TLP Prefix Types

Local TLP Prefix Type	L[3:0] (b)	Description
MR-IOV	0000	**MR-IOV TLP Prefix** – Refer to the *Multi-Root I/O Virtualization and Sharing* specification for details.
VendPrefixL0	1110	**Vendor Defined Local TLP Prefix** – Refer to Section 2.2.10.1.1 for further details.
VendPrefixL1	1111	**Vendor Defined Local TLP Prefix** – Refer to Section 2.2.10.1.1 for 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 (see Section 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 2-35, 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 (see Section 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 *End-End TLP Prefix Processing*

The following rules apply to End-End TLP Prefixes

- 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 2-36

Table 2-36: End-End TLP Prefix Types

End-End TLP Prefix Type	E[3:0] (b)	Description
TPH	0000	**TPH** – Refer to Section 2.2.10.2 and 6.17 for further details.
PASID	0001	**PASID** – Refer to Section 6.20 for further details.
VendPrefixE0	1110	**Vendor Defined End-End TLP Prefix** – Refer to Section 2.2.10.2.1 for further details.
VendPrefixE1	1111	**Vendor Defined End-End TLP Prefix** – Refer to Section 2.2.10.2.1 for 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.
- 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 (see Section 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. A TLP in violation of this rule is handled as a Malformed TLP. This is a reported error associated with the Receiving Port (see Section 6.2).
- 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 (see Section 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 (see Section 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 (see Section 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 2-36, 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 it sends.

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.

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 (see Section 1.3.1).

2.3 Handling of Received TLPs

This section describes how all Received TLPs are handled when they are delivered to the Receive Transaction Layer from the Receive Data Link Layer, after the Data Link Layer has validated the integrity of the received TLP. The rules are diagramed in the flowchart shown in Figure 2-38.

- 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).
- If the Extended Fmt Field Supported bit is Set, Received TLPs which use encodings of Fmt and Type that are Reserved are Malformed TLPs (see Table 2-12 and Table 2-3).
  - This is a reported error associated with the Receiving Port (see Section 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 which use undefined Type field values are Malformed TLPs.

  This is a reported error associated with the Receiving Port (see Section 6.2).
- All Received Malformed TLPs must be discarded.

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

- 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)
- 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)

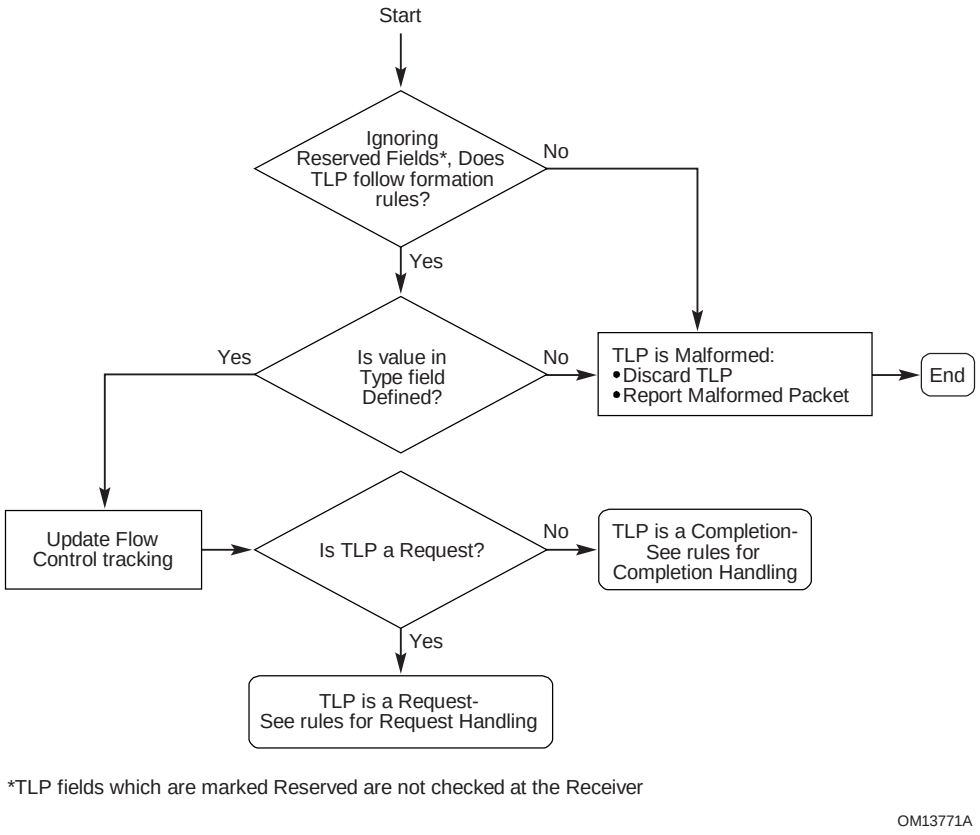


Figure 2-38: Flowchart for Handling of Received TLPs

Switches must process both TLPs which address resources within the Switch as well as TLPs which address resources residing outside the Switch. Switches handle all TLPs which address internal resources of the Switch according to the rules above. TLPs which pass through the Switch, or which address the Switch as well as passing through it, are handled according to the following rules (see Figure 2-39):

- If the value in the Type field indicates the TLP is not a Msg or MsgD Request, the TLP must be routed according to the routing mechanism used (see Sections 2.2.4.1 and 2.2.4.2).
- Switches route Completions using the information in the Requester ID field of the Completion.

- If the value in the Type field indicates the TLP is a Msg or MsgD Request, route the Request according to the routing mechanism indicated in the r[2:0] sub-field of the Type field
  - If the value in r[2:0] indicates the Msg/MsgD is routed to the Root Complex (000b), the Switch must route the Msg/MsgD to the Upstream Port of the Switch
    - It is an error to receive a Msg/MsgD Request specifying 000b routing at the Upstream Port of a Switch. Switches may check for violations of this rule – TLPs in violation are Malformed TLPs. If checked, this is a reported error associated with the Receiving Port (see Section 6.2).
  - If the value in r[2:0] indicates the Msg/MsgD is routed by address (001b), the Switch must route the Msg/MsgD in the same way it would route a Memory Request by address
  - If the value in r[2:0] indicates the Msg/MsgD is routed by ID (010b), the Switch must route the Msg/MsgD in the same way it would route a Completion by ID
  - If the value in r[2:0] indicates the Msg/MsgD is a broadcast from the Root Complex (011b), the Switch must route the Msg/MsgD to all Downstream Ports of the Switch
    - It is an error to receive a Msg/MsgD Request specifying 011b routing at the Downstream Port of a Switch. Switches may check for violations of this rule – TLPs in violation are Malformed TLPs. If checked, this is a reported error associated with the Receiving Port (see Section 6.2).
  - If the value in r[2:0] indicates the Msg/MsgD terminates at the Receiver (100b or a Reserved value), or if the Message Code field value is defined and corresponds to a Message which must be comprehended by the Switch, the Switch must process the Message according to the Message processing rules
  - If the value in r[2:0] indicates Gathered and routed to Root Complex (101b), see Section 5.3.3.2.1 for Message handling rules
  - It is an error to receive any Msg/MsgD Request other than a PME_TO_Ack that specifies 101b routing. It is an error to receive a PME_TO_Ack 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).

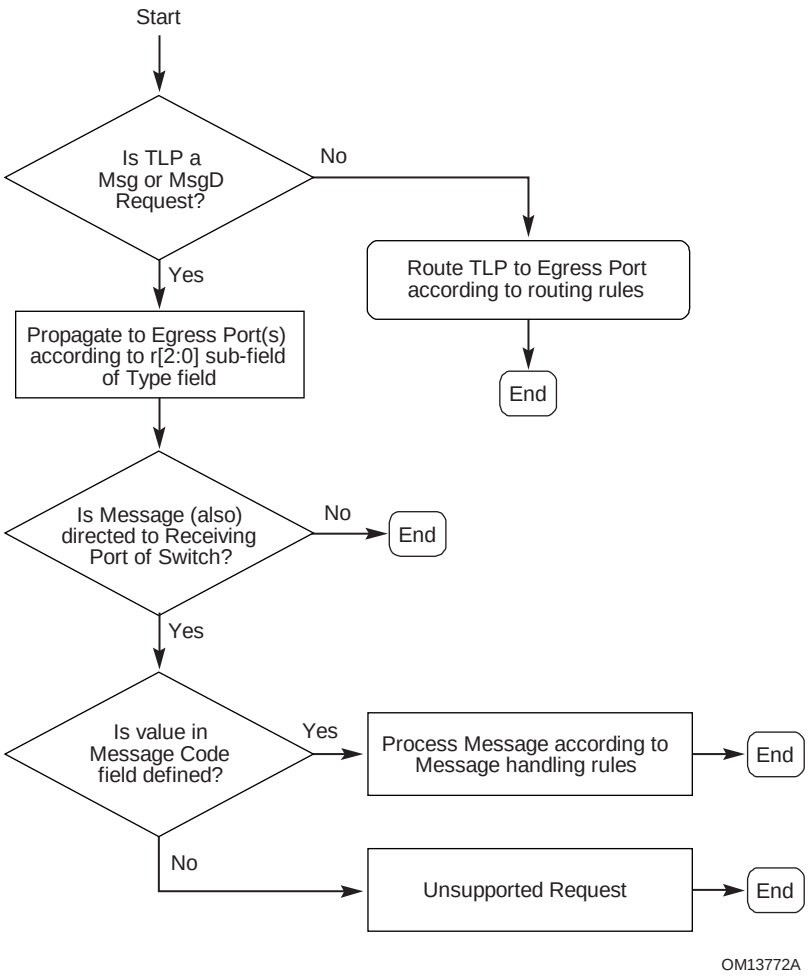


Figure 2-39: Flowchart for Switch Handling of TLPs

2.3.1 Request Handling Rules

This section describes how Received Requests are handled, following the initial processing done with all TLPs. The rules are diagramed in the flowchart shown in Figure 2-40.

- 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 to Section 6.2
  - If the Request requires Completion, a Completion Status of UR is returned (see Section 2.2.8.10)

IMPLEMENTATION NOTE

When Requests are Terminated Using Unsupported Request

In Conventional PCI, a device "claims" a request on the bus by 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 / MsgD indication 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 to Section 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)

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

- 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)

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

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 $D3_{hot}$ to $D0_{uninitialized}$ device 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.
  - 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 which requires a Completion to be returned, generate a Completion according to the rules for Completion formation (see Section 2.2.9)
  - The Completion Status is determined by the result of handling the Request
  - 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 (see Section 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.

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.

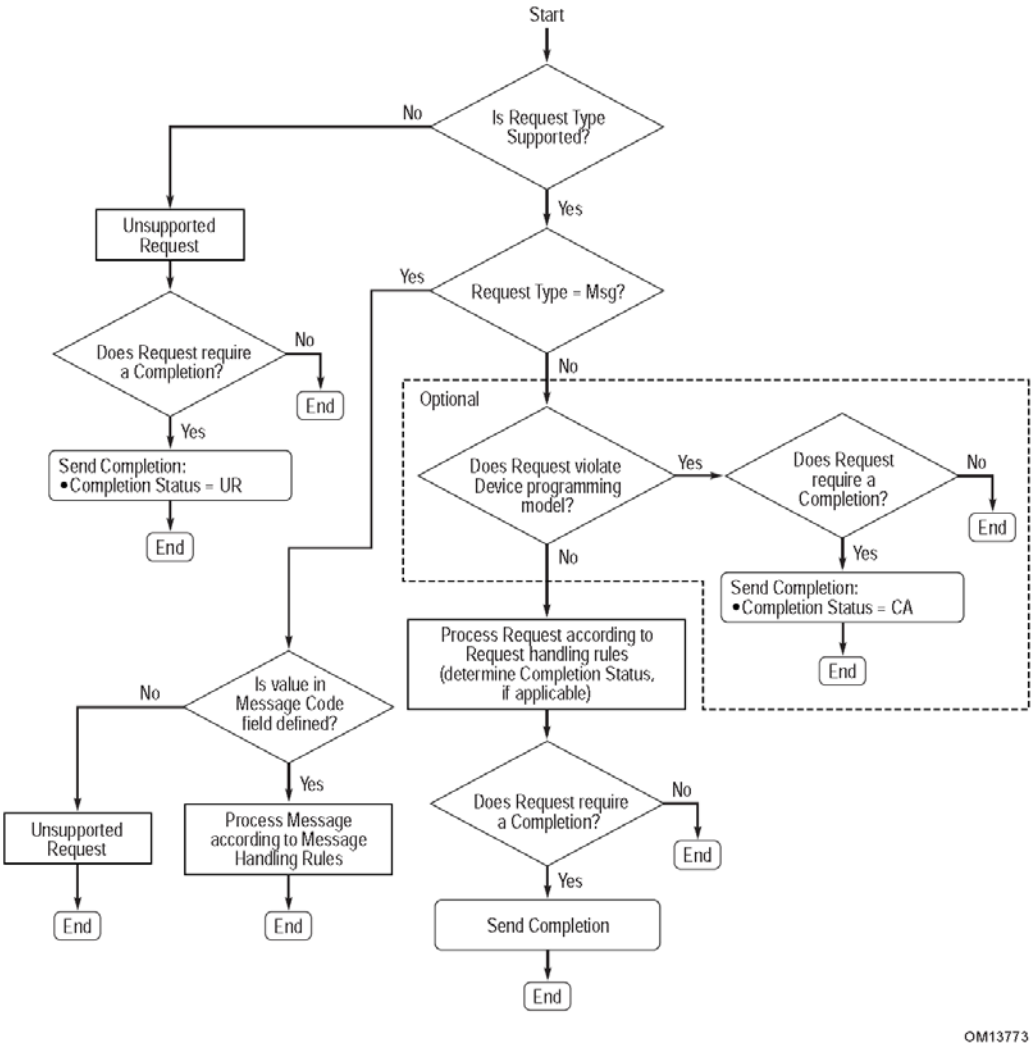


Figure 2-40: Flowchart for Handling of Received Request

IMPLEMENTATION NOTE

Configuration Request Retry Status

Some devices require a lengthy self-initialization sequence to complete before they are able to service Configuration Requests (common with intelligent I/O solutions on PCI). PCI/PCI-X architecture has specified a 2^{25} (PCI) or 2^{26} (PCI-X) clock "recovery time" T_{rhfa} following reset to provide the required self-initialization time for such devices. PCI Express "softens" the need for this time based recovery period by implementing a Configuration Request Retry Status (CRS) Completion Status. 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). Readiness Notifications (see Section 6.23) and Immediate Readiness (see Section 7.5.1.1.4 and 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.

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.

When used in systems including PCI Express to PCI/PCI-X Bridges, system software and/or the Root Complex must comprehend the limit T_{rhfa} for PCI/PCI-X agents as described in Sections 2.8 and 6.6. Similarly, systems using PCI Express components which require additional self initialization time beyond the minimum guaranteed must provide some mechanism for re-issuing Configuration Requests terminated with CRS status. In systems running legacy PCI/PCI-X based software, the Root Complex must re-issue the Configuration Request using a hardware mechanism to ensure proper enumeration of the system.

Refer to Section 6.6 for more information on reset.

2.3.1.1 *Data Return for Read Requests*

- Individual Completions for Memory Read Requests may provide less than the full amount of data Requested so long as all Completions for a given Request when combined return exactly the amount of data Requested in the Read Request.
  - Completions for different Requests cannot be combined.
  - I/O and Configuration Reads must be completed with exactly one Completion.
  - The Completion Status for a Completion corresponds only to the status associated with the data returned with that Completion
    - 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.

  Note: This is simply a specific case of the rules which apply to all TLPs with data payloads
- Memory Read Requests may be completed with one, or in some cases, multiple Completions
- 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 (see Section 7.5.3.7)

  Note: Bridges and Endpoints may implement a corresponding command bit which may be set by system software to indicate the RCB value for the Root Complex, allowing the Bridge 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 which do not cross the naturally aligned address boundaries at integer multiples of RCB bytes must include all data specified in the Request
- Requests which do cross the address boundaries at integer multiples of RCB bytes 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 (see Section 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 2-37
  - 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 2-41 presents an example of the above. The example assumes a single Completion TLP is returned:

[25] Only PCI-X completers Set the BCM bit. PCI Express completers are not permitted to set the BCM bit.

Completion Data

Request Address (DW)	Byte 0	Byte 1	Byte 2	Byte 3	Request Byte Enables
START	undefined content	undefined content	undefined content		First BE: 1000
START + 1					
START + 2					
START + 3		undefined content	undefined content	undefined content	Last BE: 0001

Length = 4d;
Byte Count = 10d;

Figure 2-41: Example Completion Data when some Byte Enables are 0b

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 Addendum to the PCI Local Bus Specification, Revision 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 alw ays 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 2-37: Calculating Byte Count from Length and Byte Enables

1st DW BE[3:0] (b)	Last DW BE[3:0] (b)	Total Byte Count
1xx1	0000[26]	4
01x1	0000	3
1x10	0000	3
0011	0000	2
0110	0000	2
1100	0000	2
0001	0000	1
0010	0000	1
0100	0000	1
1000	0000	1
0000	0000	1
xxx1	1xxx	Length[27] * 4
xxx1	01xx	(Length * 4) - 1
xxx1	001x	(Length * 4) - 2
xxx1	0001	(Length * 4) - 3
xx10	1xxx	(Length * 4) - 1
xx10	01xx	(Length * 4) - 2
xx10	001x	(Length * 4) - 3
xx10	0001	(Length * 4) - 4
x100	1xxx	(Length * 4) - 2
x100	01xx	(Length * 4) - 3
x100	001x	(Length * 4) - 4
x100	0001	(Length * 4) - 5
1000	1xxx	(Length * 4) - 3
1000	01xx	(Length * 4) - 4
1000	001x	(Length * 4) - 5
1000	0001	(Length * 4) - 6

[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.

- 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 2-38.
  - 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 2-38: Calculating Lower Address from 1st DW BE

1st 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.
    - The Completer must not transmit additional Completions for this Request.
      - Example: Completer split the Request into four parts for servicing; the second Completion had a Completer Abort Completion Status; the Completer terminated servicing for the Request, and did not Transmit the remaining two Completions.
  - The Byte Count field must indicate the remaining number of bytes that would be required to complete the Request (as if the Completion Status were Successful Completion)
  - The Lower Address field must indicate the lower bits of the byte address for the first enabled byte of data that would have been returned with the Completion if the Completion Status were Successful Completion

IMPLEMENTATION NOTE

Restricted Programming Model

When a device's programming model restricts (vs. what is otherwise permitted in PCI Express) the size and/or alignment of Read Requests directed to the device, that device is permitted to use a

Completer Abort Completion Status for Read Requests which violate the programming model. An implication of this is that such devices, generally devices where all communication will be between the device's driver software and the device itself, need not necessarily implement the buffering required to generate Completions of length RCB. However, in all cases, the boundaries specified by RCB must be respected for all reads which the device will complete with Successful Completion status.

Examples:

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

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

3. Memory Read Request with Address of 1 0020h and Length of 100h bytes (256 decimal) could be completed by a Root Complex with an RCB value of 64 bytes in one of the following combinations of Completions (bytes):
256 –or–
32, 224 –or– 32, 64, 160 –or– 32, 64, 64, 96 –or– 32, 64, 64, 64, 32 –or–
32, 64, 128, 32 –or– 32, 128, 96 –or– 32, 128, 64, 32 –or–
96, 160 –or– 96, 128, 32 –or– 96, 64, 96 –or– 96, 64, 64, 32 –or–
160, 96 –or– 160, 64, 32 –or– 224, 32

4. Memory Read Request with Address of 1 0020h and Length of 100h bytes (256 decimal) could be completed by an Endpoint in one of the following combinations of Completions (bytes):
256 –or– 96, 160 –or– 96, 128, 32 –or– 224, 32

2.3.2 Completion Handling Rules

- When a device receives a Completion which 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 and Section 2.2.9.

- However, if the Completion is otherwise properly formed, it is permitted[28] for the Receiver to handle the Completion as an Unexpected Completion.

- When an Ingress Port of a Switch receives a Completion which 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 (see Section 6.2).

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

- Completions with a Completion Status other than Successful Completion or Configuration Request Retry Status (in response to Configuration Request only) must cause the Requester to:
  - Free Completion buffer space and other resources associated with the Request.
  - 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 (see Section 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.

- 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 (see Section 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 the 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 (see Section 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 RCRB that 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 (see Sections 7.5.3.13 and 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 (see Section 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 (see Section 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 a Completion with the Completer Abort Completion Status. The Requester then must free the internal resources which had been allocated for that particular Read Request.

IMPLEMENTATION NOTE

Read Data Values with UR Completion Status

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 2-39 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 2-39, 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 2-39: 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 b) Y/N	Yes	Yes	a) Y/N b) Yes
Non-Posted Request	**Read Request** (Row B)	a) No b) Y/N	Y/N	Y/N	Y/N
	NPR with Data (Row C)	a) No b) Y/N	Y/N	Y/N	Y/N
Completion (Row D)		a) No b) Y/N	Yes	Yes	a) Y/N b) No

Explanation of the row and column headers in Table 2-39:

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 2-39:

A2a A Posted Request must not pass another Posted Request unless A2b applies.

A2b A Posted Request with RO[29] 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 A Posted Request must be able to pass Non-Posted Requests to avoid deadlocks.

A5a A Posted Request is permitted to pass a Completion, but is not required to be able to pass Completions unless A5b applies.

A5b 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.

[29] In this section, “RO” is an abbreviation for the Relaxed Ordering Attribute field.

[30] Some usages are enabled by not implementing this passing (see the No RO-enabled PR-PR Passing bit in Section 7.6.3.15).

B2a A Read Request must not pass a Posted Request unless B2b applies.

B2b 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 PASID TLP Prefix and the two PASID values are different.

C2a An NPR with Data must not pass a Posted Request unless C2b applies.

C2b An NPR with Data and with RO Set[31] 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 PASID TLP Prefix and the two PASID values are different.

B3, B4, C3, C4 A Non-Posted Request is permitted to pass another Non-Posted Request.

B5, C5 A Non-Posted Request is permitted to pass a Completion.

D2a A Completion must not pass a Posted Request unless D2b applies.

D2b An I/O or Configuration Write Completion[32] 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 A Completion must be able to pass Non-Posted Requests to avoid deadlocks.

D5a Completions with different Transaction IDs are permitted to pass each other.

D5b 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 bit set to pass any previously posted Memory Write or Message Request moving in the same direction. Switches must forward the Relaxed Ordering attribute unmodified. The Root Complex is also permitted to allow data bytes within the Request to be written to system memory in any order. (The bytes must be written to the correct system memory locations. Only the order in which they are written is unspecified).
- For Root Complex and Switch, Memory Write combining (as defined in the *PCI Local Bus Specification*) is prohibited.
- Note: This is required so that devices can be permitted to optimize their receive buffer and control logic for Memory Write sizes matching their natural expected sizes, rather than being required to support the maximum possible Memory Write payload size.

---

[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.

- Combining of Memory Read Requests, and/or Completions for different Requests is prohibited.
- The No Snoop 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.[33]
- 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.[33]
- 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.[33]
- 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.[33]

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 (see Appendix D for a discussion of relevant issues).

[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 D for a discussion of relevant issues).

IMPLEMENTATION NOTE

Large Memory Reads vs. Multiple Smaller Memory Reads

Note that the rule associated with entry D5b in Table 2-39 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) 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 B via one read transaction before initiating a subsequent read to location A to return its updated value.

As an example of update granularity, if a host CPU writes a 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.

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 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 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 Sections 6.3, 7.9.1, and 7.9.2.

Support for TCs and VCs beyond default TC0/VC0 pair is optional. The association of TC0 with VC0 is fixed, i.e., "hardwired", and must be supported by all components. Therefore the baseline TC/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.)
- A Completer must accept requests with TC label other than TC0, and must preserve the TC label, i.e., any completion that it generates must have the same TC label as the label of the request.
- A Switch must map all TCs to VC0 and must forward all transactions regardless of the TC label.

A 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 2-42 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.

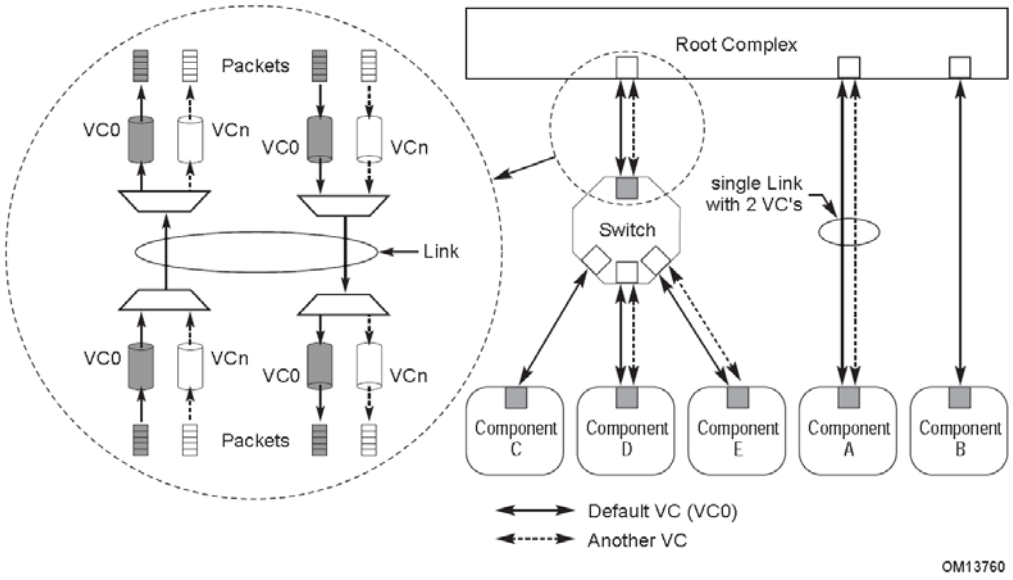


Figure 2-42: Virtual Channel Concept – An Illustration

Internal to the Switch, every Virtual Channel requires dedicated physical resources (queues/buffers and control logic) that support independent traffic flows inside the Switch. Figure 2-43 shows conceptually the VC resources within the Switch (shown in Figure 2-42) that are required to support traffic flow in the Upstream direction.

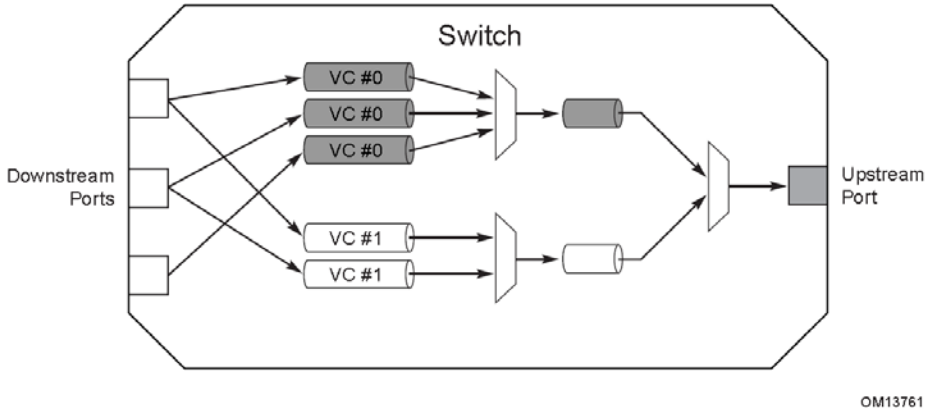


Figure 2-43: Virtual Channel Concept – Switch Internals (Upstream Flow)

A Multi-Function Device 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, i.e., an implementation may provide greater buffer depth for selected VCs as a function of implementation usage models and other Link attributes, e.g., Link width and signaling.

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

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.

All Ports that support more than VC0 must provide at least one VC Capability structure according to the definition in Section 7.9.1. A Multi-Function Device 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 a Multi-Function Device 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.
- One TC 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 2-40 provides an example of TC to VC mapping.

Table 2-40: TC to VC Mapping Example

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

VC0-VC7	TC[0:7]/VC[0:7]

Notes on conventions:

- TCn/VCk = TCn mapped to VCk
- TC(n-m)/VCk = all TCs in the range n-m mapped to VCk (i.e., to the same VC)
- TC[n:m]/VC[n:m] = TCn/VCn, TCn_{+1}/ $VCn_{+1,}$..., TCm/VCm

Figure 2-44 provides a graphical illustration of TC to VC mapping in several different Link configurations. For additional considerations on TC/VC, refer to Section 6.3.

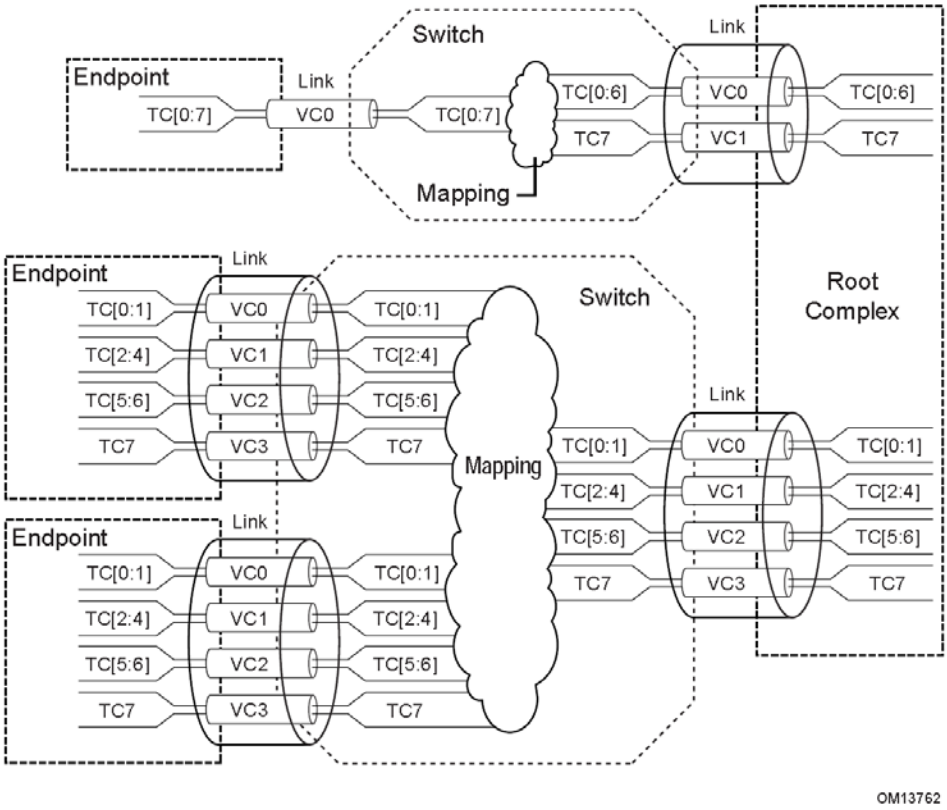


Figure 2-44: An Example of TC/VC Configurations

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.
- A Multi-Function Device's peer-to-peer capability between different Functions applies to all Virtual Channels supported by the Multi-Function Device.
- 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 target RCRB are treated as Malformed TLPs.
- For Multi-Function Devices 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 to Section 6.3.

2.6 Ordering and Receive Buffer Flow Control

Flow Control (FC) is used to prevent overflow of Receiver buffers and to enable compliance with the ordering rules defined in Section 2.4. Note that the Flow Control mechanism is used by the Requester to track the queue/buffer space available in the agent across the Link as shown in Figure 2-45. 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.

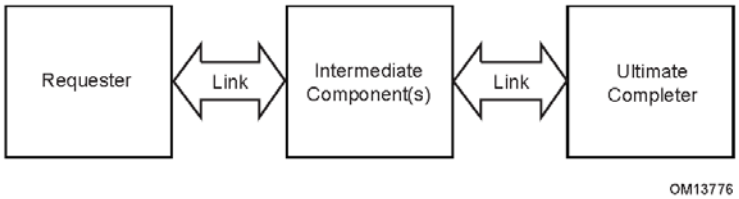


Figure 2-45: Relationship Between Requester and Ultimate Completer

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 a Multi-Function Device 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, 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 types of TLPs (note relationship to ordering rules – see Section 2.4):
  - Posted Requests (P) – Messages and Memory Writes
  - Non-Posted Requests (NP) – All Reads, I/O Writes, Configuration Writes, and AtomicOps
  - Completions (Cpl) – Associated with corresponding NP Requests

- In addition, Flow Control distinguishes the following types of TLP information within each of the three types:
  - Headers (H)
  - Data (D)
- Thus, there are six types of information tracked by Flow Control for each Virtual Channel, as shown in Table 2-41.

Table 2-41: Flow Control Credit Types

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

- TLPs consume Flow Control credits as shown in Table 2-42.

Table 2-42: TLP Flow Control Credit Consumption

TLP	Credit Consumed[34]
Memory, I/O, Configuration Read Request	1 NPH unit
Memory Write Request	1 PH + n PD units[35]
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 (see Sections 3.2 and 3.4)

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

[35] For all cases where “n” appears, n = Roundup(Length/FC unit size).

- 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 Sections 7.9.1 and 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
- During FC initialization for any Virtual Channel, including the default VC initialized as a part of Link initialization, Receivers must initially advertise VC credit values equal to or greater than those shown in Table 2-43.
  - 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 2-43: Minimum Initial Flow Control Advertisements[36]

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 a Multi-Function Device, 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 a Multi-Function Device, 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 a Multi-Function Device, 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[37].	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[37].	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[37].

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

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

Credit Type	Minimum Advertisement		
	No Scaling or Scale Factor 1	**Scale Factor 4**	**Scale Factor 16**
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 (not supporting peer-to-peer traffic between all Root Ports) and Endpoint: infinite FC units - initial credit value of all 0s.	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 (not supporting peer-to-peer traffic between all Root Ports) and Endpoint: infinite FC units - initial credit value of all 0s.	Root Complex (supporting peer-to-peer traffic between all Root Ports) and Switch: Ceiling(Largest Max_Payload_Size / (FC Unit Size * 16)) + 1. , Root Complex (not supporting peer-to-peer traffic between all Root Ports) and Endpoint: infinite FC units - initial credit value of all 0s.

- A Root Complex that 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 (see Section 6.2)
- If an Infinite Credit advertisement (value of 00h or 000h) has been made during initialization, no Flow Control updates are required following initialization.
  - If UpdateFC DLLPs are sent, the credit value fields must be 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 (see Section 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 (see Section 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 (see Section 6.2).

- A received TLP using a VC that is not enabled is a Malformed TLP
  - VC0 is always enabled
  - 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 (see Section 3.4)
  - This is a reported error associated with the Receiving Port (see Section 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 2-44 (see Section 3.4.2).

Table 2-44: [Field Size] Values

Scaled Flow Control Supported	HdrScale or DataScale	[Field Size] for PH, NPH, CplH	[Field Size] 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 *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 $2^{[\text{Field Size}]}$ (where [Field Size] is defined in Table 2-44).
    - Set to all 0's at interface initialization
    - Updated for each TLP the Transaction Layer allows to pass the Flow Control gate for Transmission
      - Updated as shown:

        CREDITS_CONSUMED :=

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

        Where Increment is the size in FC credits of the corresponding part of the TLP passed through the gate, and [Field Size] is defined in Table 2-44)
  - 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 $2^{[\text{Field Size}]}$ (where [Field Size] is defined in Table 2-44).
    - Undefined at interface initialization
    - Set to the value indicated during Flow Control initialization
    - For each FC update received,
      - if CREDIT_LIMIT is not equal to the update value, set CREDIT_LIMIT to 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 bad ECRC, 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, which require that certain types of TLPs must bypass

other specific types of TLPs when the latter are blocked. Note that TLPs using different Virtual Channels have no ordering relationship, and must not block each other.

- The Transmitter gating function test is performed as follows:
  - For each required type of credit, the number of credits required is calculated as:

    CUMULATIVE_CREDITS_REQUIRED =

    (CREDITS_CONSUMED +
    credit units required for pending TLP) mod $2^{[\text{Field Size}]}$

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

    (CREDIT_LIMIT –
    CUMULATIVE_CREDITS_REQUIRED) mod $2^{[\text{Field Size}]}$

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

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

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

- When accounting for credit use and return, information from different TLPs is never mixed within one credit.
- When some TLP is blocked from Transmission by a lack of FC Credit, Transmitters must follow the ordering rules specified in Section 2.4 when determining what types of TLPs must be permitted to bypass the stalled TLP.
- The return of FC credits for a Transaction must not be interpreted to mean that the Transaction has completed or achieved system visibility.
  - Flow Control credit return is used for receive buffer management only, and agents must not make any judgment about the Completion status or system visibility of a Transaction based on the return or lack of return of Flow Control information.
- When a Transmitter sends a nullified TLP, the Transmitter does not modify CREDITS_CONSUMED for that TLP (see Section 3.6.2.1).

2.6.1.2 *FC Information Tracked by Receiver*

- For each type of information tracked, the following quantities are tracked for Flow Control TLP Receiver accounting:
  - CREDITS_ALLOCATED
    - Count of the total number of credits granted to the Transmitter since initialization, modulo $2^{[\text{Field Size}]}$ (where [Field Size] is defined in Table 2-44)
    - Initially set according to the buffer size and allocation policies of the Receiver
    - This value is included in the InitFC and UpdateFC DLLPs (see Section 3.5)

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

  Updated as shown:
  CREDITS_ALLOCATED :=

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

  Where Increment corresponds to the credits made available, and [Field Size] is defined in Table 2-44

- 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 $2^{[\text{Field Size}]}$ (where [Field Size] is defined in Table 2-44
  - Set to all 0's at interface initialization
  - Updated as shown:

    CREDITS_RECEIVED :=

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

    (Where Increment is the size in FC units of the corresponding part of the received TLP, and [Field Size] is defined in Table 2-44

    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 be updated.

- If a Receiver implements the CREDITS_RECEIVED counter, then when a nullified TLP is received, the Receiver does not modify CREDITS_RECEIVED for that TLP (see Section 3.6.2.1)
- A Receiver may optionally check for Receiver Overflow errors (TLPs exceeding CREDITS_ALLOCATED), by checking the following equation, using unsigned arithmetic:

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

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

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

  - discard the TLP(s) without modifying the CREDITS_RECEIVED
  - de-allocate any resources which it had allocated for the TLP(s)

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

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

- For non-infinite NPH, NPD, PH, and CplH types, an UpdateFC FCP must be scheduled for Transmission each time the following 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 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. Where the scaled threshold 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 Multi-Function Device 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 Multi-Function Device 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

- 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.

IMPLEMENTATION NOTE

Flow Control Update Latency

For components subject to receiving streams of TLPs, it is desirable to implement receive buffers larger than the minimum size required to prevent Transmitter throttling due to lack of available credits. Likewise, 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 2-45, Table 2-46, Table 2-47, and Table 2-48. 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 2-45: 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 2-46: 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

Table 2-47: Maximum UpdateFC Transmission Latency Guidelines for 8.0 GT/s (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

Table 2-48: Maximum UpdateFC Transmission Latency Guidelines for 16.0 GT/s (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. A TLP is a unit of data and transaction control that is created by a data-source at the "edge" of the PCI Express domain (such as an Endpoint or Root Complex), potentially routed through intermediate components (i.e., Switches) and consumed by the ultimate PCI Express recipient. As a TLP passes through a Switch, the Switch may need to change some control fields without modifying other fields that should not change as the packet traverses the path. Therefore, the LCRC is regenerated by Switches. Data corruption may occur internally to the Switch, and the regeneration of a good LCRC for corrupted data masks the existence of errors. To ensure end-to-end data integrity detection in systems that require high data reliability, a Transaction Layer end-to-end 32-bit CRC (ECRC) can be placed in the TLP Digest field at the end of a TLP. The ECRC covers all fields that do not change as the TLP traverses the path (invariant fields). The ECRC is generated by the Transaction Layer in the source component, and checked (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.[38]

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 Port[38]
- 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.[38]

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):

- The ECRC value is calculated using the following algorithm (see Figure 2-46)
- 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 in Variant fields must be Set for ECRC calculations.
    - Bit 0 of the Type field in a TLP header is Variant[39]. This bit in an End-End TLP Prefix is invariant.
    - The EP bit is Variant
    - all other fields are Invariant

---

[38] An exception is a Multicast TLP that an Egress Port is modifying due to the MC_Overlay mechanism. See Section 6.14.5.

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

- 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.

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
12	11
13	10
14	9
15	8
16	23
17	22
18	21
19	20
20	19
21	18
22	17
23	16
24	31
25	30
26	29
27	28
28	27
29	26
30	25
31	24

- The 32-bit ECRC value is placed in the TLP Digest field at the end of the TLP (see Figure 2-3)
- For TLPs including a TLP Digest field used for an ECRC value, Receivers which support end-to-end data integrity checking, check the ECRC value in the TLP Digest field by:

- applying the same algorithm used for ECRC calculation (above) to the received TLP, not including the 32-bit TLP Digest field of the received TLP
- comparing the calculated result with the value in the TLP Digest field of the received TLP

- Receivers which support end-to-end data integrity checks report violations as an ECRC Error. This reported error is associated with the Receiving Port (see Section 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.

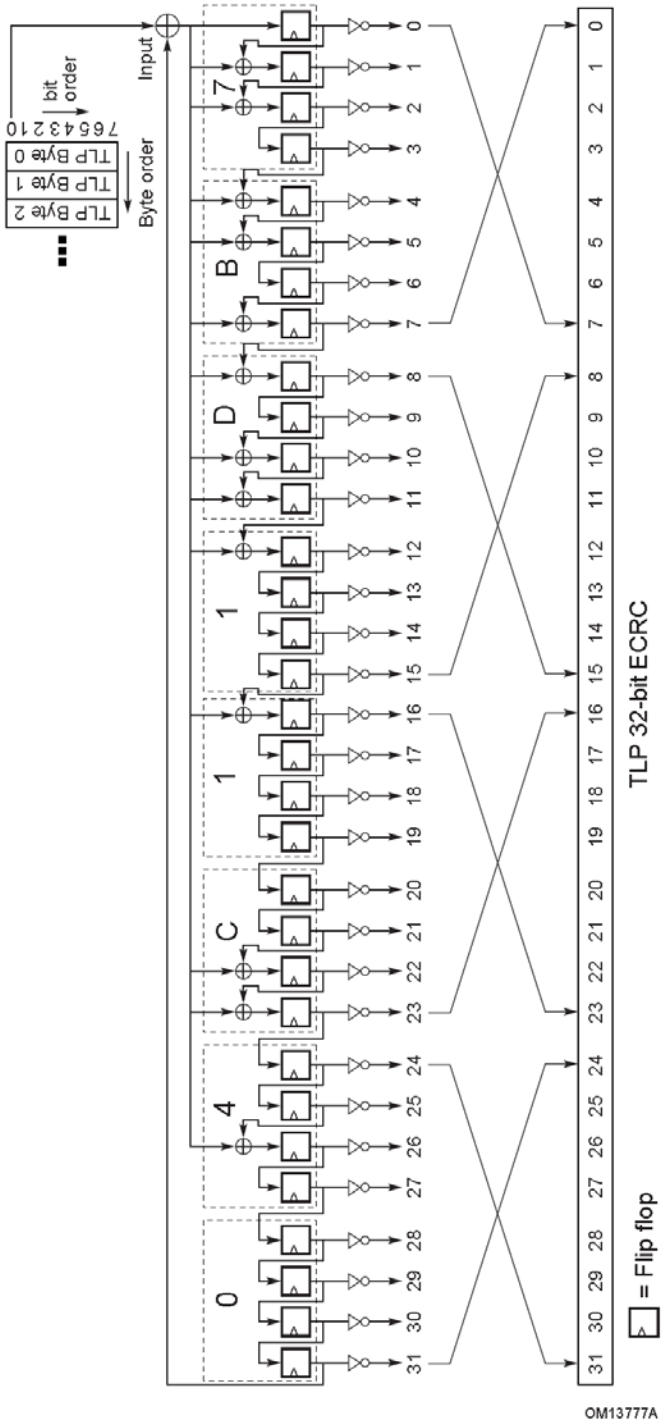


Figure 2-46: Calculation of 32-bit ECRC for TLP End to End Data Integrity Protection

IMPLEMENTATION NOTE

Protection of TD Bit Inside Switches

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

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

IMPLEMENTATION NOTE

Data Link Layer Does Not Have Internal TLP Visibility

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

2.7.2 Error Forwarding

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

- Example #1: A read from main memory encounters uncorrectable error
- Example #2: Parity error on a PCI write to main memory
- Example #3: Data integrity error on an internal data buffer or cache.

2.7.2.1 *Error Forwarding Usage Model*

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

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

2.7.2.2 *Rules For Use of Data Poisoning*

- Support for TLP poisoning in a Transmitter is optional.
- Data poisoning applies only to the data within a Write Request (Posted or Non-Posted), a Message with Data, an AtomicOp Request, a Read Completion, or an AtomicOp Completion.
  - Poisoning of a TLP is indicated by a Set EP bit.
  - Transmitters are permitted to Set the EP bit only for TLPs that include a data payload. The behavior of the Receiver is not specified if the EP bit is set for any TLP that does not include a data payload.
- If a Transmitter supports data poisoning, TLPs that are known to the Transmitter to include bad data must use the poisoning mechanism defined above.
- If a Downstream Port supports Poisoned TLP Egress Blocking, the Poisoned TLP Egress Blocking Enable bit is Set, and a poisoned TLP targets going out the Egress Port, the Port must handle the TLP as a Poisoned TLP Egress Blocked error unless there is a higher precedence error. See Sections 6.2.3.2.3, 6.2.5, and 7.9.15.2.
  Further:
  - The Port must not transmit the TLP.
  - If the Poisoned TLP Egress Blocked error is unmasked and Downstream Port Containment (DPC) is enabled, DPC must be triggered and the Port must behave as described in Section 2.9.3.
  - If DPC is not triggered and the TLP is a Non-Posted Request, the Port must return a Completion the same as if the TLP had triggered DPC. See Section 2.9.3.
- The following Requests with Poisoned data must not modify the value of the target location:
  - Configuration Write Request
  - Any of the following that target a control register or control structure in the Completer: I/O Write Request, Memory Write Request, or non-vendor-defined Message with data
  - AtomicOp Request

  Unless there is a higher precedence error, a Completer must handle these Requests as a Poisoned TLP Received error[40], and the Completer must also return a Completion with a Completion Status of Unsupported Request (UR) if the Request is Non-Posted (see Section 6.2.3.2.3, Section 6.2.3.2.4, and Section 6.2.5). Regardless of the severity of the reported

[40] Due to ambiguous language in earlier versions of this specification, a component is permitted to handle this error as an Unsupported Request, but this is strongly discouraged.

error, the reported error must be handled as an uncorrectable error, not an Advisory Non-Fatal Error.

A Switch must route the Request the same way it would route the same Request if it were not poisoned, unless the Request targets a location in the Switch itself, in which case the Switch is the Completer for the Request and must follow the above rules.

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

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

2.8 Completion Timeout Mechanism

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

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

The Completion Timeout mechanism may be disabled by configuration software. The Completion Timeout limit is set in the Completion Timeout Value field of the Device Control 2 register. A Completion Timeout is a reported error associated with the Requester Function (see Section 6.2).

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

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

IMPLEMENTATION NOTE

Completion Timeout Prefix/Header Log Capable

The prefix/header of the Request TLP associated with a Completion Timeout may optionally be recorded by Requesters that implement the AER Capability. Support for recording of the prefix/header is indicated by the value of the Completion Timeout Prefix/Header Log Capable bit in the Advanced Error Capabilities and Control register.

A Completion Timeout may be the result of improper configuration, system failure, or Async Removal (see Section 6.7.5). In order for host software to distinguish a Completion Timeout error after which continued normal operation is not possible (e.g., after one caused by improper configuration or a system failure) from one where continued normal operation is possible (e.g., after an Async Removal), it is strongly encouraged that Requesters log the Request TLP prefix/header associated with the Completion Timeout.

2.9 Link Status Dependencies

2.9.1 Transaction Layer Behavior in DL_Down Status

DL_Down status indicates that there is no connection with another component on the Link, or that the connection with the other component has been lost and is not recoverable by the Physical or Data Link Layers. This section specifies the Transaction Layer's behavior if DPC has not been triggered and the Data Link Layer reports DL_Down status to the Transaction Layer, indicating that the Link is non-operational. Section 2.9.3 specifies the behavior if DPC has been triggered.

- For a Port with DL_Down status, the Transaction Layer is not required to accept received TLPs from the Data Link Layer, provided that these TLPs have not been acknowledged by the Data Link Layer. Such TLPs do not modify receive Flow Control credits.

For a Downstream Port, DL_Down status is handled by:

- initializing back to their default state any buffers or internal states associated with outstanding requests transmitted Downstream
  - Note: Port configuration registers must not be affected, except as required to update status associated with the transition to DL_Down
- for Non-Posted Requests, forming completions for any Requests submitted by the device core for Transmission, returning Unsupported Request Completion Status, then discarding the Requests
  - This is a reported error associated with the Function for the (virtual) Bridge associated with the Port (see Section 6.2). For Root Ports, the reporting of this error is optional.
  - Non-Posted Requests already being processed by the Transaction Layer, for which it may not be practical to return Completions, are discarded.

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

- These cases are handled by the Requester using the Completion Timeout mechanism.

Note: The point at which a Non-Posted Request becomes "uncompletable" is implementation specific.

- ❑ The Port must terminate any PME_Turn_Off handshake Requests targeting the Port in such a way that the Port is considered to have acknowledged the PME_Turn_Off request (see the Implementation Note in Section 5.3.3.2.1).
- ❑ The Port must handle Vendor Defined Message Requests as described in Table F-1 (e.g., silently discard Vendor Defined Type 1 Messages Requests that it is not designed to receive) since the DL_Down prevents the Request from reaching its targeted Function.
- ❑ for all other Posted Requests, discarding the Requests
  - o This is a reported error associated with the Function for the (virtual) Bridge associated with the Port (see Section 6.2), and must be reported as an Unsupported Request. For Root Ports, the reporting of this error is optional.
  - o For a Posted Request already being processed by the Transaction Layer, the Port is permitted not to report the error.

  Note: This is equivalent to the case where the Request had been Transmitted before the Link status became DL_Down

Note: The point at which a Posted Request becomes "unreportable" is implementation specific.

- ❑ Discarding all Completions submitted by the device core for Transmission

For an Upstream Port, DL_Down status is handled as a reset by:

- ❑ Returning all PCI Express-specific registers, state machines and externally observable state to the specified default or initial conditions (except for registers defined as sticky – see Section 7.4)
- ❑ Discarding all TLPs being processed
- ❑ For Switch and Bridge propagating hot reset to all associated Downstream Ports. In Switches that support Link speeds greater than 5.0 GT/s, the Upstream Port must direct the LTSSM of each Downstream Port to the Hot Reset state, but not hold the LTSSMs in that state. This permits each Downstream Port to begin Link training immediately after its hot reset completes. This behavior is recommended for all Switches.

2.9.2 Transaction Layer Behavior in DL_Up Status

- ❑ DL_Up status indicates that a connection has been established with another component on the associated Link. This section specifies the Transaction Layer's behavior when the Data Link Layer reports entry to the DL_Up status to the Transaction Layer, indicating that the Link is operational. The Transaction Layer of a Port with DL_Up status must accept received TLPs that conform to the other rules of this specification.

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

- ❑ When transitioning from a non-DL_Up status to a DL_Up status and the Auto Slot Power Limit Disable bit is Cleared in the Slot Control Register, the Port must initiate the transmission

of a Set_Slot_Power_Limit Message to the other component on the Link to convey the value programmed in the Slot Power Limit Scale and Value fields of the Slot Capabilities register. This Transmission is optional if the Slot Capabilities register has not yet been initialized.

2.9.3 Transaction Layer Behavior During Downstream Port Containment

During Downstream Port Containment (DPC), the LTSSM associated with the Downstream Port is directed to the Disabled state. Once it reaches the Disabled state, it remains there as long as the DPC Trigger Status bit in the DPC Status register is Set. See Section 6.2.10 for requirements on how long software must leave the Downstream Port in DPC. This section specifies the Transaction Layer's behavior once DPC has been triggered, and as long as the Downstream Port remains in DPC.

- Once DPC has been triggered, no additional (Upstream) TLPs are accepted from the Data Link Layer.
- If the condition that triggered DPC was associated with an Upstream TLP, any subsequent Upstream TLPs that were already accepted from the Data Link Layer must be discarded silently.

The Downstream Port handles (Downstream) TLPs submitted by the device core in the following manner.

- If the condition that triggered DPC was associated with a Downstream TLP, any prior Downstream TLPs are permitted to be dropped silently or transmitted before the Link goes down. Otherwise, the following rules apply.
- For each Non-Posted Request, the Port must return a Completion and discard the Request silently. The Completer ID field must contain the value associated with the Downstream Port.
    - If the DPC Completion Control bit is Set in the DPC Control register, then Completions are generated with Unsupported Request (UR) Completion Status.
    - If the DPC Completion Control bit is Clear, Completions are generated with Completer Abort (CA) Completion Status.
- The Port must terminate any PME_Turn_Off handshake Requests targeting the Port in such a way that the Port is considered to have acknowledged the PME_Turn_Off Request (see the Implementation Note in Section 5.3.3.2.1).
- The Port must handle Vendor Defined Message Requests as described in Table F-1. (e.g., silently discard Vendor Defined Type 1 Message Requests that it is not designed to receive) since the DL_Down prevents the Request from reaching its targeted Function.
- For all other Posted Requests and Completions, the Port must silently discard the TLP.

For any outstanding Non-Posted Requests where DPC being triggered prevents their associated Completions from being returned, the following apply:

- For Root Ports that support RP Extensions for DPC, the Root Port may track certain Non-Posted Requests, and when DPC is triggered, synthesize a Completion for each tracked Request. This helps avoid Completion Timeouts that would otherwise occur as a side-effect of DPC being triggered. Each synthesized Completion must have a UR or CA Completion Status as

determined by the DPC Completion Control bit. The set of Non-Posted Requests that get tracked is implementation-specific, but it is strongly recommended that all Non-Posted Requests that are generated by host processor instructions (e.g., "read", "write", "load", "store", or one that corresponds to an AtomicOp) be tracked. Other candidates for tracking include peer-to-peer Requests coming from other Root Ports and Requests coming from RCiEPs.

- Otherwise, the associated Requesters may encounter Completion Timeouts. The software solution stack should comprehend and account for this possibility.

3

3 Data Link Layer Specification

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

3.1 Data Link Layer Overview

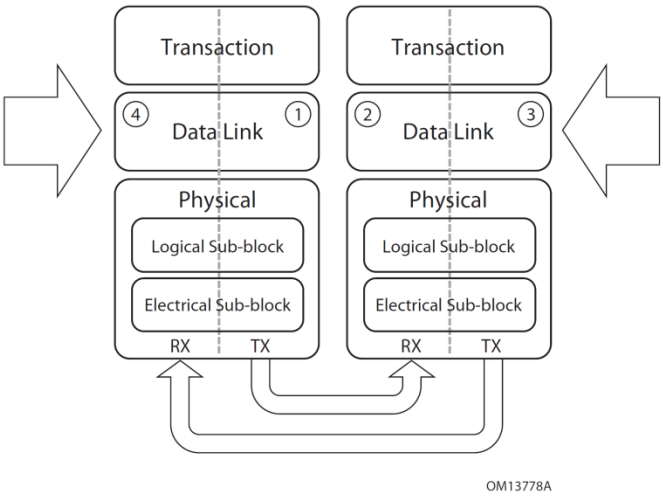


Figure 3-1: Layering Diagram Highlighting the Data Link Layer

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

Data Exchange:

- Accept TLPs for transmission from the Transmit Transaction Layer and convey them to the Transmit Physical Layer
- Accept TLPs received over the Link from the Physical Layer and convey them to the Receive Transaction Layer

Error Detection and Retry:

- TLP Sequence Number and LCRC generation
- Transmitted TLP storage for Data Link Layer Retry
- Data integrity checking for TLPs and Data Link Layer Packets (DLLPs)
- Positive and negative acknowledgement DLLPs
- Error indications for error reporting and logging mechanisms
- Link Acknowledgement Timeout replay mechanism

Initialization and power management:

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

DLLPs are:

- used for Link Management functions including TLP acknowledgement, power management, and exchange of Flow Control information.
- transferred between Data Link Layers of the two directly connected components on a Link

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

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

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

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

The Data Link Layer appears as an information conduit with varying latency to the Transaction Layer. On any given individual Link all TLPs fed into the Transmit Data Link Layer (1 and 3) will appear at the output of the Receive Data Link Layer (2 and 4) in the same order at a later time, as

illustrated in Figure 3-1. The latency will depend on a number of factors, including pipeline latencies, width and operational frequency of the Link, transmission of electrical signals across the Link, and delays caused by Data Link Layer Retry. As a result of these delays, the Transmit Data Link Layer (1 and 3) can apply backpressure to the Transmit Transaction Layer, and the Receive Data Link Layer (2 and 4) communicates the presence or absence of valid information to the Receive Transaction Layer.

3.2 Data Link Control and Management State Machine

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

States:

- DL_Inactive – Physical Layer reporting Link is non-operational or nothing is connected to the Port
- DL_Feature (optional) – Physical Layer reporting Link is operational, perform the Data Link Feature Exchange
- DL_Init – Physical Layer reporting Link is operational, initialize Flow Control for the default Virtual Channel
- DL_Active – Normal operation mode

Status Outputs:

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

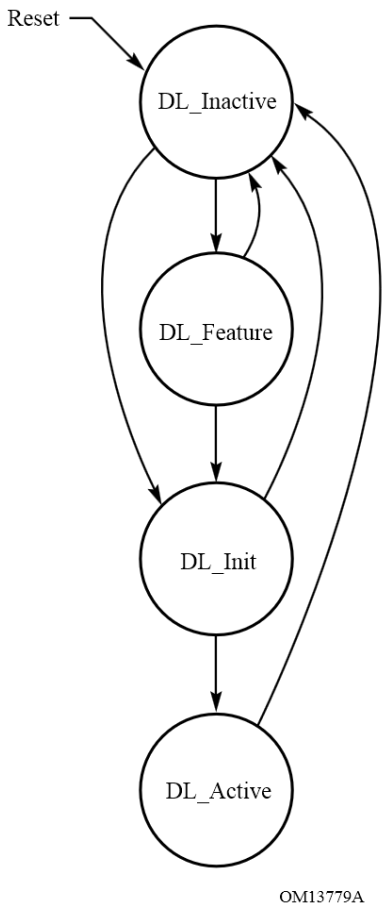


Figure 3-2: Data Link Control and Management State Machine

3.2.1 Data Link Control and Management State Machine Rules

Rules per state:

- DL_Inactive
  - Initial state following PCI Express hot, warm, or cold reset (see Section 6.6). Note that DL states are unaffected by an FLR (see Section 6.6).
  - Upon entry to DL_Inactive
    - Reset all Data Link Layer state information to default values
    - If the Port supports the optional Data Link Feature Exchange, the Remote Data Link Feature Supported, and Remote Data Link Feature Supported Valid fields must be cleared.
    - Discard the contents of the Data Link Layer Retry Buffer (see Section 3.6)
  - While in DL_Inactive:
    - Report DL_Down status to the Transaction Layer as well as to the rest of the Data Link Layer

Note: This will cause the Transaction Layer to discard any outstanding transactions and to terminate internally any attempts to transmit a TLP. For a Downstream Port, this is equivalent to a "Hot-Remove". For an Upstream Port, having the Link go down is equivalent to a hot reset (see Section 2.9).

- Discard TLP information from the Transaction and Physical Layers
- Do not generate or accept DLLPs

- Exit to DL_Feature if:
  - The Port supports the optional Data Link Feature Exchange, the Data Link Feature Exchange Enable bit is Set, the Transaction Layer indicates that the Link is not disabled by software, and the Physical Layer reports Physical LinkUp = 1b
- Exit to DL_Init if:
  - The Port does not support the optional Data Link Feature Exchange, the Transaction Layer indicates that the Link is not disabled by software and the Physical Layer reports Physical LinkUp = 1b

    or
  - The Port supports the optional Data Link Feature Exchange, the Data Link Feature Exchange Enable bit is Clear, the Transaction Layer indicates that the Link is not disabled by software, and the Physical Layer reports Physical LinkUp = 1b

- DL_Feature
  - While in DL_Feature:
    - Perform the Data Link Feature Exchange protocol as described in Section 3.3
    - Report DL_Down status
    - The Data Link Layer of a Port with DL_Down status is permitted to discard any received TLPs provided that it does not acknowledge those TLPs by sending one or more Ack DLLPs
  - Exit to DL_Init if:
    - Data Link Feature Exchange completes successfully, and the Physical Layer continues to report Physical LinkUp = 1b,

      or
    - Data Link Feature Exchange determines that the remote Data Link Layer does not support the optional Data Link Feature Exchange protocol, and the Physical Layer continues to report Physical LinkUp = 1b
  - Terminate the Data Link Feature Exchange protocol and exit to DL_Inactive if:
    - Physical Layer reports Physical LinkUp = 0b
- DL_Init
  - While in DL_Init:

        - Initialize Flow Control for the default Virtual Channel, VC0, following the Flow Control initialization protocol described in Section 3.4
        - Report DL_Down status while in state FC_INIT1; DL_Up status in state FC_INIT2
        - The Data Link Layer of a Port with DL_Down status is permitted to discard any received TLPs provided that it does not acknowledge those TLPs by sending one or more Ack DLLPs
    - Exit to DL_Active if:
        - Flow Control initialization completes successfully, and the Physical Layer continues to report Physical LinkUp = 1b
    - Terminate attempt to initialize Flow Control for VC0 and exit to DL_Inactive if:
        - Physical Layer reports Physical LinkUp = 0b
- DL_Active
    - DL_Active is referred to as the normal operating state
    - While in DL_Active:
        - Accept and transfer TLP information with the Transaction and Physical Layers as specified in this chapter
        - Generate and accept DLLPs as specified in this chapter
        - Report DL_Up status to the Transaction and Data Link Layers
    - Exit to DL_Inactive if:
        - Physical Layer reports Physical LinkUp = 0b
        - Downstream Ports that are Surprise Down Error Reporting Capable (see Section 7.5.3.6) must treat this transition from DL_Active to DL_Inactive as a Surprise Down error, except in the following cases where this error detection is blocked:
            - If the Secondary Bus Reset bit in the Bridge Control register has been Set by software, then the subsequent transition to DL_Inactive must not be considered an error.
            - If the Link Disable bit has been Set by software, then the subsequent transition to DL_Inactive must not be considered an error.
            - If a Switch Downstream Port transitions to DL_Inactive due to an event above that Port, that transition to DL_Inactive must not be considered an error. Example events include the Switch Upstream Port propagating Hot Reset, the Switch Upstream Link transitioning to DL_Down, and the Secondary Bus Reset bit in the Switch Upstream Port being Set.
            - If a PME_Turn_Off Message has been sent through this Port, then the subsequent transition to DL_Inactive must not be considered an error.

- Note that the DL_Inactive transition for this condition will not occur until a power off, a reset, or a request to restore the Link is sent to the Physical Layer.
- Note also that in the case where the PME_Turn_Off/PME_TO_Ack handshake fails to complete successfully, a Surprise Down error may be detected.
- If the Port is associated with a hot-pluggable slot (the Hot-Plug Capable bit in the Slot Capabilities register Set), and the Hot-Plug Surprise bit in the Slot Capabilities register is Set, then any transition to DL_Inactive must not be considered an error.
- If the Port is associated with a hot-pluggable slot (Hot-Plug Capable bit in the Slot Capabilities register Set), and Power Controller Control bit in Slot Control register is Set (Power-Off), then any transition to DL_Inactive must not be considered an error.

Error blocking initiated by one or more of the above cases must remain in effect until the Port exits DL_Active and subsequently returns to DL_Active with none of the blocking cases in effect at the time of the return to DL_Active.

Note that the transition out of DL_Active is simply the expected transition as anticipated per the error detection blocking condition.

If implemented, this is a reported error associated with the detecting Port (see Section 6.2).

IMPLEMENTATION NOTE

Physical Layer Throttling

Note that there are conditions where the Physical Layer may be temporarily unable to accept TLPs and DLLPs from the Data Link Layer. The Data Link Layer must comprehend this by providing mechanisms for the Physical Layer to communicate this condition, and for TLPs and DLLPs to be temporarily blocked by the condition.

3.3 Data Link Feature Exchange

The Data Link Feature Exchange protocol is optional. Ports that implement this protocol contain the Data Link Feature Extended Capability (see Section 7.7.4). This capability contains four fields:

- The Local Data Link Feature Supported field indicates the Data Link Features supported by the local Port
- The Remote Data Link Feature Supported field indicates the Data Link Features supported by the remote Port
- The Remote Data Link Feature Supported Valid bit indicates that the Remote Data Link Feature Supported field contains valid data

- The Data Link Feature Exchange Enable field permits systems to disable the Data Link Feature Exchange. This can be used to work around legacy hardware that does not correctly ignore the DLLP.

The Data Link Feature Exchange protocol transmits a Port's Local Feature Supported information to the Remote Port and captures that Remote Port's Feature Supported information.

Rules for this protocol are:

- On entry to DL_Feature:
  - The Remote Data Link Feature Supported and Remote Data Link Feature Supported Valid fields must be Cleared
- While in DL_Feature:
  - Transaction Layer must block transmission of TLPs
  - Transmit the Data Link Feature DLLP
    - The transmitted Feature Supported field must equal the Local Data Link Feature Supported field.
    - The transmitted Feature Ack bit must equal the Remote Data Link Feature Supported Valid bit.
  - The Data Link Feature DLLP must be transmitted at least once every 34 µs. Time spent in the Recovery or Configuration LTSSM states does not contribute to this limit.
  - Process received Data Link Feature DLLPs:
    - If the Remote Data Link Feature Supported Valid bit is Clear, record the Feature Supported field from the received Data Link Feature DLLP in the Remote Data Link Feature Supported field and Set the Remote Data Link Feature Supported Valid bit.
- Exit DL_Feature if:
  - An InitFC1 DLLP has been received.
  - An MR-IOV MRInit DLLP (encoding 0000 0001b) has been received.

    or
  - While in DL_Feature, at least one Data Link Feature DLLP has been received with the Feature Ack bit Set.

Each Data Link Feature has an associated bit in the Feature Supported field. A Data Link Feature is activated when that bit is Set in both the Local Data Link Feature Supported and Remote Data Link Feature Supported fields.

Data Link Features and their corresponding bit locations are shown in Table 3-1.

Table 3-1: Data Link Feature Supported Bit Definition

Bit Location	Description
0	**Scaled Flow Control** – indicates support for Scaled Flow Control. Scaled Flow Control must be supported in Ports that support 16.0 GT/s or higher operation.
22:1	Reserved

3.4 Flow Control Initialization Protocol

Before starting normal operation following power-up or interconnect reset, it is necessary to initialize Flow Control for the default Virtual Channel, VC0 (see Section 6.6). In addition, when additional Virtual Channels (VCs) are enabled, the Flow Control initialization process must be completed for each newly enabled VC before it can be used (see Section 2.6.1). This section describes the initialization process that is used for all VCs. Note that since VC0 is enabled before all other VCs, no TLP traffic of any kind will be active prior to initialization of VC0. However, when additional VCs are being initialized there will typically be TLP traffic flowing on other, already enabled, VCs. Such traffic has no direct effect on the initialization process for the additional VC(s).

There are two states in the VC initialization process. These states are:

- FC_INIT1
- FC_INIT2

The rules for this process are given in the following section.

3.4.1 Flow Control Initialization State Machine Rules

- If at any time during initialization for VCs 1-7 the VC is disabled, the flow control initialization process for the VC is terminated
- Rules for state FC_INIT1:
  - Entered when initialization of a VC (VCx) is required
    - Entrance to DL_Init state (VCx = VC0)
    - When a VC (VCx = VC1-7) is enabled by software (see Sections 7.9.1 and 7.9.2)
  - While in FC_INIT1:
    - Transaction Layer must block transmission of TLPs using VCx
    - Transmit the following three InitFC1 DLLPs for VCx in the following relative order:
      - InitFC1–P (first)
      - InitFC1–NP (second)

        - InitFC1–Cpl (third)
    - The three InitFC1 DLLPs must be transmitted at least once every 34 µs.
        - Time spent in the Recovery or Configuration LTSSM states does not contribute to this limit.
        - It is strongly encouraged that the InitFC1 DLLP transmissions are repeated frequently, particularly when there are no other TLPs or DLLPs available for transmission.
    - If Scaled Flow Control is activated on the Link, set the HdrScale and DataScale fields in the InitFC1 DLLPs to 01b, 10b, or 11b to indicate the scaling factor it is using on the corresponding HdrFC and DataFC values.
    - If the Transmitter does not support Scaled Flow Control or if Scaled Flow Control is not activated on the Link, set the HdrScale and DataScale fields to 00b.
    - Except as needed to ensure at least the required frequency of InitFC1 DLLP transmission, the Data Link Layer must not block other transmissions
        - Note that this includes all Physical Layer initiated transmissions (for example, Ordered Sets), Ack and Nak DLLPs (when applicable), and TLPs using VCs that have previously completed initialization (when applicable)
    - Process received InitFC1 and InitFC2 DLLPs:
        - Record the indicated HdrFC and DataFC values
        - If the Receiver supports Scaled Flow Control, record the indicated HdrScale and DataScale values.
        - Set flag FI1 once FC unit values have been recorded for each of P, NP, and Cpl for VCx
  - Exit to FC_INIT2 if:
    - Flag FI1 has been set indicating that FC unit values have been recorded for each of P, NP, and Cpl for VCx
- Rules for state FC_INIT2:
  - While in FC_INIT2:
    - Transaction Layer must block transmission of TLPs using VCx
    - Transmit the following three InitFC2 DLLPs for VCx in the following relative order:
        - InitFC2– (first)
        - InitFC2–NP (second)
        - InitFC2–Cpl (third)
    - The three InitFC2 DLLPs must be transmitted at least once every 34 µs.

        - Time spent in the Recovery or Configuration LTSSM states does not contribute to this limit.
        - It is strongly encouraged that the InitFC2 DLLP transmissions are repeated frequently, particularly when there are no other TLPs or DLLPs available for transmission.
    - If Scaled Flow Control is activated on the Link, set the HdrScale and DataScale fields in the InitF2 DLLPs to 01b, 10b, or 11b to indicate the scaling factor it is using on the corresponding HdrFC and DataFC values.
    - If the Transmitter does not support Scaled Flow Control or if Scaled Flow Control is not activated on the Link, set the HdrScale and DataScale fields to 00b.
    - Except as needed to ensure at least the required frequency of InitFC2 DLLP transmission, the Data Link Layer must not block other transmissions
        - Note that this includes all Physical Layer initiated transmissions (for example, Ordered Sets), Ack and Nak DLLPs (when applicable), and TLPs using VCs that have previously completed initialization (when applicable)
    - Process received InitFC1 and InitFC2 DLLPs:
        - Ignore the received HdrFC, HdrScale, DataFC, and DataScale values
        - Set flag FI2 on receipt of any InitFC2 DLLP for VCx
    - Set flag FI2 on receipt of any TLP on VCx, or any UpdateFC DLLP for VCx
- Signal completion and exit if:
    - Flag FI2 has been Set
    - If Scaled Flow Control is activated on the Link, the Transmitter must send 01b, 10b, or 11b for HdrScale and DataScale in all UpdateFC DLLPs for VCx.
    - If the Scaled Flow Control is not supported or if Scaled Flow Control is not activated on the Link, the Transmitter must send 00b for HdrScale and DataScale in all UpdateFC DLLPs for VCx.

IMPLEMENTATION NOTE

Example of Flow Control Initialization

Figure 3-3 illustrates an example of the Flow Control initialization protocol for VC0 between a Switch and a Downstream component. In this example, each component advertises the minimum permitted values for each type of Flow Control credit. For both components the largest Max_Payload_Size value supported is 1024 bytes, corresponding to a data payload credit advertisement of 040h. All DLLPs are shown as received without error.

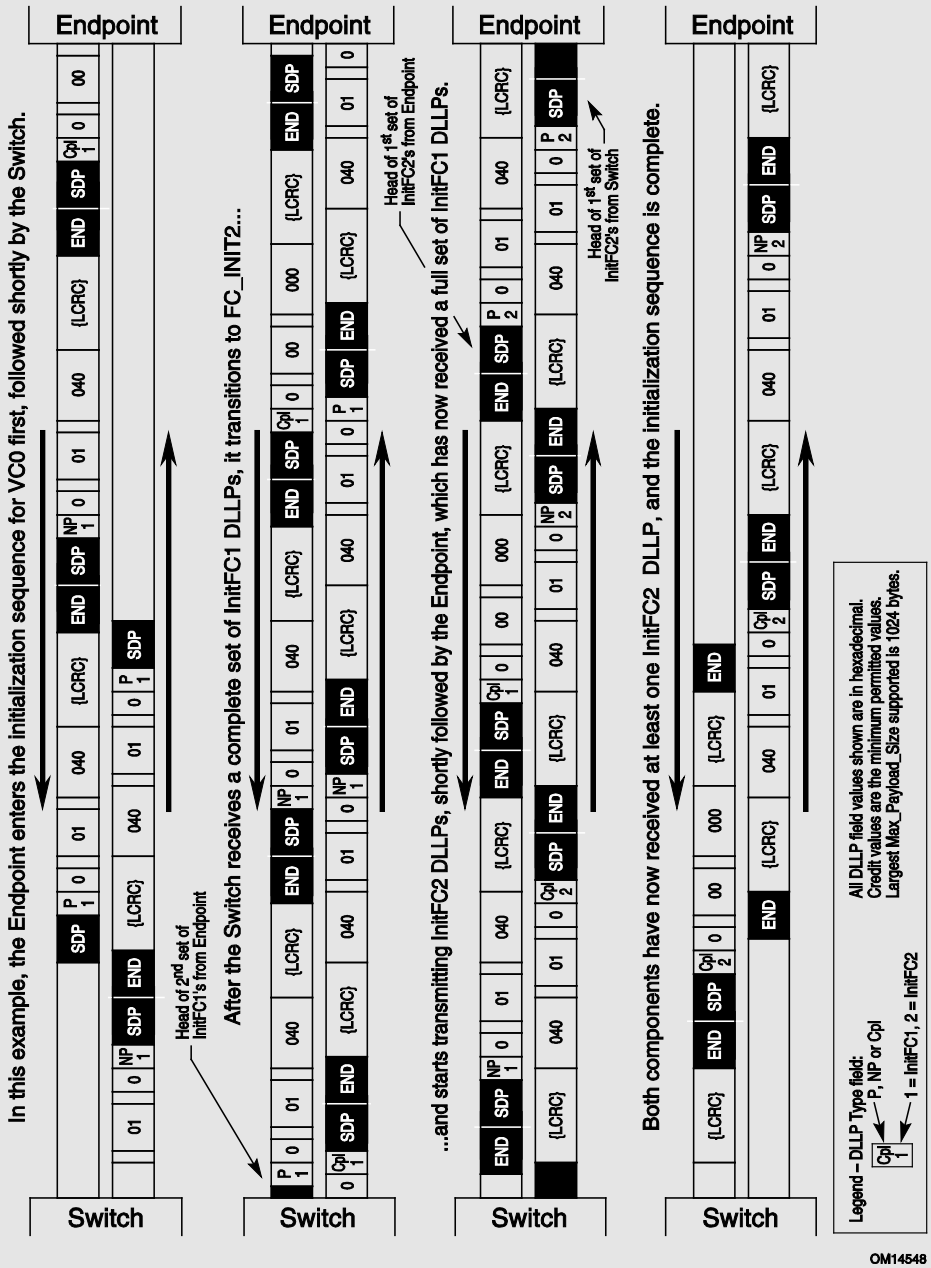


Figure 3-3: VC0 Flow Control Initialization Example with 8b/10b Encoding-based Framing

3.4.2 Scaled Flow Control

Link performance can be affected when there are insufficient flow control credits available to account for t he Link round trip time. This effect becomes more noticeable at higher Link speeds and the limitation of 127 header credits and 2047 data credits can limit performance. The Scaled Flow Control mechanism is designed to address this limitation.

All Ports are permitted to support Scaled Flow Control. Ports that support 16.0 GT/s must support Scaled Flow Control. Scaled Flow Control activation does not affect the ability to operate at 16.0 GT/s.

The following rules apply when Scaled Flow Control is not activated for the Link:

- The InitFC1, InitFC2, and UpdateFC DLLPs must contain 00b in the HdrScale and DataScale fields.
- The HdrFC counter is 8 bits wide and the HdrFC DLLP field includes all bits of the counter.
- The DataFC counter is 12 bits wide and the DataFC DLLP field includes all bits of the counter.

The following rules apply when Scaled Flow Control is activated for the Link:

- The InitFC1 and InitFC2 DLLPs must contain 01b, 10b, or 11b in the HdrScale field. The value is determined by the maximum number of header credits that will be outstanding of the indicated credit type as defined in Table 3-2.
- The InitFC1 and InitFC2 DLLPs must contain 01b, 10b, or 11b in the DataScale field. The value is determined by the maximum number of data payload credits that will be outstanding of the indicated credit type as defined in Table 3-2.
- If the received HdrScale and DataScale values recorded in state FC_INIT1 were non-zero, then Scaled Flow Control is enabled on this VC and UpdateFC DLLPs must contain 01b, 10b, or 11b in the HdrScale and DataScale fields.
- If the received HdrScale and DataScale values recorded in state FC_INIT1 were zero, then Scaled Flow Control is not enabled on this VC and UpdateFC DLLPs must contain 00b in the HdrScale and DataScale fields.

Table 3-2: Scaled Flow Control Scaling Factors

Scale Factor	Scaled Flow Control Supported	Header Credits					Data Credits				
		Min Credits	Max Credits	HdrFC Width	HdrFC DLLP field		Min Credits	Max Credits	DataFC Width	DataFC DLLP Field	
					Transmitted	Received				Transmitted	Received
00b	No	1	127	8 bits	HdrFC	HdrFC	1	2047	12 bits	DataFC	DataFC
01b	Yes	1	127	8 bits	HdrFC	HdrFC	1	2047	12 bits	DataFC	DataFC
10b	Yes	4	508	10 bits	HdrFC>>2	HdrFC<<2	4	8188	14 bits	DataFC>>2	DataFC<<2
11b	Yes	16	2032	12 bits	HdrFC>>4	HdrFC<<4	16	32,752	16 bits	DataFC>>4	DataFC<<4

3.5 Data Link Layer Packets (DLLPs)

The following DLLPs are used to support Link operations:

- Ack DLLP: TLP Sequence Number acknowledgement; used to indicate successful receipt of some number of TLPs
- Nak DLLP: TLP Sequence Number negative acknowledgement; used to initiate a Data Link Layer Retry
- InitFC1, InitFC2, and UpdateFC DLLPs: For Flow Control
- DLLPs used for Power Management

3.5.1 Data Link Layer Packet Rules

All DLLP fields marked Reserved (sometimes abbreviated as R) must be filled with all 0's when a DLLP is formed. Values in such fields must be ignored by Receivers. The handling of Reserved values in encoded fields is specified for each case.

All DLLPs include the following fields:

- DLLP Type - Specifies the type of DLLP. The defined encodings are shown in Table 3-3.
- 16-bit CRC

See Figure 3-4 below.

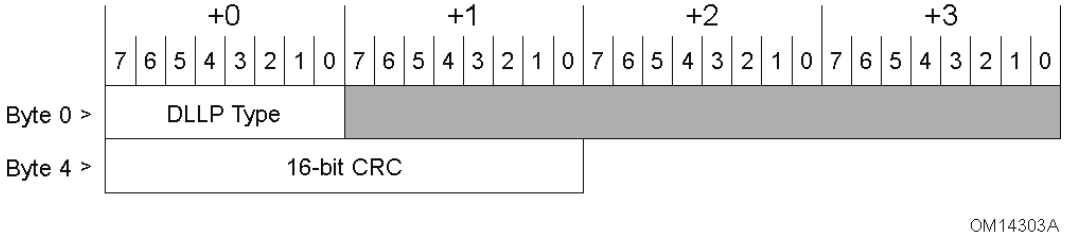


Figure 3-4: DLLP Type and CRC Fields

Table 3-3: DLLP Type Encodings

Encodings (b)	DLLP Type
0000 0000	Ack
0000 0001	MRInit – See the MR-IOV Specification[41]
0000 0010	Data_Link_Feature
0001 0000	Nak
0010 0000	PM_Enter_L1
0010 0001	PM_Enter_L23
0010 0011	PM_Active_State_Request_L1
0010 0100	PM_Request_Ack
0011 0000	Vendor-specific
0011 0001	NOP
0100 $0v_2v_1v_0$	InitFC1-P (v[2:0] specifies Virtual Channel)
0101 $0v_2v_1v_0$	InitFC1-NP
0110 $0v_2v_1v_0$	InitFC1-Cpl
0111 $0v_2v_1v_0$	MRInitFC1 (v[2:0] specifies Virtual Link) – See the MR-IOV Specification[42]
1100 $0v_2v_1v_0$	InitFC2-P
1101 $0v_2v_1v_0$	InitFC2-NP
1110 $0v_2v_1v_0$	InitFC2-Cpl
1111 $0v_2v_1v_0$	MRInitFC2 – See the MR-IOV Specification[42]
1000 $0v_2v_1v_0$	UpdateFC-P
1001 $0v_2v_1v_0$	UpdateFC-NP
1010 $0v_2v_1v_0$	UpdateFC-Cpl
1011 $0v_2v_1v_0$	MRUpdateFC – See the MR-IOV Specification[42]
All other encodings	Reserved

- For Ack and Nak DLLPs (see Figure 3-5):
  - The AckNak_Seq_Num field is used to indicate what TLPs are affected
  - Transmission and reception is handled by the Data Link Layer according to the rules provided in Section 3.6.
- For InitFC1, InitFC2, and UpdateFC DLLPs:
  - The HdrFC field contains the credit value for headers of the indicated type (P, NP, or Cpl).
  - The DataFC field contains the credit value for payload Data of the indicated type (P, NP, or Cpl).
  - The HdrScale field contains the scaling factor for headers of the indicated type. Encodings are defined in Table 3-4.

[41] The MR-IOV protocol uses this encoding for the MRInit negotiation. The MR-IOV protocol assumes that non-MR-IOV components will silently ignore these DLLPs..

[42] The MR-IOV protocol uses these encodings after the successful completion of MRInit negotiation.

- The DataScale field contains the scaling factor for payload data of the indicated type. Encodings are defined in Table 3-4.
- If Scaled Flow Control is activated, the HdrScale and DataScale fields must be set to 01b, 10b, or 11b in all InitFC1, InitFC2, and UpdateFC DLLPs transmitted.
- In UpdateFCs, a Transmitter is only permitted to send non-zero values in the HdrScale and DataScale fields if it supports Scaled Flow Control and it received non-zero values for HdrScale and DataScale in the InitFC1s and InitFC2s it received for this VC.
- The packet formats are shown in Figure 3-7, Figure 3-8, and Figure 3-9.
- Transmission is triggered by the Data Link Layer when initializing Flow Control for a Virtual Channel (see Section 3.4), and following Flow Control initialization by the Transaction Layer according to the rules in Section 2.6.
- Checked for integrity on reception by the Data Link Layer and if correct, the information content of the DLLP is passed to the Transaction Layer. If the check fails, the information is discarded.

Note: InitFC1 and InitFC2 DLLPs are used only for VC initialization

Table 3-4: HdrScale and DataScale Encodings

HdrScale or DataScale Value	Scaled Flow Control Supported	Scaling Factor	HdrFC DLLP Field	DataFC DLLP Field
00b	No	1	HdrFC[7:0]	DataFC[11:0]
01b	Yes	1	HdrFC[7:0]	DataFC[11:0]
10b	Yes	4	HdrFC[9:2]	DataFC[13:2]
11b	Yes	16	HdrFC[11:4]	DataFC[15:4]

- Power Management (PM) DLLPs (see Figure 3-10):
  - Transmission is triggered by the component's power management logic according to the rules in Chapter 5
  - Checked for integrity on reception by the Data Link Layer, then passed to the component's power management logic
- Vendor-specific DLLP (see Figure 3-11)
  - It is recommended that receivers silently ignore Vendor Specific DLLPs unless enabled by implementation specific mechanisms.
  - It is recommended that transmitters not send Vendor Specific DLLPs unless enabled by implementation specific mechanisms.
- NOP DLLP (see Figure 3-6)
  - Receivers shall discard this DLLP without action after checking it for data integrity.[43]
- Data Link Feature (see Figure 3-12)

[43] This is a special case of the more general rule for unsupported DLLP Type encodings (see Section 3.6.2.2).

- The Feature Ack bit is set to indicate that the transmitting Port has received a Data Link Feature DLLP.
- The Feature Supported bits indicate the features supported by the transmitting Port. These bits equal the value of the Local Data Link Feature Supported field (see Section 7.7.4.2)

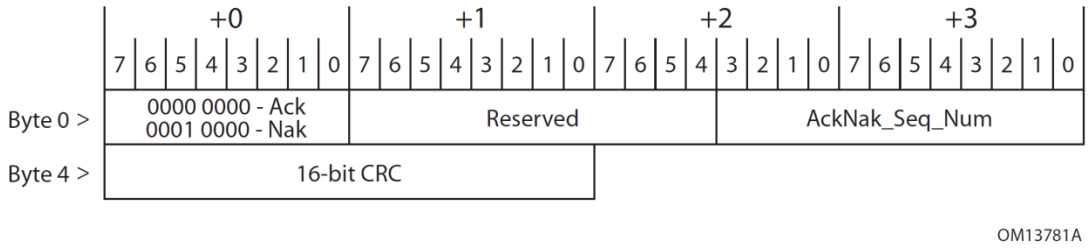


Figure 3-5: Data Link Layer Packet Format for Ack and Nak

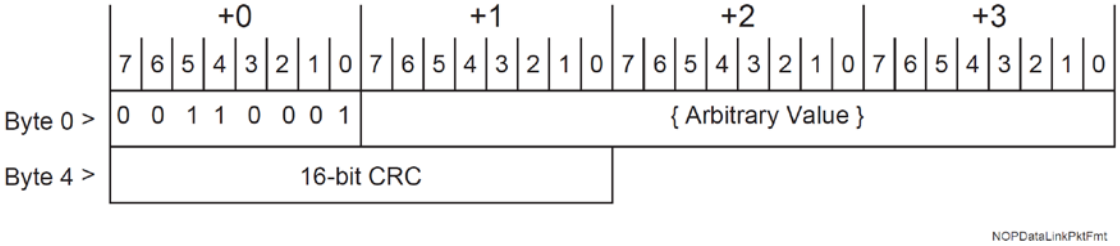


Figure 3-6: NOP Data Link Layer Packet Format

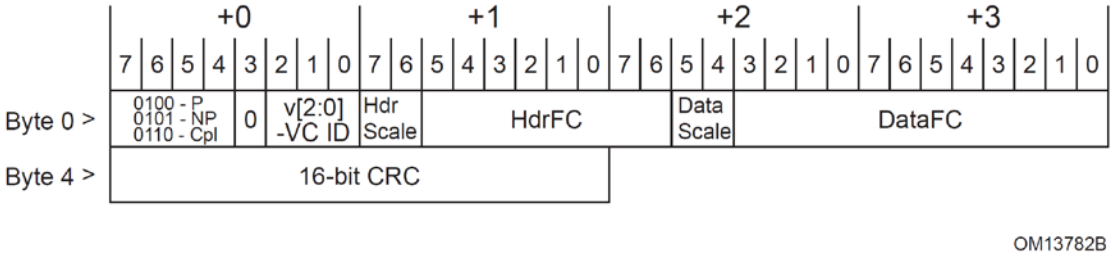


Figure 3-7: Data Link Layer Packet Format for InitFC1

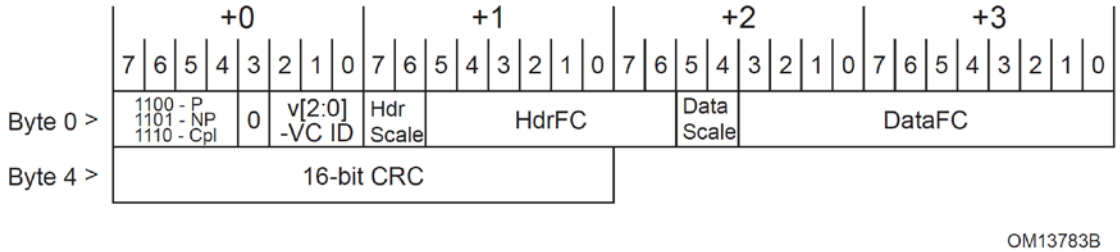


Figure 3-8: Data Link Layer Packet Format for InitFC2

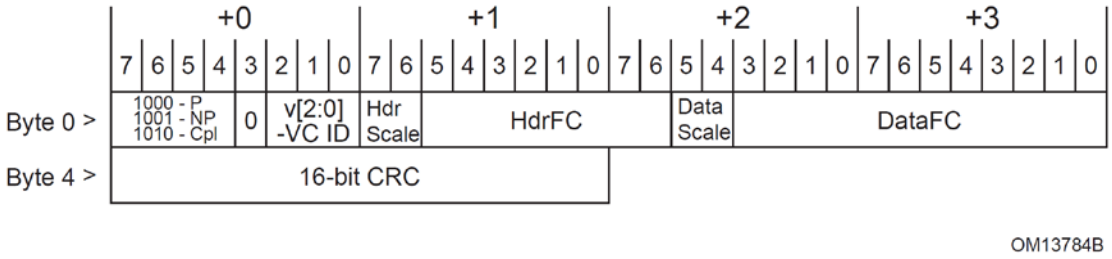


Figure 3-9: Data Link Layer Packet Format for UpdateFC

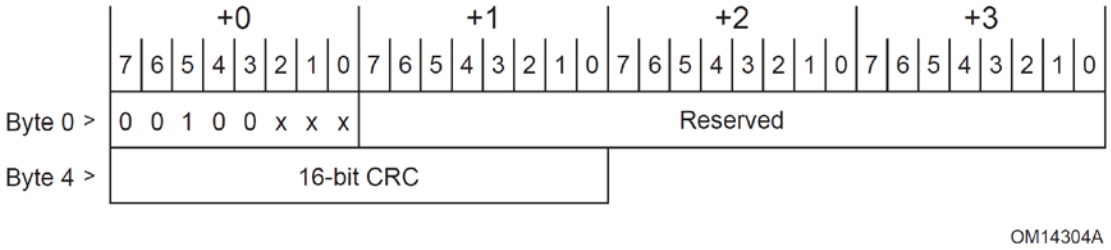


Figure 3-10: PM Data Link Layer Packet Format

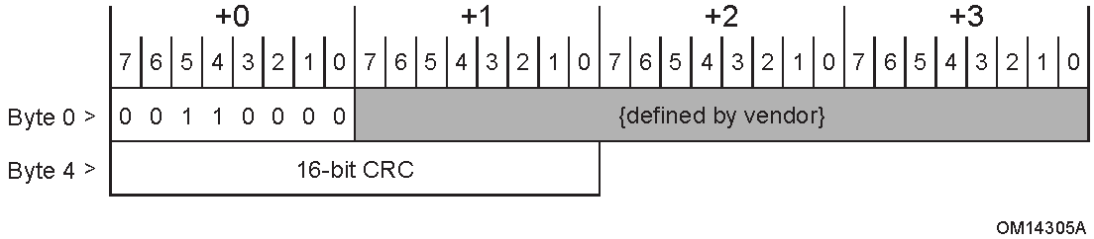


Figure 3-11: Vendor-specific Data Link Layer Packet Format

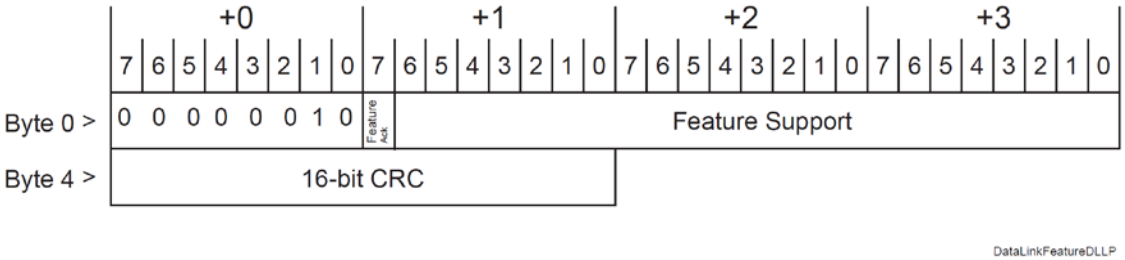


Figure 3-12: Data Link Feature DLLP

The following are the characteristics and rules associated with Data Link Layer Packets (DLLPs):

- DLLPs are differentiated from TLPs when they are presented to, or received from, the Physical Layer.
- DLLP data integrity is protected using a 16-bit CRC
- The CRC value is calculated using the following rules (see Figure 3-13):
  - The polynomial used for CRC calculation has a coefficient expressed as 100Bh

- The seed value (initial value for CRC storage registers) is FFFFh
- CRC calculation starts with bit 0 of byte 0 and proceeds from bit 0 to bit 7 of each byte
- Note that CRC calculation uses all bits of the DLLP, regardless of field type, including Reserved fields. The result of the calculation is complemented, then placed into the 16-bit CRC field of the DLLP as shown in Table 3-5.

Table 3-5: Mapping of Bits into CRC Field

CRC Result Bit	Corresponding Bit Position in the 16-Bit CRC Field
0	7
1	6
2	5
3	4
4	3
5	2
6	1
7	0
8	15
9	14
10	13
11	12
12	11
13	10
14	9
15	8

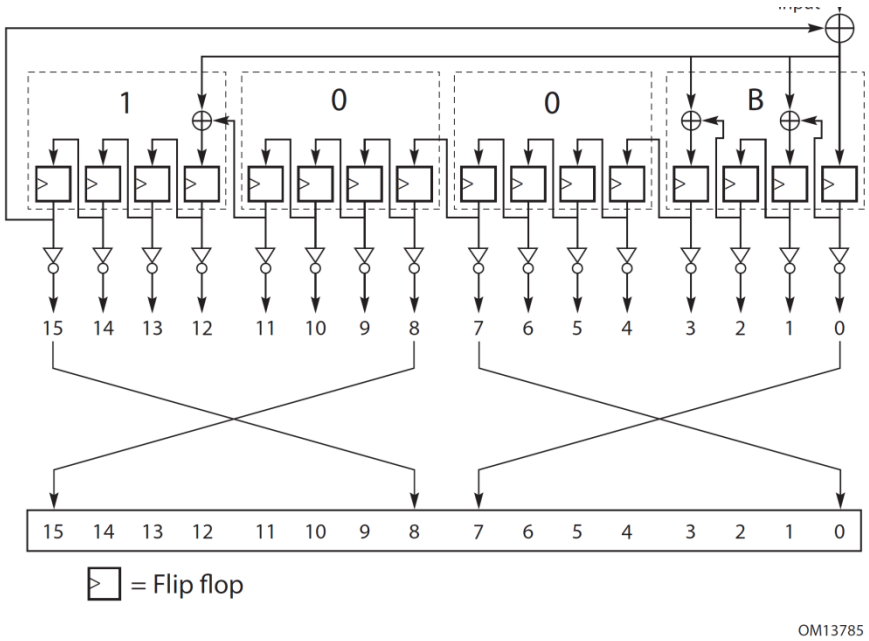


Figure 3-13: Diagram of CRC Calculation for DLLPs

3.6 Data Integrity

3.6.1 Introduction

The Transaction Layer provides TLP boundary information to the Data Link Layer. This allows the Data Link Layer to apply a TLP Sequence Number and a Link CRC (LCRC) for error detection to the TLP. The Receive Data Link Layer validates received TLPs by checking the TLP Sequence Number, LCRC code and any error indications from the Receive Physical Layer. In case any of these errors are in a TLP, Data Link Layer Retry is used for recovery.

The format of a TLP with the TLP Sequence Number and LCRC code applied is shown in Figure 3-14.

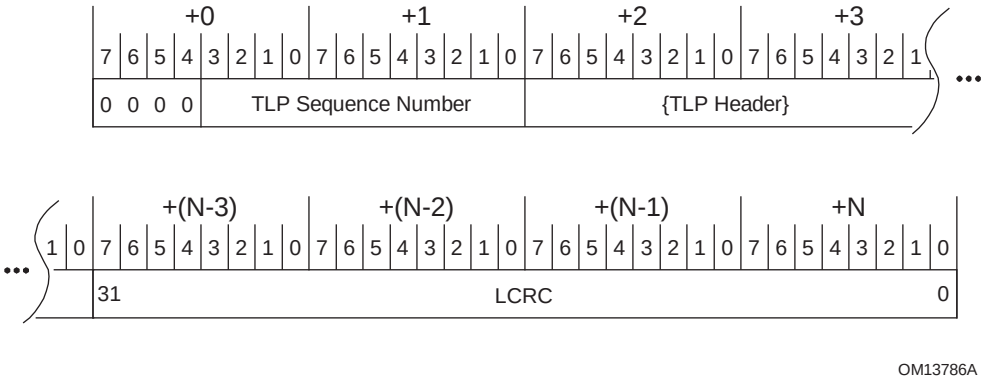


Figure 3-14: TLP with LCRC and TLP Sequence Number Applied

On Ports that support Protocol Multiplexing, packets containing a non-zero value in Symbol +0, bits 7:4 are PMUX Packets. For TLPs, these bits must be 0000b. See Appendix G for details.

On Ports that do not support Protocol Multiplexing, Symbol +0, bits 7:4 are Reserved.

3.6.2 LCRC, Sequence Number, and Retry Management (TLP Transmitter)

The TLP transmission path through the Data Link Layer (paths labeled 1 and 3 in Figure 3-1) prepares each TLP for transmission by applying a sequence number, then calculating and appending a Link CRC (LCRC) which is used to ensure the integrity of TLPs during transmission across a Link from one component to another. TLPs are stored in a retry buffer, and are re-sent unless a positive acknowledgement of receipt is received from the other component. If repeated attempts to transmit a TLP are unsuccessful, the Transmitter will determine that the Link is not operating correctly, and instruct the Physical Layer to retrain the Link (via the LTSSM Recovery state, Section 4.2.6). If Link retraining fails, the Physical Layer will indicate that the Link is no longer up, causing the DLCMSM to move to the DL_Inactive state.

The mechanisms used to determine the TLP LCRC and the Sequence Number and to support Data Link Layer Retry are described in terms of conceptual "counters" and "flags". This description does not imply nor require a particular implementation and is used only to clarify the requirements.

3.6.2.1 *LCRC and Sequence Number Rules (TLP Transmitter)*

The following counters and timer are used to explain the remaining rules in this section:

- The following 12-bit counters are used:

  NEXT_TRANSMIT_SEQ – Stores the packet sequence number applied to TLPs

  - Set to 000h in DL_Inactive state

  - ACKD_SEQ – Stores the sequence number acknowledged in the most recently received Ack or Nak DLLP.
    - Set to FFFh in DL_Inactive state

- The following 2-bit counter is used:

  REPLAY_NUM – Counts the number of times the Retry Buffer has been re-transmitted

  - Set to 00b in DL_Inactive state

- The following timer is used:

  - REPLAY_TIMER - Counts time that determines when a replay is required, according to the following rules:
    - Started at the last Symbol of any TLP transmission or retransmission, if not already running
    - For each replay, reset and restart REPLAY_TIMER when sending the last Symbol of the first TLP to be retransmitted

- Resets and restarts for each Ack DLLP received while there are more unacknowledged TLPs outstanding, if, and only if, the received Ack DLLP acknowledges some TLP in the retry buffer
    - Note: This ensures that REPLAY_TIMER is reset only when forward progress is being made
- Reset and hold until restart conditions are met for each Nak received (except during a replay) or when the REPLAY_TIMER expires
- Not advanced during Link retraining (holds its value when the LTSSM is in the Recovery or Configuration state). Refer to Sections 4.2.5.3 and 4.2.5.4.
- If Protocol Multiplexing is supported, optionally not advanced during the reception of PMUX Packets (see Appendix G).
- Resets and holds when there are no outstanding unacknowledged TLPs

The following rules describe how a TLP is prepared for transmission before being passed to the Physical Layer:

- The Transaction Layer indicates the start and end of the TLP to the Data Link Layer while transferring the TLP
    - The Data Link Layer treats the TLP as a "black box" and does not process or modify the contents of the TLP
- Each TLP is assigned a 12-bit sequence number when it is accepted from the Transmit side of Transaction Layer
    - Upon acceptance of the TLP from the Transaction Layer, the packet sequence number is applied to the TLP by:
        - prepending the 12-bit value in NEXT_TRANSMIT_SEQ to the TLP
        - prepending 4 Reserved bits to the TLP, preceding the sequence number (see Figure 3-15)
    - If the equation:

    (NEXT_TRANSMIT_SEQ – ACKD_SEQ) mod 4096 >= 2048

    is true, the Transmitter must cease accepting TLPs from the Transaction Layer until the equation is no longer true

    - Following the application of NEXT_TRANSMIT_SEQ to a TLP accepted from the Transmit side of the Transaction Layer, NEXT_TRANSMIT_SEQ is incremented (except in the case where the TLP is nullified):

    NEXT_TRANSMIT_SEQ := (NEXT_TRANSMIT_SEQ + 1) mod 4096

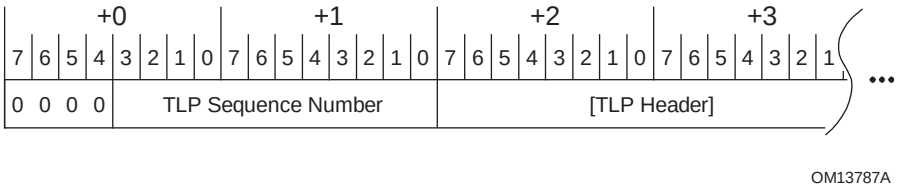


Figure 3-15: TLP Following Application of TLP Sequence Number and Reserved Bits

- TLP data integrity is protected during transfer between Data Link Layers using a 32-bit LCRC
- The LCRC value is calculated using the following mechanism (Figure 3-16):
  - The polynomial used has coefficients expressed as 04C1 1DB7h
  - The seed value (initial value for LCRC storage registers) is FFFF FFFFh
  - The LCRC is calculated using the TLP following sequence number application (see Figure 3-15)
  - LCRC calculation starts with bit 0 of byte 0 (bit 8 of the TLP sequence number) and proceeds from bit 0 to bit 7 of each successive byte.
  - Note that LCRC calculation uses all bits of the TLP, regardless of field type, including Reserved fields
  - The remainder of the LCRC calculation is complemented, and the complemented result bits are mapped into the 32-bit LCRC field as shown in Table 3-6.

Table 3-6: Mapping of Bits into LCRC Field

LCRC Result Bit	Corresponding Bit Position in the 32-Bit LCRC Field
0	7
1	6
2	5
3	4
4	3
5	2
6	1
7	0
8	15
9	14
10	13
11	12
12	11
13	10
14	9
15	8
16	23
17	22
18	21

LCRC Result Bit	Corresponding Bit Position in the 32-Bit LCRC Field
19	20
20	19
21	18
22	17
23	16
24	31
25	30
26	29
27	28
28	27
29	26
30	25
31	24

The 32-bit LCRC field is appended to the TLP following the bytes received from the Transaction Layer (see Figure 3-14).

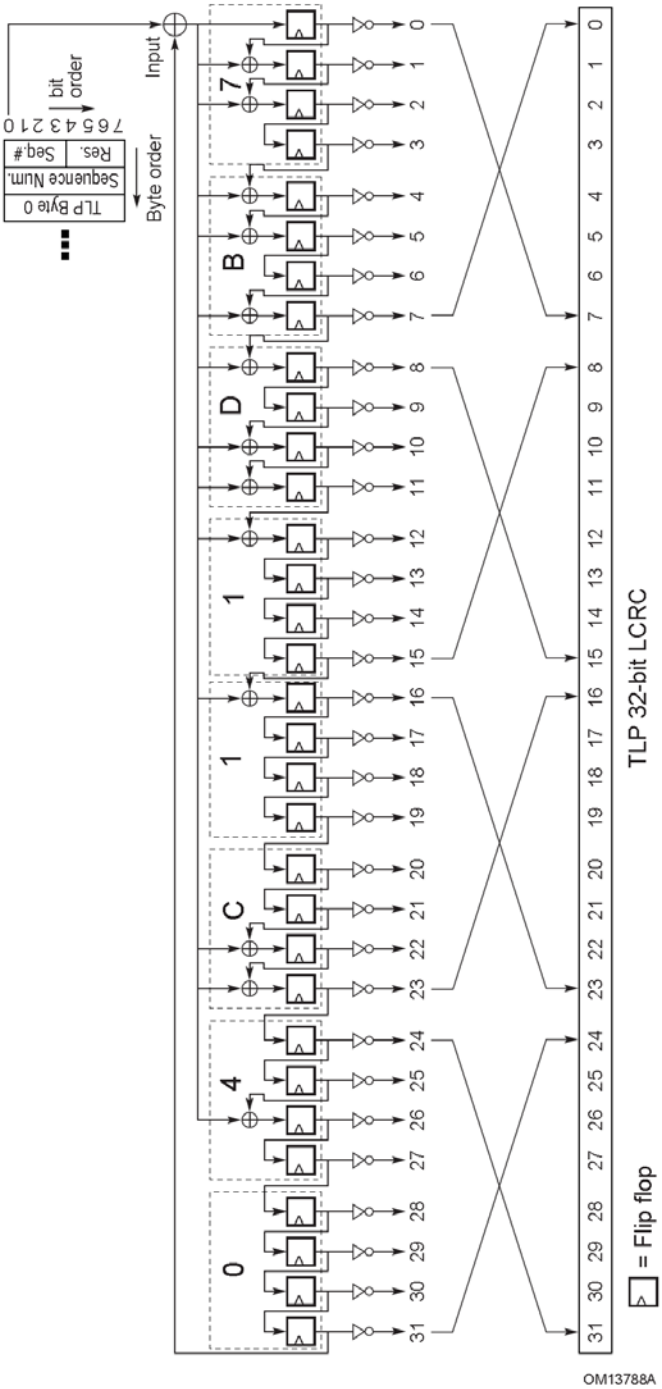


Figure 3-16: Calculation of LCRC

To support cut-through routing of TLPs, a Transmitter is permitted to modify a transmitted TLP to indicate that the Receiver must ignore that TLP ("nullify" the TLP).

- A Transmitter is permitted to nullify a TLP being transmitted; to do this in a way which will robustly prevent misinterpretation or corruption, the Transmitter must do the following:
  - transmit all DWs of the TLP when the Physical Layer is using 128b/130b encoding (see Section 4.2.2.3.1)
  - use the remainder of the calculated LCRC value without inversion (the logical inverse of the value normally used)
  - indicate to the Transmit Physical Layer that the TLP is nullified
- When this is done, the Transmitter does not increment NEXT_TRANSMIT_SEQ

The following rules describe the operation of the Data Link Layer Retry Buffer, from which TLPs are re-transmitted when necessary:

- Copies of Transmitted TLPs must be stored in the Data Link Layer Retry Buffer, except for nullified TLPs.

When a replay is initiated, either due to reception of a Nak or due to REPLAY_TIMER expiration, the following rules describe the sequence of operations that must be followed:

- If all TLPs transmitted have been acknowledged (the Retry Buffer is empty), terminate replay, otherwise continue.
- Increment REPLAY_NUM. When the replay is initiated by the reception of a Nak which acknowledged some TLPs in the retry buffer, REPLAY_NUM is reset. It is then permitted (but not required) to be incremented.
  - If REPLAY_NUM rolls over from 11b to 00b, the Transmitter signals the Physical Layer to retrain the Link, and waits for the completion of retraining before proceeding with the replay. This is a reported error associated with the Port (see Section 6.2).

    Note that Data Link Layer state, including the contents of the Retry Buffer, are not reset by this action unless the Physical Layer reports Physical LinkUp = 0b (causing the Data Link Control and Management State Machine to transition to the DL_Inactive state).
  - If REPLAY_NUM does not roll over from 11b to 00b, continue with the replay.
- Block acceptance of new TLPs from the Transmit Transaction Layer.
- Complete transmission of any TLP currently being transmitted.
- Retransmit unacknowledged TLPs, starting with the oldest unacknowledged TLP and continuing in original transmission order
  - Reset and restart REPLAY_TIMER when sending the last Symbol of the first TLP to be retransmitted
  - Once all unacknowledged TLPs have been re-transmitted, return to normal operation.
  - If any Ack or Nak DLLPs are received during a replay, the Transmitter is permitted to complete the replay without regard to the Ack or Nak DLLP(s), or to skip retransmission of any newly acknowledged TLPs.
    - Once the Transmitter has started to resend a TLP, it must complete transmission of that TLP in all cases.

- Ack and Nak DLLPs received during a replay must be processed, and may be collapsed
  - Example: If multiple Acks are received, only the one specifying the latest Sequence Number value must be considered – Acks specifying earlier Sequence Number values are effectively "collapsed" into this one
  - Example: During a replay, Nak is received, followed by an Ack specifying a later Sequence Number – the Ack supersedes the Nak, and the Nak is ignored

  Note: Since all entries in the Retry Buffer have already been allocated space in the Receiver by the Transmitter's Flow Control gating logic, no further flow control synchronization is necessary.

- ❑ Re-enable acceptance of new TLPs from the Transmit Transaction Layer.

A replay can be initiated by the expiration of REPLAY_TIMER, or by the receipt of a Nak. The following rule covers the expiration of REPLAY_TIMER:

- ❑ If the Transmit Retry Buffer contains TLPs for which no Ack or Nak DLLP has been received, and (as indicated by REPLAY_TIMER) no Ack or Nak DLLP has been received for a period exceeding the REPLAY_TIMER Limit, the Transmitter initiates a replay.
  - Simplified REPLAY_TIMER Limits are:
    - A value from 24,000 to 31,000 Symbol Times when the Extended Synch bit is Clear.
    - A value from 80,000 to 100,000 Symbol Times when the Extended Synch bit is Set.
    - If the Extended Synch bit changes state while unacknowledged TLPs are outstanding, implementations are permitted to adjust their REPLAY_TIMER Limit when the Extended Synch bit changes state or the next time the REPLAY_TIMER is reset.
  - Implementations that support 16.0 GT/s must use the Simplified REPLAY_TIMER Limits for operation at all data rates.
  - Implementations that only support data rates less than 16.0 GT/s are strongly recommended to use the Simplified REPLAY_TIMER Limits for operation at all data rates, but they are permitted to use the REPLAY_TIMER Limits described in the *PCI Express Base Specification, Revision 3.1*.

This is a Replay Timer Timeout error and it is a reported error associated with the Port (see Section 6.2).

IMPLEMENTATION NOTE

Determining REPLAY_TIMER Limit Values

Replays are initiated primarily with a Nak DLLP, and the REPLAY_TIMER serves as a secondary mechanism. Since it is a secondary mechanism, the REPLAY_TIMER Limit has a relatively small effect on the average time required to convey a TLP across a Link. The Simplified REPLAY_TIMER Limits have been defined so that no adjustments are required for ASPM L0s, Retimers, or other items as in previous revisions of the *PCI Express Base Specification*.

TLP Transmitters and compliance tests must base replay timing as measured at the Port of the TLP Transmitter. Timing starts with either the last Symbol of a transmitted TLP, or else the last Symbol of a received Ack DLLP, whichever determines the oldest unacknowledged TLP. Timing ends with the First Symbol of TLP retransmission.

When measuring replay timing to the point when TLP retransmission begins, compliance tests must allow for any other TLP or DLLP transmission already in progress in that direction (thus preventing the TLP retransmission).

IMPLEMENTATION NOTE

Recommended Priority of Scheduled Transmissions

When multiple DLLPs of the same type are scheduled for transmission but have not yet been transmitted, it is possible in many cases to "collapse" them into a single DLLP. For example, if a scheduled Ack DLLP transmission is stalled waiting for another transmission to complete, and during this time another Ack is scheduled for transmission, it is only necessary to transmit the second Ack, since the information it provides will supersede the information in the first Ack.

In addition to any TLP from the Transaction Layer (or the Retry Buffer, if a replay is in progress), Multiple DLLPs of different types may be scheduled for transmission at the same time, and must be prioritized for transmission. The following list shows the preferred priority order for selecting information for transmission. Note that the priority of the NOP DLLP and the Vendor-Specific DLLP is not listed, as usage of these DLLPs is completely implementation specific, and there is no recommended priority. Note that this priority order is a guideline, and that in all cases a fairness mechanism is highly recommended to ensure that no type of traffic is blocked for an extended or indefinite period of time by any other type of traffic. Note that the Ack Latency Limit value and REPLAY_TIMER Limit specify requirements measured at the Port of the component, and the internal arbitration policy of the component must ensure that these externally measured requirements are met.

1) Completion of any transmission (TLP or DLLP) currently in progress (highest priority)
2) Nak DLLP transmissions
3) Ack DLLP transmissions scheduled for transmission as soon as possible due to:
   receipt of a duplicate TLP –OR–
   expiration of the Ack latency timer (see Section 3.6.3.1)
4) FC DLLP transmissions required to satisfy Section 2.6
5) Retry Buffer re-transmissions
6) TLPs from the Transaction Layer
7) FC DLLP transmissions other than those required to satisfy Section 2.6
8) All other DLLP transmissions (lowest priority)

3.6.2.2 *Handling of Received DLLPs*

Since Ack/Nak and Flow Control DLLPs affect TLPs flowing in the opposite direction across the Link, the TLP transmission mechanisms in the Data Link Layer are also responsible for Ack/Nak and Flow Control DLLPs received from the other component on the Link. These DLLPs are processed according to the following rules (see Figure 3-17):

- If the Physical Layer indicates a Receiver Error, discard any DLLP currently being received and free any storage allocated for the DLLP. Note that reporting such errors to software is done by the Physical Layer (and, therefore, are not reported by the Data Link Layer).
- For all received DLLPs, the CRC value is checked by:
  - applying the same algorithm used for calculation of transmitted DLLPs to the received DLLP, not including the 16-bit CRC field of the received DLLP
  - comparing the calculated result with the value in the CRC field of the received DLLP
    - if not equal, the DLLP is corrupt
  - A corrupt received DLLP is discarded. This is a Bad DLLP error and is a reported error associated with the Port (see Section 6.2).
- A received DLLP which is not corrupt, but which uses unsupported DLLP Type encodings is discarded without further action. This is not considered an error.
- Non-zero values in Reserved fields are ignored.
- Receivers must process all DLLPs received at the rate they are received

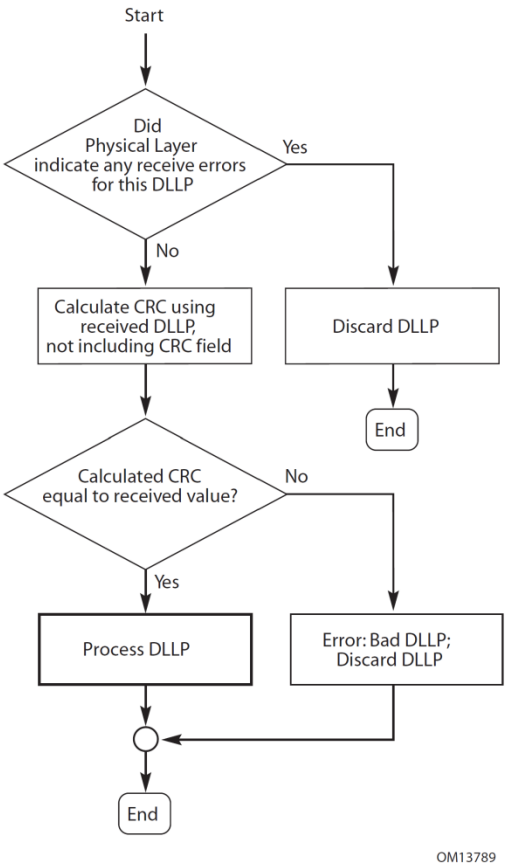


Figure 3-17: Received DLLP Error Check Flowchart

- Received NOP DLLPs are discarded
- Received FC DLLPs are passed to the Transaction Layer
- Received PM DLLPs are passed to the component's power management control logic
- For Ack and Nak DLLPs, the following steps are followed (see Figure 3-18):
  - If the Sequence Number specified by the AckNak_Seq_Num does not correspond to an unacknowledged TLP, or to the value in ACKD_SEQ, the DLLP is discarded
    - This is a Data Link Protocol Error which is a reported error associated with the Port (see Section 6.2).

      Note that it is not an error to receive an Ack DLLP when there are no outstanding unacknowledged TLPs, including the time between reset and the first TLP transmission, as long as the specified Sequence Number matches the value in ACKD_SEQ.
  - If the AckNak_Seq_Num does not specify the Sequence Number of the most recently acknowledged TLP, then the DLLP acknowledges some TLPs in the retry buffer:
    - Purge from the retry buffer all TLPs from the oldest to the one corresponding to the AckNak_Seq_Num

- Load ACKD_SEQ with the value in the AckNak_Seq_Num field
- Reset REPLAY_NUM and REPLAY_TIMER

o If the DLLP is a Nak, initiate a replay (see above)

Note: Receipt of a Nak is not a reported error.

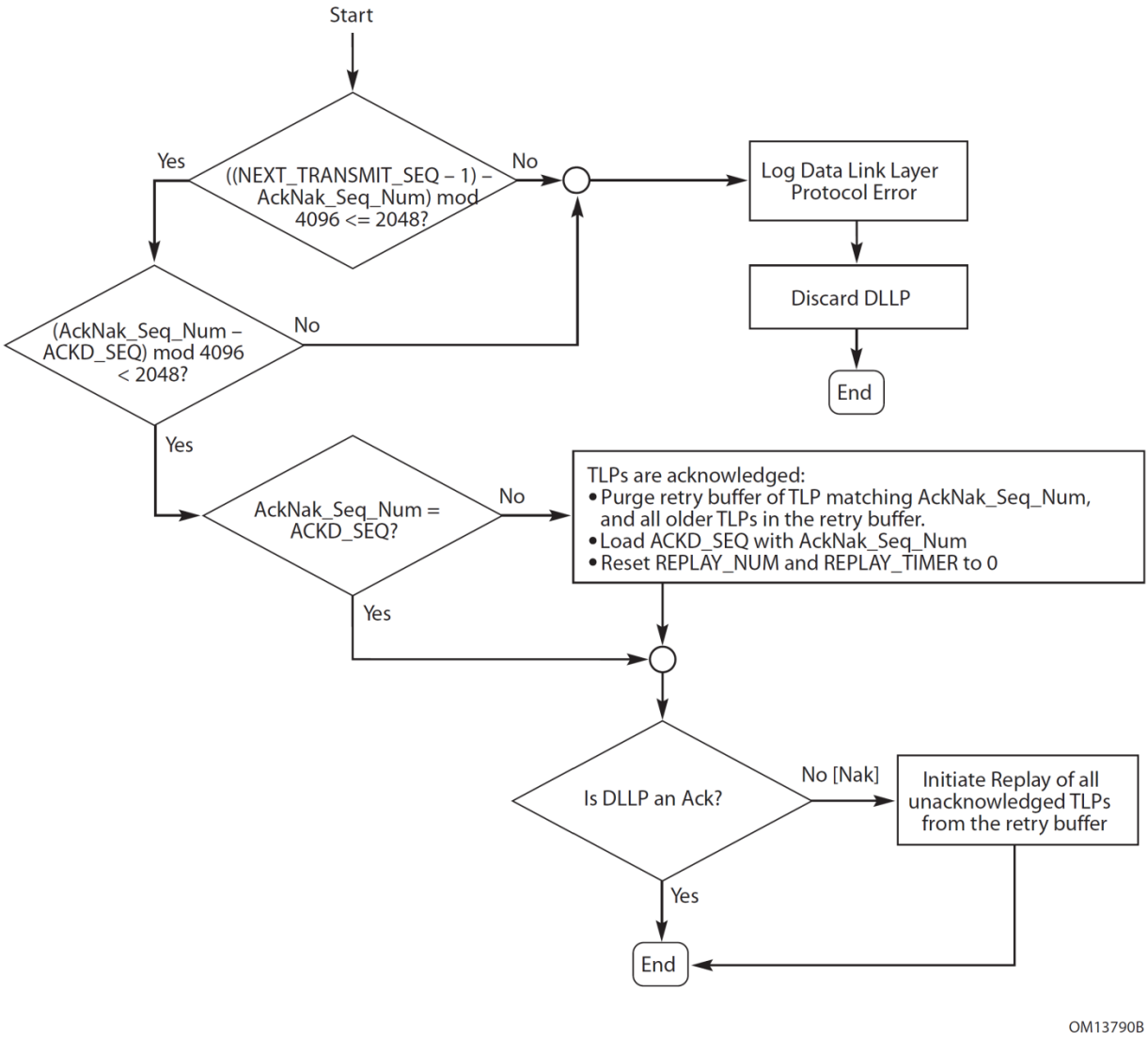


Figure 3-18: Ack/Nak DLLP Processing Flowchart

The following rules describe the operation of the Data Link Layer Retry Buffer, from which TLPs are re-transmitted when necessary:

❑ Copies of Transmitted TLPs must be stored in the Data Link Layer Retry Buffer

3.6.3 LCRC and Sequence Number (TLP Receiver)

The TLP Receive path through the Data Link Layer (paths labeled 2 and 4 in Figure 3-1) processes TLPs received by the Physical Layer by checking the LCRC and sequence number, passing the TLP to the Receive Transaction Layer if OK and requesting a replay if corrupted.

The mechanisms used to check the TLP LCRC and the Sequence Number and to support Data Link Layer Retry are described in terms of conceptual "counters" and "flags". This description does not imply or require a particular implementation and is used only to clarify the requirements.

3.6.3.1 *LCRC and Sequence Number Rules (TLP Receiver)*

The following counter, flag, and timer are used to explain the remaining rules in this section:

- The following 12-bit counter is used:
  - NEXT_RCV_SEQ – Stores the expected Sequence Number for the next TLP
    - Set to 000h in DL_Inactive state
- The following flag is used:
  - NAK_SCHEDULED
    - Cleared when in DL_Inactive state
- The following timer is used:
  - AckNak_LATENCY_TIMER – Counts time that determines when an Ack DLLP becomes scheduled for transmission, according to the following rules:
    - Set to 0 in DL_Inactive state
    - Restart from 0 each time an Ack or Nak DLLP is scheduled for transmission; Reset to 0 when all TLPs received have been acknowledged with an Ack DLLP
    - If there are initially no unacknowledged TLPs and a TLP is then received, the AckNak_LATENCY_TIMER starts counting only when the TLP has been forwarded to the Receive Transaction Layer

The following rules are applied in sequence to describe how received TLPs are processed, and what events trigger the transmission of Ack and Nak DLLPs (see Figure 3-19):

- If the Physical Layer indicates a Receiver Error, discard any TLP currently being received and free any storage allocated for the TLP. Note that reporting such errors to software is done by the Physical Layer (and so are not reported by the Data Link Layer).
  - If a TLP was being received at the time the Receiver Error was indicated and the NAK_SCHEDULED flag is clear,
    - schedule a Nak DLLP for transmission immediately
    - set the NAK_SCHEDULED flag
- If the Physical Layer reports that the received TLP was nullified, and the LCRC is the logical NOT of the calculated value, discard the TLP and free any storage allocated for the TLP. This is not considered an error.

- If TLP was nullified but the LCRC does not match the logical NOT of the calculated value, the TLP is corrupt - discard the TLP and free any storage allocated for the TLP.
  - If the NAK_SCHEDULED flag is clear,
    - schedule a Nak DLLP for transmission immediately
    - set the NAK_SCHEDULED flag

  This is a Bad TLP error and is a reported error associated with the Port (see Section 6.2).
- The LCRC value is checked by:
  - applying the same algorithm used for calculation (above) to the received TLP, not including the 32-bit LCRC field of the received TLP
  - comparing the calculated result with the value in the LCRC field of the received TLP
    - if not equal, the TLP is corrupt - discard the TLP and free any storage allocated for the TLP
      - If the NAK_SCHEDULED flag is clear,
        - schedule a Nak DLLP for transmission immediately
        - set the NAK_SCHEDULED flag

    This is a Bad TLP error and is a reported error associated with the Port (see Section 6.2).
- If the TLP Sequence Number is not equal to the expected value, stored in NEXT_RCV_SEQ:
  - discard the TLP and free any storage allocated for the TLP
  - If the TLP Sequence Number satisfies the following equation:

  (NEXT_RCV_SEQ - TLP Sequence Number) mod 4096 <= 2048

  the TLP is a duplicate, and an Ack DLLP is scheduled for transmission (per transmission priority rules)
  - Otherwise, the TLP is out of sequence (indicating one or more lost TLPs):
    - if the NAK_SCHEDULED flag is clear,
      - schedule a Nak DLLP for transmission immediately
      - set the NAK_SCHEDULED flag

      This is a Bad TLP error and is a reported error associated with the Port (see Section 6.2).

  Regardless of the state of the NAK_SCHEDULED flag, it is permitted for this to be a reported error associated with the Port (see Section 6.2), and this permitted behavior is illustrated in Figure 3-17. However, in order to prevent error pollution it is recommended that the Port only report such an error when the NAK_SCHEDULED flag is clear.

- If the TLP Sequence Number is equal to the expected value stored in NEXT_RCV_SEQ:
  - The four Reserved bits, TLP Sequence Number, and LCRC (see Figure 3-14) are removed and the remainder of the TLP is forwarded to the Receive Transaction Layer
    - The Data Link Layer indicates the start and end of the TLP to the Transaction Layer while transferring the TLP
      - The Data Link Layer treats the TLP as a "black box" and does not process or modify the contents of the TLP
    - Note that the Receiver Flow Control mechanisms do not account for any received TLPs until the TLP(s) are forwarded to the Receive Transaction Layer
  - NEXT_RCV_SEQ is incremented
  - If Set, the NAK_SCHEDULED flag is cleared

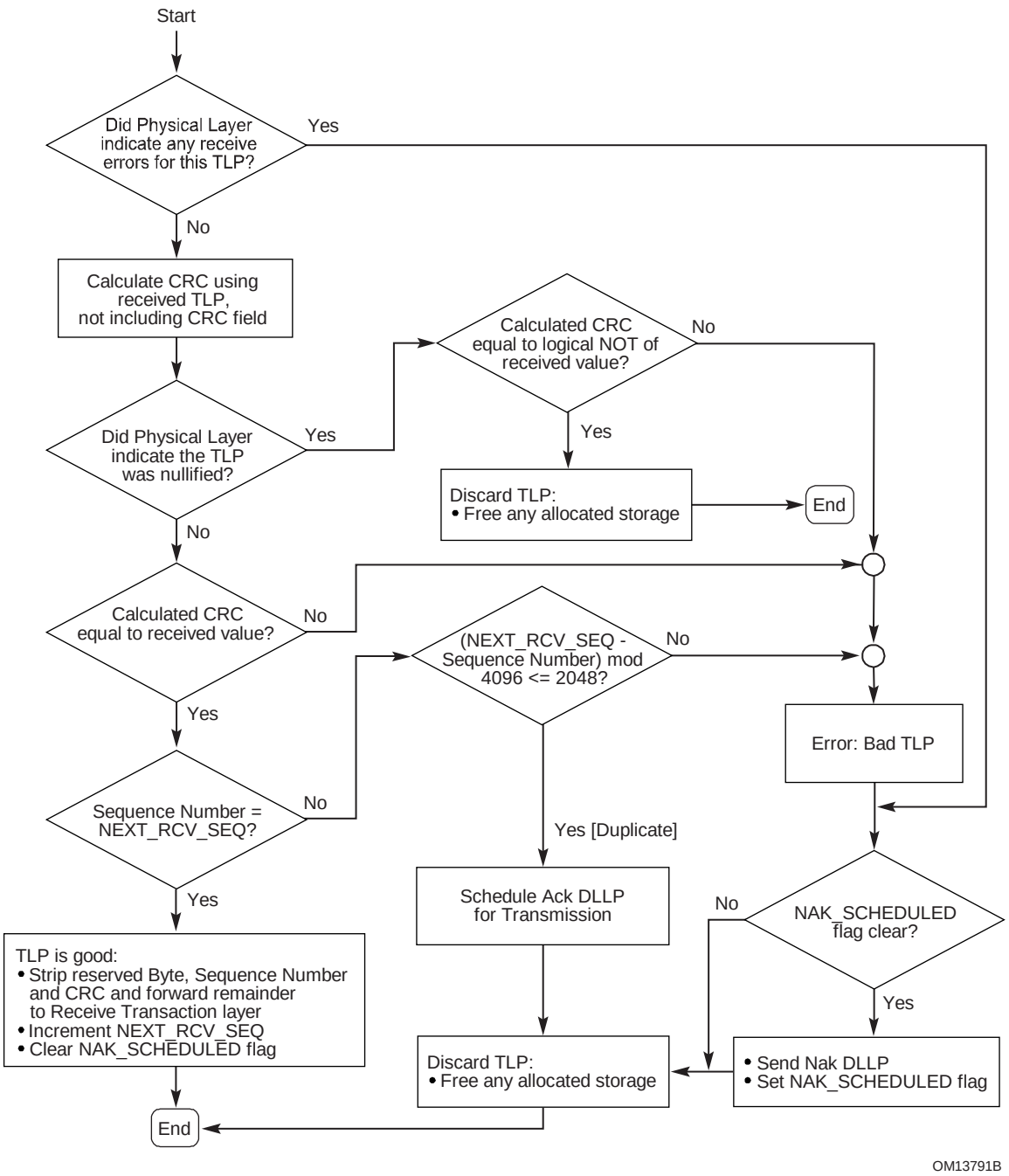


Figure 3-19: Receive Data Link Layer Handling of TLPs

- A TLP Receiver must schedule an Ack DLLP such that it will be transmitted no later than when all of the following conditions are true:
  - The Data Link Control and Management State Machine is in the DL_Active state
  - TLPs have been forwarded to the Receive Transaction Layer, but not yet acknowledged by sending an Ack DLLP
  - The AckNak_LATENCY_TIMER reaches or exceeds the value specified in Table 3-7 for 2.5 GT/s mode operation, Table 3-8 for 5.0 GT/s mode operation, Table 3-9 for 8.0 GT/s mode operation, and Table 3-10 for 16.0 GT/s mode operation
  - The Link used for Ack DLLP transmission is already in L0 or has transitioned to L0
  - Note: if not already in L0, the Link must transition to L0 in order to transmit the Ack DLLP
  - Another TLP or DLLP is not currently being transmitted on the Link used for Ack DLLP transmission
  - The NAK_SCHEDULED flag is clear
  - Note: The AckNak_LATENCY_TIMER must be restarted from 0 each time an Ack or Nak DLLP is scheduled for transmission
- Data Link Layer Ack DLLPs may be scheduled for transmission more frequently than required
- Data Link Layer Ack and Nak DLLPs specify the value (NEXT_RCV_SEQ - 1) in the AckNak_Seq_Num field

Table 3-7, Table 3-8, and Table 3-9, and Table 3-10 define the threshold values for the AckNak_LATENCY_TIMER, which for any specific case is called the Ack Latency Limit.

TLP Receivers and compliance tests must base Ack Latency timing as measured at the Port of the TLP Receiver, starting with the time the last Symbol of a TLP is received to the first Symbol of the Ack DLLP being transmitted.

When measuring until the Ack DLLP is transmitted, compliance tests must allow for any TLP or other DLLP transmission already in progress in that direction (thus preventing the Ack DLLP transmission). If L0s is enabled, compliance tests must allow for the L0s exit latency of the Link in the direction that the Ack DLLP is being transmitted. If the Extended Synch bit of the Link Control register is Set, compliance tests must also allow for its effect on L0s exit latency.

TLP Receivers are not required to adjust their Ack DLLP scheduling based upon L0s exit latency or the value of the Extended Synch bit.

Table 3-7: Maximum Ack Latency Limits 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 3-8: Maximum Ack Latency Limits 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

Table 3-9: Maximim Ack Latency Limits for 8.0 GT/s (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

Table 3-10: Maximum Ack Latency Limits for 16.0 GT/s (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

IMPLEMENTATION NOTE

Retry Buffer Sizing

The Retry Buffer should be large enough to ensure that under normal operating conditions, transmission is never throttled because the retry buffer is full. In determining the optimal buffer size, one must consider the Ack Latency value, Ack delay caused by the Receiver already transmitting another TLP, the delays caused by the physical Link interconnect, and the time required to process the received Ack DLLP.

Given two components A and B, the L0s exit latency required by A's Receiver should be accounted for when sizing A's transmit retry buffer, as is demonstrated in the following example:

- A exits L0s on its Transmit path to B and starts transmitting a long burst of write Requests to B
- B initiates L0s exit on its Transmit path to A, but the L0s exit time required by A's Receiver is large
- Meanwhile, B is unable to send Ack DLLPs to A, and A stalls due to lack of Retry Buffer space
- The Transmit path from B to A returns to L0, B transmits an Ack DLLP to A, and the stall is resolved

This stall can be avoided by matching the size of a component's Transmitter Retry Buffer to the L0s exit latency required by the component's Receiver, or, conversely, by matching the Receiver L0s exit latency to the desired size of the Retry Buffer.

Ack Latency Limit values were chosen to allow implementations to achieve good performance without requiring an uneconomically large retry buffer. To enable consistent performance across a general purpose interconnect with differing implementations and applications, it is necessary to set the same requirements for all components without regard to the application space of any specific component. If a component does not require the full transmission bandwidth of the Link, it may reduce the size of its retry buffer below the minimum size required to maintain available retry buffer space with the Ack Latency Limit values specified.

Note that the Ack Latency Limit values specified ensure that the range of permitted outstanding TLP Sequence Numbers will never be the limiting factor causing transmission stalls.

Retimers add latency (see Section 4.3.8) and operating in SRIS can add latency. Implementations are strongly encouraged to consider these effects when determining the optimal buffer size.

4

4 Physical Layer Logical Block

4.1 Introduction

The Physical Layer isolates the Transaction and Data Link Layers from the signaling technology used for Link data interchange. The Physical Layer is divided into the logical and electrical sub-blocks (see Figure 4-1).

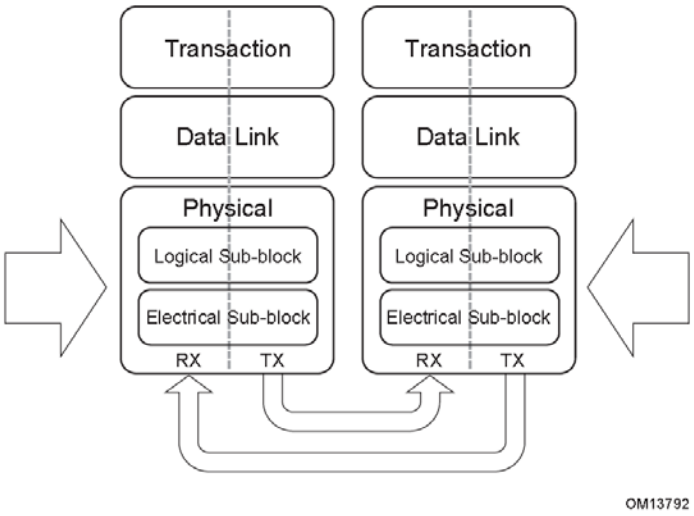


Figure 4-1: Layering Diagram Highlighting Physical Layer

Chapter 4 describes the logical sub-block and Chapter 9 describes the electrical sub-block.[44]

4.2 Logical Sub-block

The logical sub-block has two main sections: a Transmit section that prepares outgoing information passed from the Data Link Layer for transmission by the electrical sub-block, and a Receiver section that identifies and prepares received information before passing it to the Data Link Layer.

---

[44] Prior to the *PCI Express Base Specification, Revision 4.0*, Chapter 4 described both logical and electrical sub-blocks. With *PCI Express Base Specification, Revision 4.0* the following reorganization occurred:

- Section 4.1 was incorporated into the introductory material in Chapters 4 and 8.
- Section 4.2 became Chapter 4. Section numbers of form 4.2.*x* are now 4.*x*.
- Section 4.3 became Chapter 9. Section numbers of form 4.3.*y* are now 8.*y*.

The logical sub-block and electrical sub-block coordinate the state of each Transceiver through a status and control register interface or functional equivalent. The logical sub-block directs control and management functions of the Physical Layer.

PCI Express uses 8b/10b encoding when the data rate is 2.5 GT/s or 5.0 GT/s. For data rates greater than or equal to 8.0 GT/s, it uses a per-Lane code along with physical layer encapsulation.

4.2.1 Encoding for 2.5 GT/s and 5.0 GT/s Data Rates

4.2.1.1 *Symbol Encoding*

At 2.5 and 5.0 GT/s, PCI Express uses an 8b/10b transmission code. The definition of this transmission code is identical to that specified in ANSI X3.230-1994, clause 11 (and also IEEE 802.3z, 36.2.4). Using this scheme, 8-bit data characters are treated as 3 bits and 5 bits mapped onto a 4-bit code group and a 6-bit code group, respectively. The control bit in conjunction with the data character is used to identify when to encode one of the 12 Special Symbols included in the 8b/10b transmission code. These code groups are concatenated to form a 10-bit Symbol. As shown in Figure 4-2, ABCDE maps to abcdei and FGH maps to fghj.

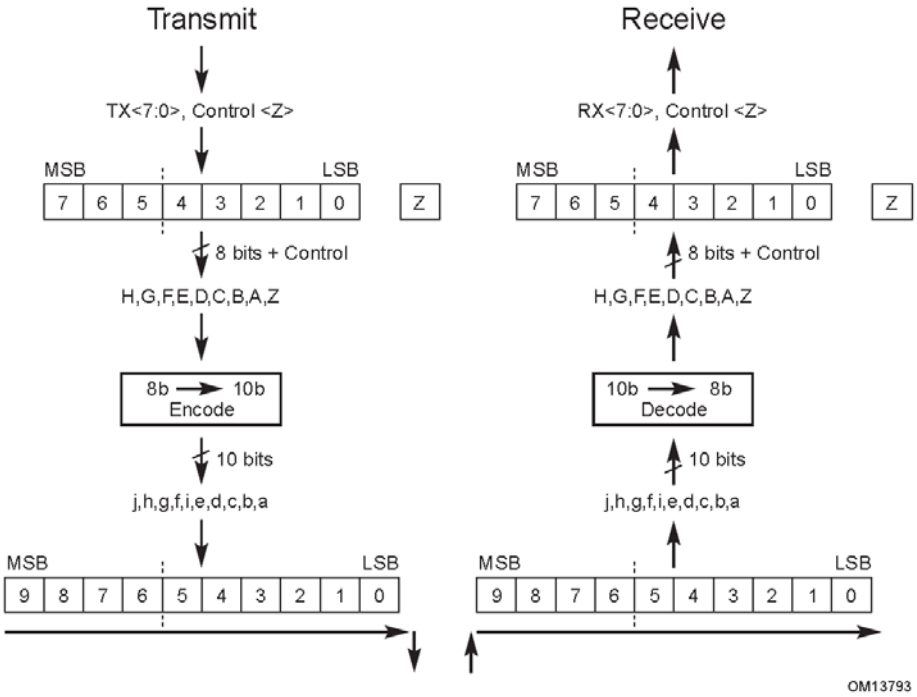


Figure 4-2: Character to Symbol Mapping

4.2.1.1.1 Serialization and De-serialization of Data

The bits of a Symbol are placed on a Lane starting with bit "a" and ending with bit "j". Examples are shown in Figure 4-3 and Figure 4-4.

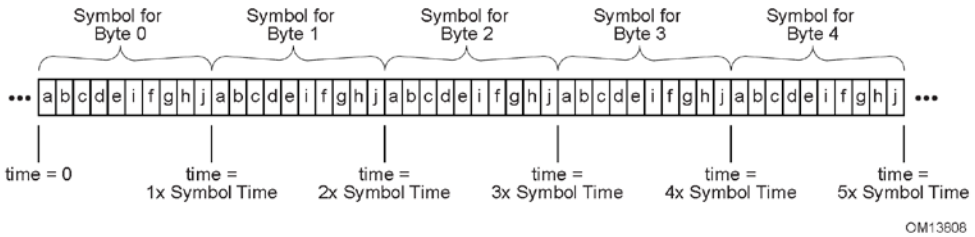


Figure 4-3: Bit Transmission Order on Physical Lanes - x1 Example

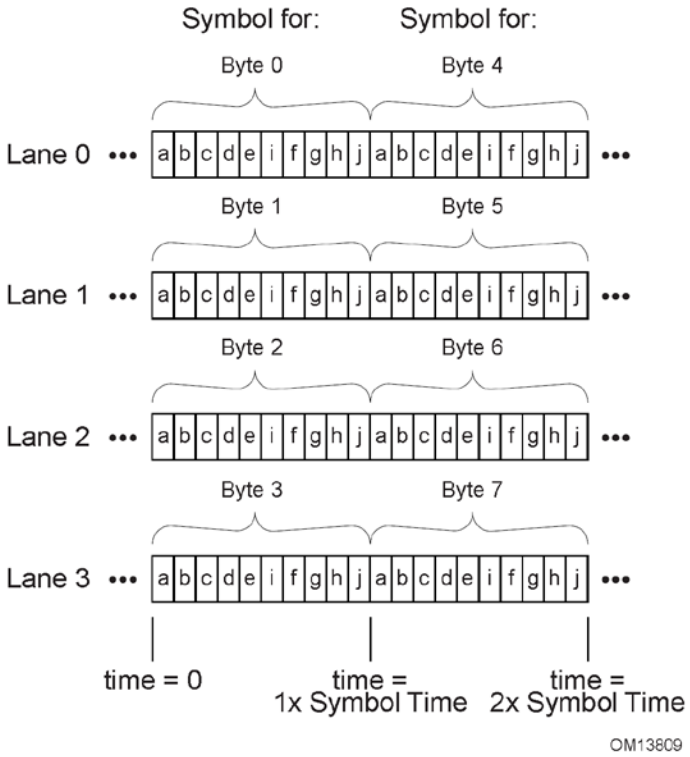


Figure 4-4: Bit Transmission Order on Physical Lanes - x4 Example

4.2.1.1.2 Special Symbols for Framing and Link Management (K Codes)

The 8b/10b encoding scheme provides Special Symbols that are distinct from the Data Symbols used to represent Characters. These Special Symbols are used for various Link Management mechanisms described later in this chapter. Special Symbols are also used to frame DLLPs and TLP[45]s, using distinct Special Symbols to allow these two types of Packets to be quickly and easily distinguished.

[45] In Chapter 4, PMUX packets follow the TLP framing rules.

Table 4-1 shows the Special Symbols used for PCI Express and provides a brief description for each. These Symbols will be discussed in greater detail in following sections. Each of these Special Symbols, as well as the data Symbols, must be interpreted by looking at the 10-bit Symbol in its entirety.

Table 4-1: Special Symbols

Encoding	Symbol	Name	Description
K28.5	COM	Comma	Used for Lane and Link initialization and management
K27.7	STP	Start TLP	Marks the start of a Transaction Layer Packet
K28.2	SDP	Start DLLP	Marks the start of a Data Link Layer Packet
K29.7	END	End	Marks the end of a Transaction Layer Packet or a Data Link Layer Packet
K30.7	EDB	EnD Bad	Marks the end of a nullified TLP
K23.7	PAD	Pad	Used in Framing and Link Width and Lane ordering negotiations
K28.0	SKP	Skip	Used for compensating for different bit rates for two communicating Ports
K28.1	FTS	Fast Training Sequence	Used within an Ordered Set to exit from L0s to L0
K28.3	IDL	Idle	Used in the Electrical Idle Ordered Set (EIOS)
K28.4			Reserved
K28.6			Reserved
K28.7	EIE	Electrical Idle Exit	Reserved in 2.5 GT/s Used in the Electrical Idle Exit Ordered Set (EIEOS) and sent prior to sending FTS at data rates other than 2.5 GT/s

4.2.1.1.3 8b/10b Decode Rules

The Symbol tables for the valid 8b/10b codes are given in Appendix B. These tables have one column for the positive disparity and one column for the negative disparity.

A Transmitter is permitted to pick any disparity, unless otherwise required, when first transmitting differential data after being in an Electrical Idle state. The Transmitter must then follow proper 8b/10b encoding rules until the next Electrical Idle state is entered.

The initial disparity for a Receiver that detects an exit from Electrical Idle is set to the disparity of the first Symbol used to obtain Symbol lock. Disparity may also be reinitialized if Symbol lock is lost and regained during the transmission of differential information due to an implementation

specific number of errors. All following received Symbols after the initial disparity is set must be found in the proper column corresponding to the current running disparity.

If a received Symbol is found in the column corresponding to the incorrect running disparity or if the Symbol does not correspond to either column, the Physical Layer must notify the Data Link Layer that the received Symbol is invalid. This is a Receiver Error, and is a reported error associated with the Port (see Section 6.2).

4.2.1.2 *Framing and Application of Symbols to Lanes*

There are two classes of framing and application of Symbols to Lanes. The first class consists of the Ordered Sets and the second class consists of TLPs and DLLPs. Ordered Sets are always transmitted serially on each Lane, such that a full Ordered Set appears simultaneously on all Lanes of a multi-Lane Link.

The Framing mechanism uses Special Symbol K28.2 "SDP" to start a DLLP and Special Symbol K27.7 "STP" to start a TLP. The Special Symbol K29.7 "END" is used to mark the end of either a TLP or a DLLP.

The conceptual stream of Symbols must be mapped from its internal representation, which is implementation dependent, onto the external Lanes. The Symbols are mapped onto the Lanes such that the first Symbol (representing Character 0) is placed onto Lane 0; the second is placed onto Lane 1; etc. The x1 Link represents a degenerate case and the mapping is trivial, with all Symbols placed onto the single Lane in order.

When no packet information or special Ordered Sets are being transmitted, the Transmitter is in the Logical Idle state. During this time idle data must be transmitted. The idle data must consist of the data byte 0 (00 Hexadecimal), scrambled according to the rules of Section 4.2.1.3 and 8b/10b encoded according to the rules of Section 4.2.1.1, in the same way that TLP and DLLP Data Symbols are scrambled and encoded. Likewise, when the Receiver is not receiving any packet information or special Ordered Sets, the Receiver is in Logical Idle and shall receive idle data as described above. During transmission of the idle data, the SKP Ordered Set must continue to be transmitted as specified in Section 4.2.7.

For the following rules, "placed" is defined to mean a requirement on the Transmitter to put the Symbol into the proper Lane of a Link.

- TLPs must be framed by placing an STP Symbol at the start of the TLP and an END Symbol or EDB Symbol at the end of the TLP (see Figure 4-5).
- A properly formed TLP contains a minimum of 18 symbols between the STP and END or EDB Symbols. If a received sequence has less than 18 symbols between the STP and END or EDB symbols, the receiver is permitted to treat this as a Receiver Error.
    - If checked, this is a reported error associated with the Receiving Port (see Section 6.2).
- DLLPs must be framed by placing an SDP Symbol at the start of the DLLP and an END Symbol at the end of the DLLP (see Figure 4-6).
- Logical Idle is defined to be a period of one or more Symbol Times when no information: TLPs, DLLPs or any type of Special Symbol is being Transmitted/Received. Unlike Electrical Idle, during Logical Idle the Idle Symbol (00h) is being transmitted and received.

- When the Transmitter is in Logical Idle, the Logical Idle data (00h) shall be transmitted on all Lanes. This is scrambled according to the rules in Section 4.2.1.3.
- Receivers must ignore incoming Logical Idle data, and must not have any dependency other than scramble sequencing on any specific data patterns.

- For Links wider than x1, the STP Symbol (representing the start of a TLP) must be placed in Lane 0 when starting Transmission of a TLP from a Logical Idle Link condition.
- For Links wider than x1, the SDP Symbol (representing the start of a DLLP) must be placed in Lane 0 when starting Transmission of a DLLP from a Logical Idle Link condition.
- The STP Symbol must not be placed on the Link more frequently than once per Symbol Time.
- The SDP Symbol must not be placed on the Link more frequently than once per Symbol Time.
- As long as the above rules are satisfied, TLP and DLLP Transmissions are permitted to follow each other successively.
- One STP Symbol and one SDP Symbol may be placed on the Link in the same Symbol Time.
  - Links wider than x4 can have STP and SDP Symbols placed in Lane 4*N, where N is a positive integer. For example, for x8, STP and SDP Symbols can be placed in Lanes 0 and 4; and for x16, STP and SDP Symbols can be placed in Lanes 0, 4, 8, or 12.
- For xN Links where N is 8 or more, if an END or EDB Symbol is placed in a Lane K, where K does not equal N-1, and is not followed by a STP or SDP Symbol in Lane K+1 (i.e., there is no TLP or DLLP immediately following), then PAD Symbols must be placed in Lanes K+1 to Lane N-1.
  - For example, on a x8 Link, if END or EDB is placed in Lane 3, PAD must be placed in Lanes 4 to 7, when not followed by STP or SDP.
- The EDB Symbol is used to mark the end of a nullified TLP. Refer to Section 3.6.2.1 for information on the usage of EDB.
- Receivers may optionally check for violations of the rules of this section. These checks are independently optional (see Section 6.2.3.4). If checked, violations are Receiver Errors, and are reported errors associated with the Port (see Section 6.2).

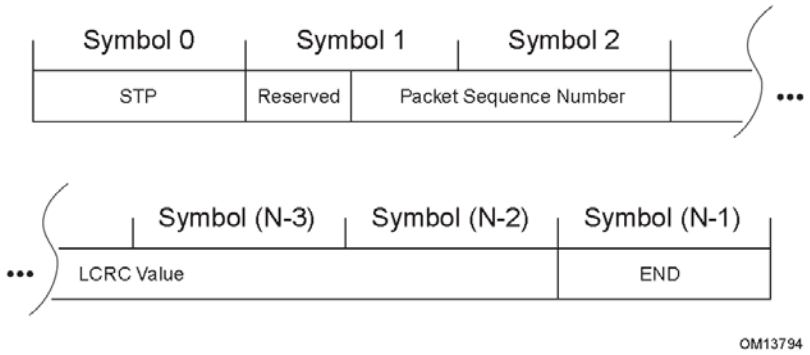


Figure 4-5: TLP with Framing Symbols Applied

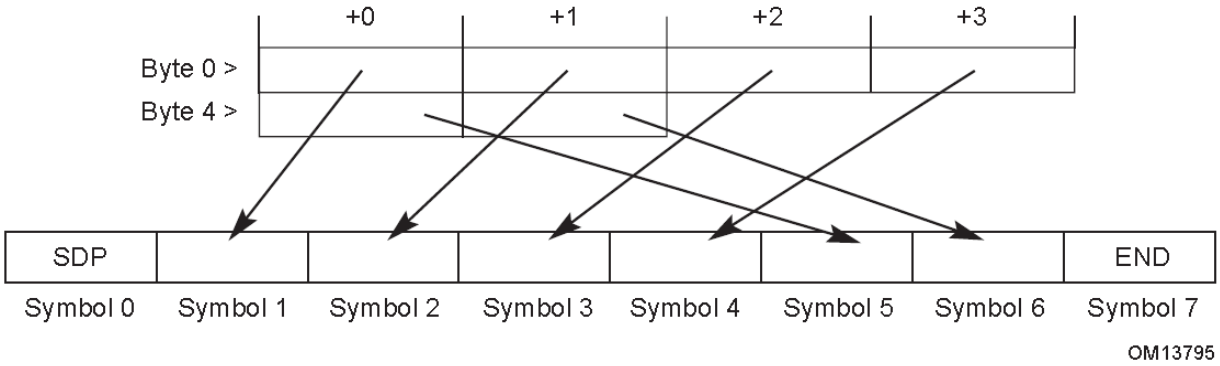


Figure 4-6: DLLP with Framing Symbols Applied

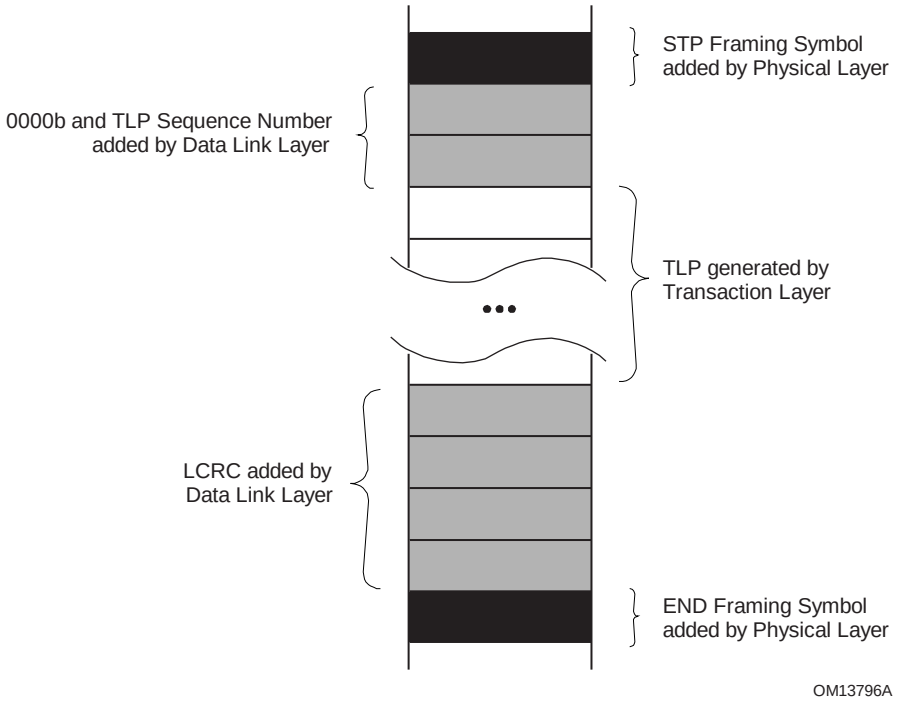


Figure 4-7: Framed TLP on a x1 Link

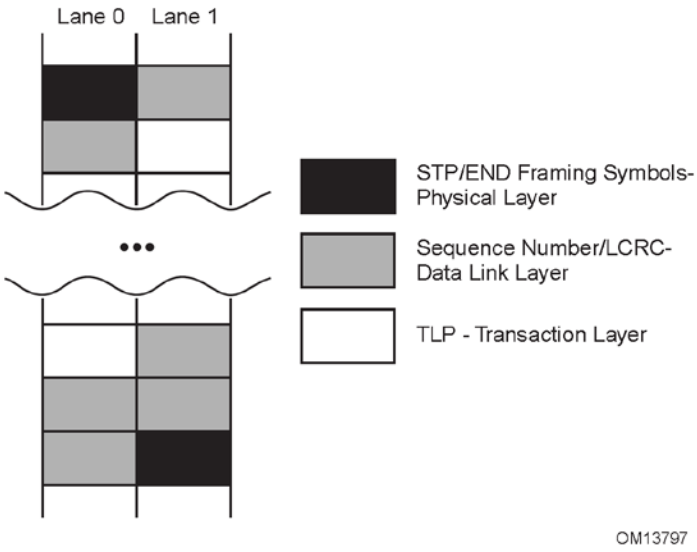


Figure 4-8: Framed TLP on a x2 Link

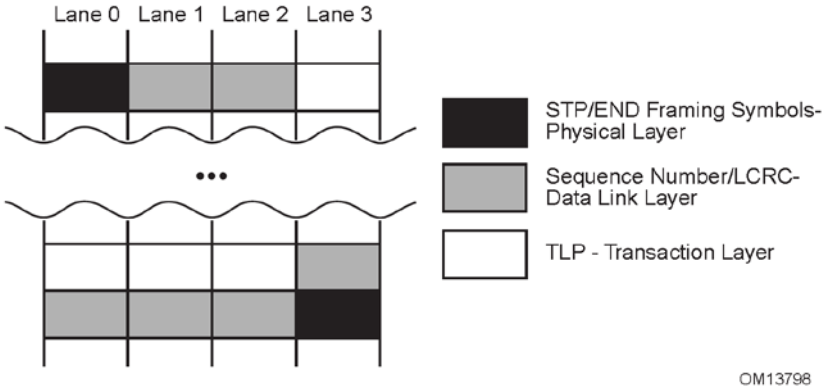


Figure 4-9: Framed TLP on a x4 Link

4.2.1.3 *Data Scrambling*

In order to improve electrical characteristics of a Link, data is typically scrambled. This involves XORing the data stream with a pattern generated by a Linear Feedback Shift Register (LFSR). On the Transmit side, scrambling is applied to characters prior to the 8b/10b encoding. On the Receive side, de-scrambling is applied to characters after 8b/10b decoding.

On a multi-Lane Link, the scrambling function can be implemented with one or many LFSRs. When there is more than one Transmit LFSR per Link, these must operate in concert, maintaining the same simultaneous (Lane-to-Lane Output Skew) value in each LFSR. When there is more than one Receive LFSR per Link, these must operate in concert, maintaining the same simultaneous (Lane-to-Lane Skew) value in each LFSR. Regardless of how they are implemented, LFSRs must

interact with data on a Lane-by-Lane basis as if there was a separate LFSR as described here for each Lane within that Link.

The LFSR is graphically represented in Figure 4-10. Scrambling or unscrambling is performed by serially XORing the 8-bit (D0-D7) character with the 16-bit (D0-D15) output of the LFSR. An output of the LFSR, D15, is XORed with D0 of the data to be processed. The LFSR and data register are then serially advanced and the output processing is repeated for D1 through D7. The LFSR is advanced after the data is XORed. The LFSR implements the polynomial:

$G(X)=X^{16}+X^{5}+X^{4}+X^{3}+1$

The mechanism(s) and/or interface(s) utilized by the Data Link Layer to notify the Physical Layer to disable scrambling is implementation specific and beyond the scope of this specification.

The data scrambling rules are the following:

- The COM Symbol initializes the LFSR.
- The LFSR value is advanced eight serial shifts for each Symbol except the SKP.
- All data Symbols (D codes) except those within Ordered Sets (e.g., TS1, TS2, EIEOS), the Compliance Pattern (see Section 4.2.8), and the Modified Compliance Pattern (see Section 4.2.9) are scrambled.
- All special Symbols (K codes) are not scrambled.
- The initialized value of an LFSR seed (D0-D15) is FFFFh. Immediately after a COM exits the Transmit LFSR, the LFSR on the Transmit side is initialized. Every time a COM enters the Receive LFSR on any Lane of that Link, the LFSR on the Receive side is initialized.
- Scrambling can only be disabled at the end of Configuration (see Section 4.2.6.3.5).
- Scrambling does not apply to a loopback slave.
- Scrambling is always enabled in Detect by default.

IMPLEMENTATION NOTE

Disabling Scrambling

Disabling scrambling is intended to help simplify test and debug equipment. Control of the exact data patterns is useful in a test and debug environment. Since scrambling is reset at the Physical Layer there is no reasonable way to reliably control the state of the data transitions through software. Thus, the Disable Scrambling bit in the TS1 and TS2 Ordered Sets is provided for these purposes.

The mechanism(s) and/or interface(s) utilized by the Data Link Layer to notify the Physical Layer to disable scrambling is implementation specific and beyond the scope of this specification.

For more information on scrambling, see Appendix C.

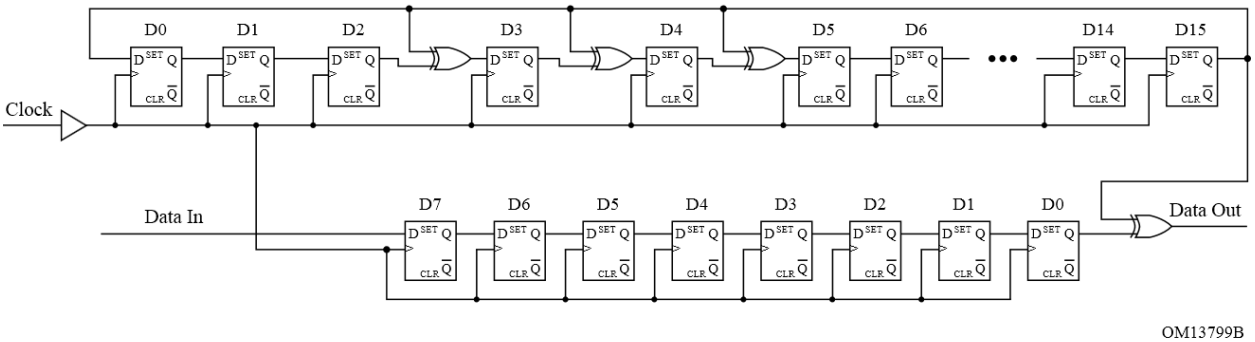


Figure 4-10: LFSR with 8b/10b Scrambling Polynomial

4.2.2 Encoding for 8.0 GT/s and Higher Data Rates

When a PCI Express Link is operating at a data rate of 8.0 GT/s or higher, it uses the encoding rules described in this subsection: 128b/130b encoding. For backwards compatibility, the Link initially trains to L0 at the 2.5 GT/s data rate using 8b/10b encoding as described in Section 4.2.1, then when the data rate is changed to 8.0 GT/s or higher, 128b/130b encoding is used. 128b/130b encoding is a Link-wide packetization mechanism and a per-Lane block code with scrambling. The basic entity of data transmission is an 8-bit data character, referred to as a Symbol, as shown in Figure 4-11 and Figure 4-12.

IMPLEMENTATION NOTE

Symbol in 128b/130b Encoding Scheme

In the 128b/130b encoding scheme, the Symbol is one byte long, similar to the 10-bit Symbol of 8b/10b encoding.

4.2.2.1 *Lane Level Encoding*

The physical layer uses a per-Lane block code. Each Block consists of a 2-bit Sync Header and a payload. There are two valid Sync Header encodings: 10b and 01b. The Sync Header defines the type of payload that the Block contains.

A Sync Header of 10b indicates a Data Block. Each Data Block has a 128 bit payload, resulting in a Block size of 130 bits. The payload is a Data Stream described in Section 4.2.2.3.

A Sync Header of 01b indicates an Ordered Set Block. Each Ordered Set Block has a 128 bit payload, resulting is a Block size of 130 bits except for the SKP Ordered Set which can be of variable length.

All Lanes of a multi-Lane Link must transmit Blocks with the same Sync Header simultaneously, except when transmitting Jitter Measurement Pattern in Polling.Compliance.

The bit transmission order is as follows. A Sync Header represented as 'H_1H_0' is placed on a Lane starting with 'H_0' and ending with 'H_1'. A Symbol, represented as '$S_7S_6S_5S_4S_3S_2S_1S_0$', is placed on a Lane starting with 'S_0' and ending with 'S_7'. In the diagrams that show a time scale, bits represent the

transmission order. In layout diagrams, bits are arranged in little-endian format, consistent with packet layout diagrams in other chapters of this specification.

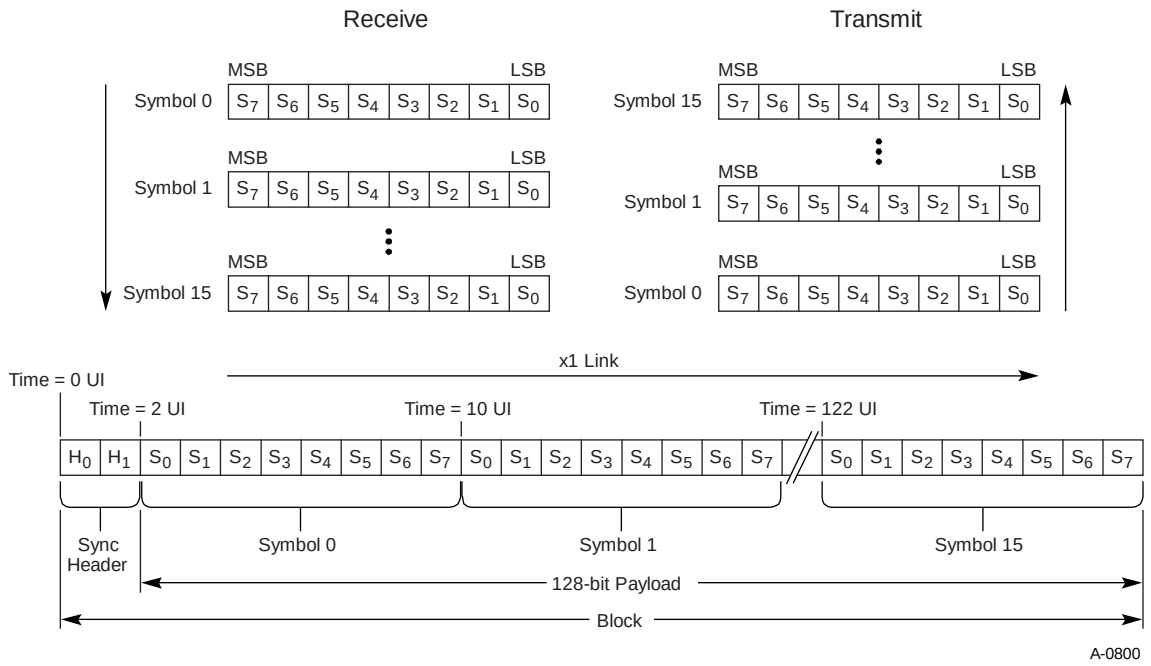


Figure 4-11: Example of Bit Transmission Order in a x1 Link Showing 130 Bits of a Block

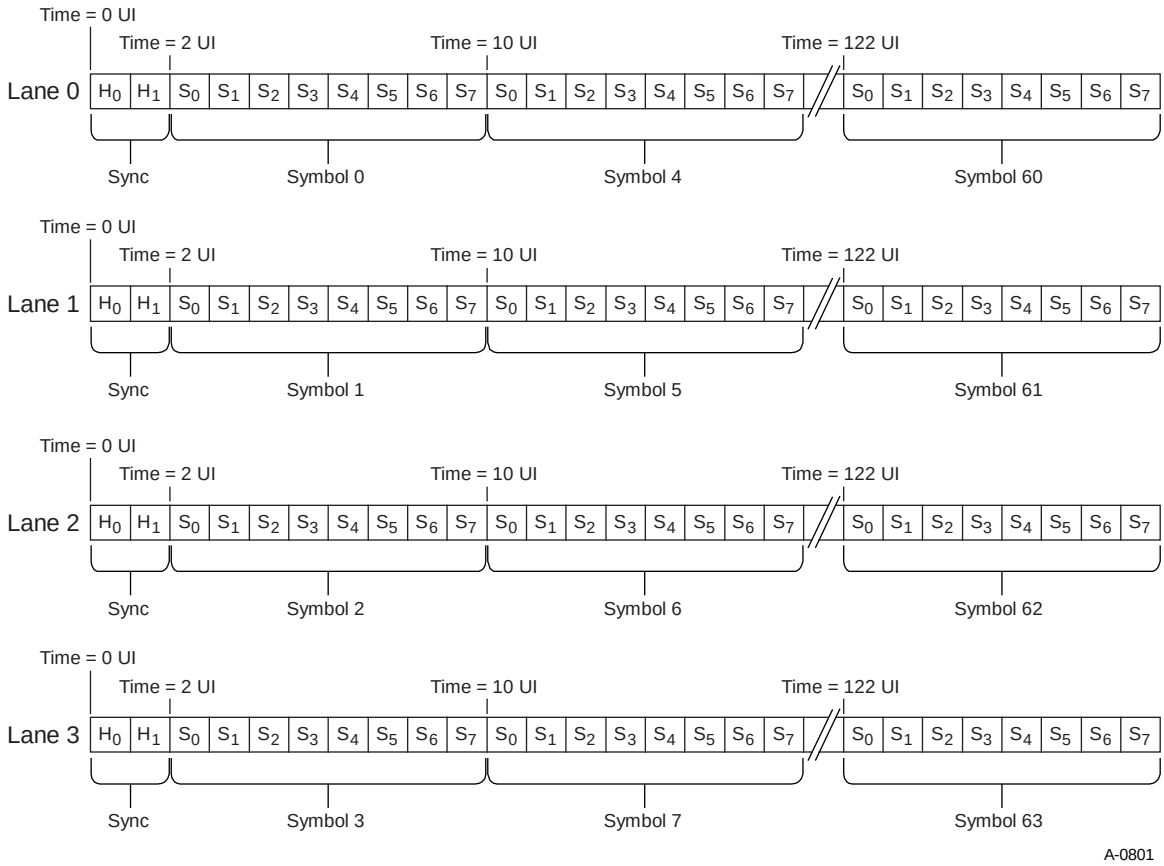


Figure 4-12: Example of Bit Placement in a x4 Link with One Block per Lane

4.2.2.2 *Ordered Set Blocks*

An Ordered Set Block contains a Sync Header followed by one Ordered Set. All Lanes of a multi-Lane Link must transmit the same Ordered Set type simultaneously. The first Symbol of the Ordered Set defines the type of Ordered Set. Subsequent symbols of the Ordered Set are defined by the Ordered Set type and need not be identical across lanes of a multi-Lane Link. The Ordered Sets are described in detail in Section 4.2.4 and Section 4.2.7.

4.2.2.2.1 Block Alignment

During Link training, the 130 bits of the Electrical Idle Exit Ordered Set (EIEOS) are a unique bit pattern that Receivers use to determine the location of the Block Sync Headers in the received bit stream. Conceptually, Receivers can be in three different phases of Block alignment: Unaligned, Aligned, and Locked. These phases are defined to illustrate the required behavior, but are not meant to specify a required implementation.

Unaligned Phase: Receivers enter this phase after a period of Electrical Idle, such as when the data rate is changed to one that uses 128b/130b encoding or when they exit a low-power Link state, or if directed. In this phase, Receivers monitor the received bit stream for the EIEOS bit pattern. When one is detected, they adjust their alignment to it and proceed to the Aligned phase.

Aligned Phase: Receivers monitor the received bit stream for the EIEOS bit pattern and the received Blocks for a Start of Data Stream (SDS) Ordered Set. If an EIEOS bit pattern is detected on an alignment that does not match the current alignment, Receivers must adjust their alignment to the newly received EIEOS bit pattern. If an SDS Ordered Set is received, Receivers proceed to the Locked phase. Receivers are permitted to return to the Unaligned phase if an undefined Sync Header (00b or 11b) is received.

Locked Phase: Receivers must not adjust their Block alignment while in this phase. Data Blocks are expected to be received after an SDS Ordered Set, and adjusting the Block alignment would interfere with the processing of these Blocks. Receivers must return to the Unaligned or Aligned phase if an undefined Sync Header is received.

IMPLEMENTATION NOTE

Detection of Loss of Block Alignment

The sequence of EIEOS and TS Ordered Sets transmitted during training sequences will cause misaligned Receivers to detect an undefined Sync Header.

Additional Requirements:

- While in the Aligned or Locked phase, Receivers must adjust their alignment as necessary when a SKP Ordered Set is received. See Section 4.2.7 for more information on SKP Ordered Sets.
- After any LTSSM transition to Recovery, Receivers must ignore all received TS Ordered Sets until they receive an EIEOS. Conceptually, receiving an EIEOS validates the Receiver's alignment and allows TS Ordered Set processing to proceed. If a received EIEOS initiates an LTSSM transition from L0 to Recovery, Receivers are permitted to process any TS Ordered Sets that follow the EIEOS or ignore them until another EIEOS is received after entering Recovery.
- Receivers are permitted to be directed from the Locked phase to the Unaligned or Aligned phase as long as Data Stream processing is stopped. See Section 4.2.2.3 for more information on Data Stream requirements.
- Loopback Masters: While in Loopback.Entry, Masters must be capable of adjusting their Receiver's Block alignment to received EIEOS bit patterns. While in Loopback.Active, Masters are permitted to transmit an EIEOS and adjust their Receiver's Block alignment to the looped back bit stream.
- Loopback Slaves: While in Loopback.Entry, Slaves must be capable of adjusting their Receiver's Block alignment to received EIEOS bit patterns. While in Loopback.Active, Slaves must not adjust their Receiver's Block alignment. Conceptually, the Receiver is directed to the Locked phase when the Slave starts to loop back the received bit stream.

4.2.2.3 *Data Blocks*

The payload of Data Blocks is a stream of Symbols defined as a "Data Stream" that consists of Framing Tokens, TLPs, and DLLPs. Each Symbol of the Data Stream is placed on a single Lane of the Link, and the stream of Symbols is striped across all Lanes of the Link and spans Block boundaries.

A Data Stream starts with the first Symbol of the Data Block that follows an SDS Ordered Set. It ends either when a Framing Error is detected or with the last Symbol of the Data Block that precedes an Ordered Set other than a SKP Ordered Set. SKP Ordered Sets that occur within a Data Stream have specific requirements as described in the following sections.

4.2.2.3.1 Framing Tokens

The Framing Tokens used by the Physical Layer are shown in Table 4-2. Each Framing Token specifies or implies the number of Symbols associated with the Token and therefore the location of the next Framing Token. Figure 4-15 shows an example of TLPs, DLLPs, and IDLs transmitted on a x8 link.

The first Framing Token of a Data Stream is always located in Symbol 0 of Lane 0 of the first Data Block of the Data Stream. For the rest of this chapter, the terms Framing Token and Token are used interchangeably.

Table 4-2: Framing Token Encoding

Framing Token Type	Description
IDL	Logical Idle. The Framing Token is 1 Symbol. This Token is transmitted when no TLPs or DLLPs or other Framing Tokens are being transmitted.
SDP	Start of DLLP. The Framing Token is 2 Symbols long and is followed by the DLLP information.
STP	Start of TLP. The Framing Token is 4 Symbols long and includes the 12-bit TLP Sequence Number. It is followed by the TLP information.
EDB	EnD Bad. The Framing Token is 4 Symbols long and is used to confirm that the previous TLP was nullified.
EDS	End of Data Stream. The Framing Token is four Symbols long and indicates that the next Block will be an Ordered Set Block.

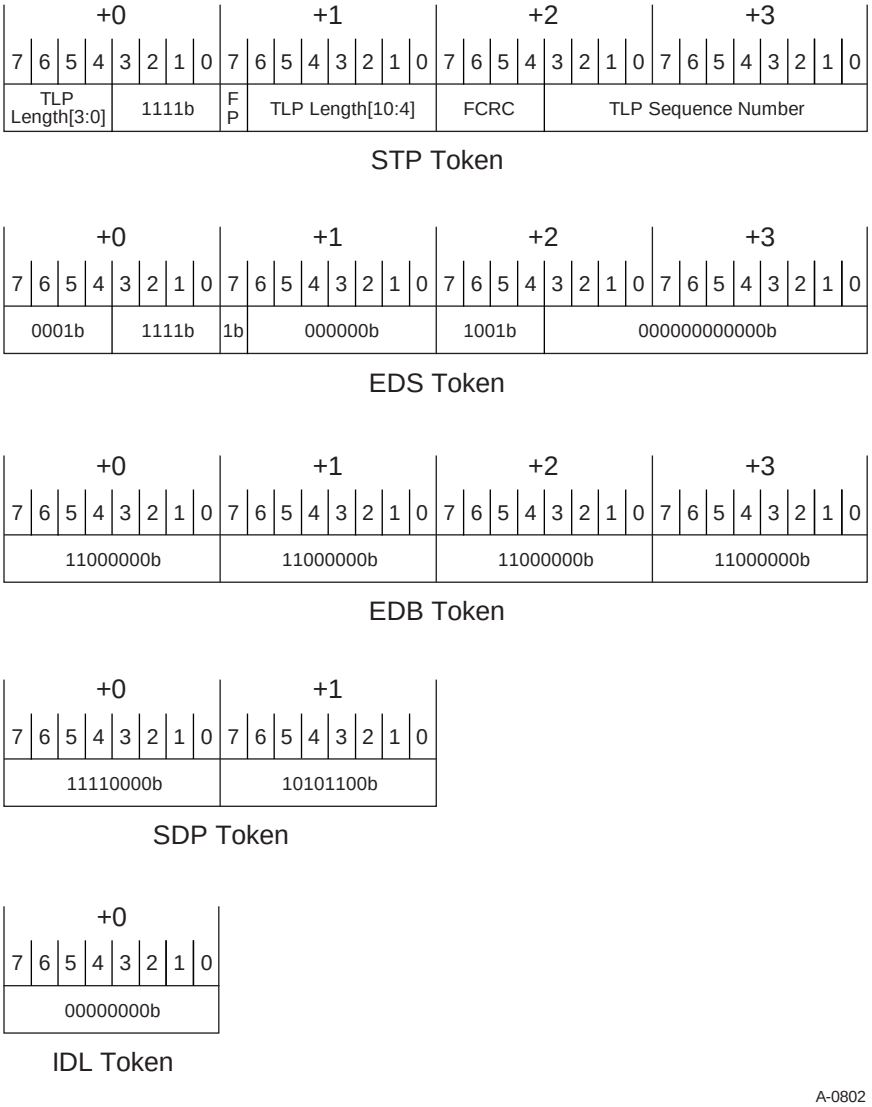


Figure 4-13: Layout of Framing Tokens

The Physical Layer DLLP layout is shown in Figure 4-14. Symbols 0 and 1 are the SDP Token, and Symbols 2 through 7 are the Data Link Layer DLLP information.

The Physical Layer TLP layout is shown in Figure 4-14. Details of the STP Framing Token are shown in Figure 4-13. The length of the TLP (in DWs) being transmitted is specified by an 11-bit field called TLP Length. The TLP Length field is the total amount of information transferred, including the Framing Token, TLP Prefixes (if any), TLP Header, TLP data payload (if any), TLP digest (if any), and TLP LCRC. For example, if a TLP has a 3 DW header, a 1 DW data payload, and does not include a TLP digest, the TLP Length field value is 6: 1 (Framing Token) + 0 (TLP Prefixes) + 3 (TLP header) + 1 (TLP data payload) + 0 (TLP digest) + 1 (TLP LCRC). If the same TLP included a TLP digest, the TLP Length field value would be 7. When a TLP is nullified, the EDB Token is considered an extension of the TLP but is not included in the calculation of the TLP Length field.

The TLP Length field is protected by a 4-bit CRC (Frame CRC), and an even parity bit (Frame Parity) protects both the TLP Length and Frame CRC fields. The Frame CRC and Frame Parity are calculated as follows:

C[0] = L[10] ^ L[7] ^ L[6] ^ L[4] ^ L[2] ^ L[1] ^ L[0]

C[1] = L[10] ^ L[9] ^ L[7] ^ L[5] ^ L[4] ^ L[3] ^ L[2]

C[2] = L[9] ^ L[8] ^ L[6] ^ L[4] ^ L[3] ^ L[2] ^ L[1]

C[3] = L[8] ^ L[7] ^ L[5] ^ L[3] ^ L[2] ^ L[1] ^ L[0]

P = L[10] ^ L[9] ^ L[8] ^ L[7] ^ L[6] ^ L[5] ^ L[4] ^ L[3] ^ L[2] ^ L[1] ^ L[0] ^ C[3] ^ C[2] ^ C[1] ^ C[0]

The Frame Parity reduces to P = L[10] ^ L[9] ^ L[8] ^ L[6] ^ L[5] ^ L[2] ^ L[0]

The TLP Length field is represented in the above equations as L[10:0], where L[0] is the least significant bit and L[10] is the most significant bit. Transmitters calculate the Frame CRC and Frame Parity before transmission. Receivers must calculate the Frame CRC and Frame Parity using the same algorithm as the transmitter and then compare the calculated values to the received values.

STP Tokens do not have a TLP Length field value of 1. If a received sequence of Symbols matches the format of an STP Token with a TLP Length field value of 1, the Symbols are evaluated to determine whether they match the EDS Token.

IMPLEMENTATION NOTE

Frame CRC and Frame Parity

The Frame CRC bits are effectively calculated as $(L[0]\ X^{14} + L[1]\ X^{13} + \ldots + L[9]\ X^{5} + L[10]\ X^{4}) \bmod (X^{4} + X + 1)$. It should be noted that $X^{4} + X + 1$ is a primitive polynomial and the CRC can detect two bit errors. The Frame Parity bit can detect an odd number of bit errors. Thus, the Frame CRC and Frame Parity together guarantee three bit error detection for the TLP Length field. It must be noted that even though in the reduced Frame Parity equation all terms are not present, it still maintains the property of detecting odd bit errors. Only those TLP Length field bits which are present in an even number of CRC terms are used in the calculation.

Note that, for TLPs, the Data Link Layer prepends 4 Reserved bits (0000b) to the TLP Sequence Number field before it calculates the LCRC. These Reserved bits are not explicitly transmitted when using 128b/130b encoding, and Receivers assume that the 4 bits received are 0000b when calculating the LCRC.

Packets containing a TLP Length field that is greater than 1535 are PMUX Packets. For such packets, the actual packet length is computed differently, the TLP Sequence Number field in the STP Token contains other information, and the Link CRC is computed using different rules. See Appendix G for details.

Packets containing a TLP Length field that is between 1152 and 1535 (inclusive) are reserved for future standardization.

Transmitters must transmit all DWs of a TLP specified by the TLP Length field of the STP Framing Token. TLPs are never truncated when using 128b/130b encoding - even when nullified. Figure 4-16 shows an example of a nullified 23 DW TLP.

Figure 4-17 shows an example of TLPs, DLLPs, IDLs, and an EDS Token followed by a SKP Ordered Set. SKP Ordered Sets are defined in Section 4.2.7.2.

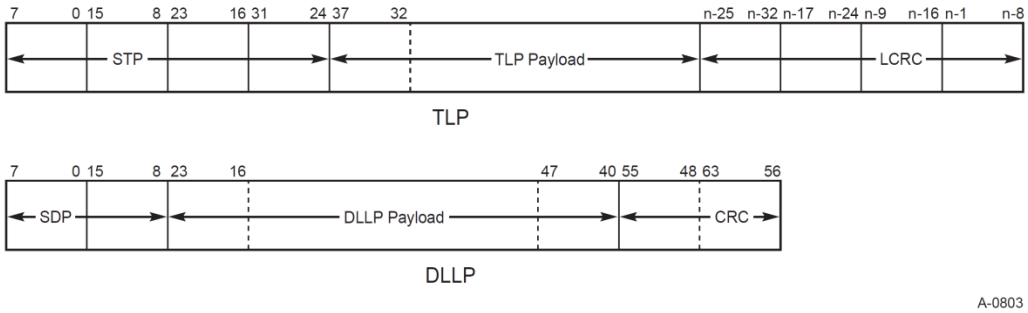


Figure 4-14: TLP and DLLP Layout

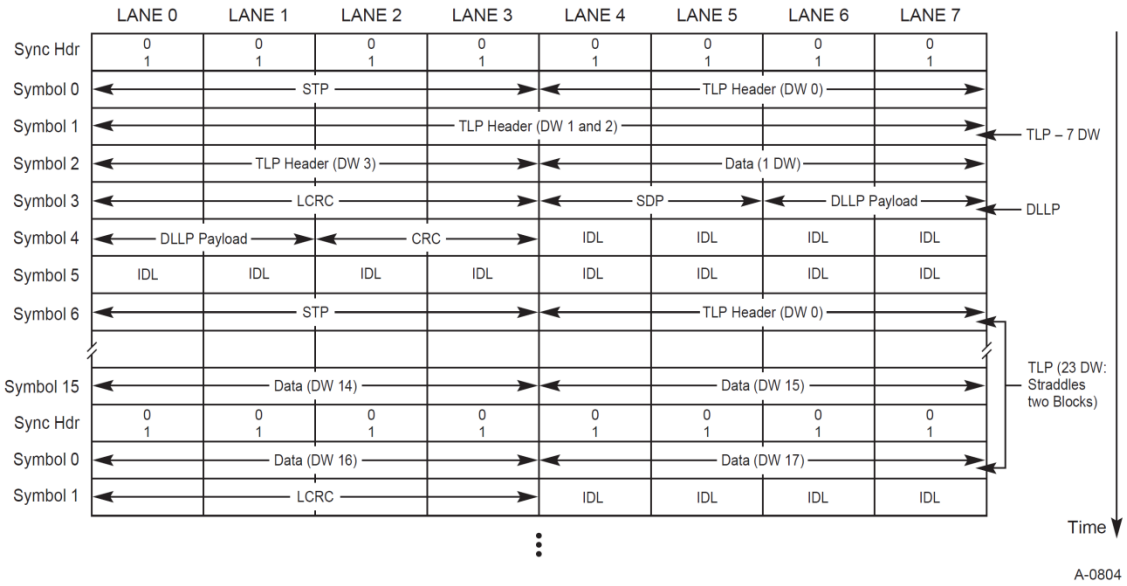


Figure 4-15: Packet Transmission in a x8 Link

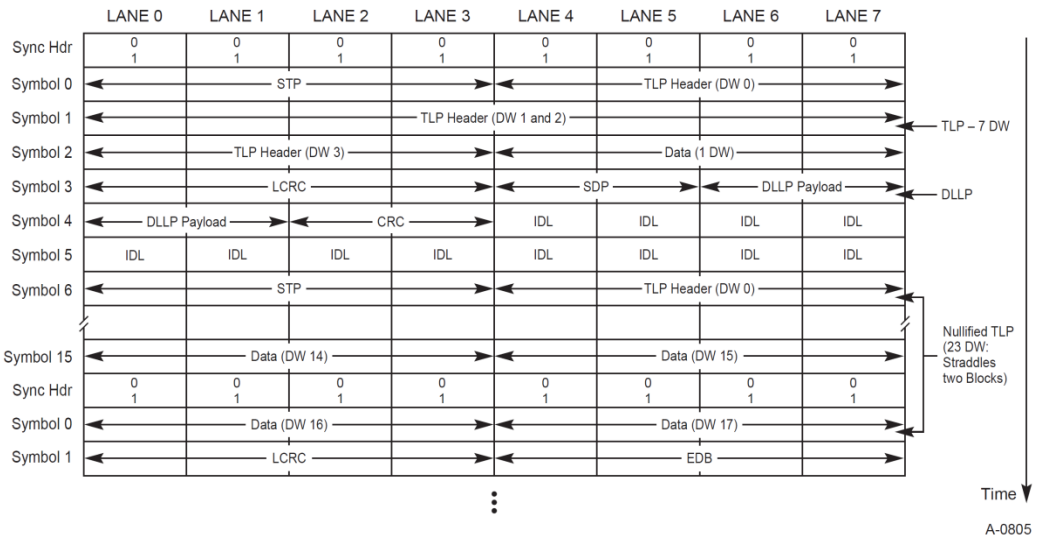


Figure 4-16: Nullified TLP Layout in a x8 Link with Other Packets

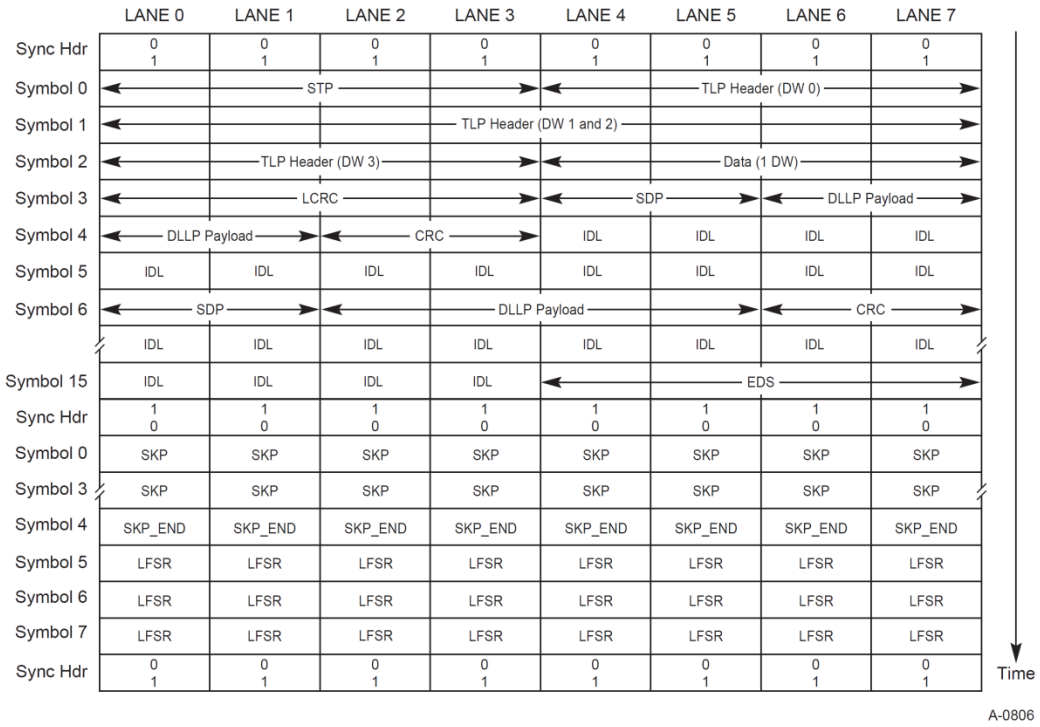


Figure 4-17: SKP Ordered Set of Length 66-bit in a x8 Link

4.2.2.3.2 Transmitter Framing Requirements

The following requirements apply to the transmitted Data Stream.

- To Transmit a TLP:
  - Transmit an STP Token immediately followed by the complete TLP information provided by the Data Link Layer.
  - All DWs of the TLP, as specified by the TLP Length field of the STP Token, must be transmitted, even if the TLP is nullified.
  - If the TLP is nullified, an EDB Token must be transmitted immediately following the TLP. There must be no Symbols between the last Symbol of the TLP and the first Symbol of the EDB Token. The value of the TLP Length field of a nullified TLP's STP Token is NOT adjusted to account for the EDB Token.
  - The STP Token must not be transmitted more frequently than once per Symbol Time.
- To Transmit a DLLP:
  - Transmit an SDP Token immediately followed by the complete DLLP information provided by the Data Link Layer.
  - All 6 Symbols of the DLLP must be transmitted.
  - The SDP Token must not be transmitted more frequently than once per Symbol Time.
- To Transmit a SKP Ordered Set within a Data Stream:
  - Transmit an EDS Token in the last DW of the current Data Block. For example, the Token is transmitted on Lane 0 in Symbol Times 12-15 of the Block for a x1 Link, and on Lanes 12-15 of Symbol Time 15 of the Block for a x16 Link.
  - Transmit the SKP Ordered Set following the current Data Block.
  - Transmit a Data Block following the SKP Ordered Set. The Data Stream resumes with the first Symbol of the Data Block. If multiple SKP Ordered Sets are scheduled for transmission, each SKP Ordered Set must be preceded by a Data Block with an EDS Token.
- To end a Data Stream:
  - Transmit an EDS Token in the last DW of the current Data Block, followed in the next block by an EIOS or an EIEOS. An EIOS is transmitted for LTSSM power management state transitions, and an EIEOS is transmitted for all other cases. For example, the Token is transmitted on Lane 0 in Symbol Times 12-15 of the Block for a x1 Link, and on Lanes 12-15 of Symbol Time 15 of the Block for a x16 Link.
- The IDL Token must be transmitted on all Lanes when not transmitting a TLP, DLLP, or other Framing Token.

- Multi-Lane Links:
  - After transmitting an IDL Token, the first Symbol of the next STP or SDP Token must be transmitted in Lane 0 of a future Symbol Time. An EDS Token can be transmitted after an IDL Token in the same Symbol Time, since it must be transmitted in the last DW of a Block.
  - For xN Links where N is 8 or more, if an EDB Token, TLP, or DLLP ends in a Lane K, where K does not equal N-1, and it is not followed by the first Symbol of an STP, SDP, or EDB Token in Lane K+1, then IDL Tokens must be placed in Lanes K+1 to N-1. For example, on a x8 Link, if a TLP or DLLP ends in Lane 3, IDL Tokens must be placed in Lanes 4 to 7. The EDS Token is an exception to this requirement, and can be transmitted following IDL Tokens.
  - Tokens, TLPs, and DLLPs are permitted to follow each other successively such that more than one Token may be transmitted in the same Symbol Time as long as their transmission conforms with the other requirements stated in this section.
    - Links wider than x4 can have Tokens placed starting on Lane 4*N, where N is a positive integer. For example, Tokens can be placed in Lanes 0 and 4 of a x8 Link, and Tokens can be placed in Lanes 0, 4, 8, or 12 of a x16 Link.

4.2.2.3.3 Receiver Framing Requirements

The following requirements apply to the received Data Stream and the Block type transitions that occur at the beginning and end of the Data Stream.

- When processing Symbols that are expected to be a Framing Token, receiving a Symbol or sequence of Symbols that does not match the definition of a Framing Token is a Framing Error. It is strongly recommended that Receivers of a multi-Lane Link report an error in the Lane Error Status Register for the Lane that receives the first Symbol of an expected Framing Token when that Symbol does not match Symbol +0 of an STP (bits [3:0] only), IDL, SDP, EDB, or EDS Token (see Figure 4-13).
- All optional error checks and error reports in this section are independently optional (see Section 6.2.3.4).
- When an STP Token is received:
  - Receivers must calculate the Frame CRC and Frame Parity of the received TLP Length field and compare the results to the received Frame CRC and Frame Parity fields. A Frame CRC or Frame Parity mismatch is a Framing Error.
    - An STP Token with Framing Error is not considered part of a TLP for the purpose of reporting to the Data Link Layer.
  - If the TLP Length field is 1, the Symbols are not an STP Token and are instead evaluated to determine whether they are an EDS Token.
  - Receivers are permitted to check whether the TLP Length field has a value of 0. If checked, receiving a TLP Length field of 0 is a Framing Error.
  - Receivers are permitted to check whether the TLP Length field has a value of 2, 3, or 4. If checked, receiving such a TLP Length field is a Framing Error.

- Receivers are permitted to check whether the TLP Length field has a value between 1152 and 1535 (inclusive). If checked, receiving such a TLP Length field is a Framing Error.
- Receivers on Ports that do not support Protocol Multiplexing are permitted to check whether the TLP Length field has a value greater than 1535. If checked, receiving such a TLP Length field is a Framing Error.
- Receivers on Ports that support Protocol Multiplexing, shall process STP Tokens with a TLP Length field that is greater than 1535 as the start of a PMUX Packet as defined in Appendix G.
- The Symbol immediately following the last DW of the TLP is the next Token to be processed.
- Receivers must evaluate the Symbol immediately following the last DW of the TLP and determine whether it is the first Symbol of an EDB Token and therefore whether the TLP is nullified. See the EDB Token requirements.
- Receivers are permitted to check whether more than one STP Token is received in a single Symbol Time. If checked, receiving more than one STP Token in a single Symbol Time is a Framing Error

❑ When an EDB Token is received:

- If an EDB Token is received immediately following a TLP (there are no Symbols between the last Symbol of the TLP and the first Symbol of the EDB Token), receivers must inform the Data Link Layer that an EDB Token has been received. Receivers are permitted to inform the Data Link Layer that an EDB Token has been received after processing the first Symbol of the EDB Token or after processing any or all of the remaining Symbols of the EDB Token. Regardless of when they inform the Data Link Layer of a received EDB Token, Receivers must check all Symbols of the EDB Token. Receiving a Symbol that does not match the definition of an EDB Token is a Framing Error.
- Receiving an EDB Token at any time other than immediately following a TLP is a Framing Error.
- The Symbol immediately following the EDB Token is the next Token to be processed.

❑ When an EDS Token is received in the last four Symbols of the Data Block across the Link:

- Receivers must stop processing the Data Stream.
- Receiving an Ordered Set other than SKP, EIOS, or EIEOS in the Block following the EDS Token is a Framing Error.
- If a SKP Ordered Set is received in the Block following the EDS Token, Receivers resume Data Stream processing with the first Symbol of the Data Block that follows the SKP Ordered Set unless a Framing Error has been detected.

❑ When an SDP Token is received:

- The Symbol immediately following the last Symbol of the DLLP is the next Token to be processed.

- Receivers are permitted to check whether more than one SDP Token is received in a single Symbol Time. If checked, receiving more than one SDP Token in a single Symbol Time is a Framing Error.

- When an IDL Token is received:
  - For a x1 Link, the next Token to be processed is the next Symbol received.
  - For a x2 Link, the next Token to be processed is the Symbol received in Lane 0 of the next Symbol Time. It is strongly recommended that Receivers check whether the Symbol received in Lane 1, if it did not receive IDL, after an IDL Token was received in Lane 0 is also IDL and report an error for Lane 1 in the Lane Error Status Register. If checked, receiving a Symbol other than IDL is a Framing Error.
  - For a x4 Link, the next Token to be processed is the Symbol received in Lane 0 of the next Symbol Time. It is strongly recommended that Receivers check whether the Symbols received in Lanes 1-3, after an IDL Token was received in Lane 0 are also IDL and report an error for the Lane(s) that did not receive IDL, in the Lane Error Status Register. If checked, receiving a Symbol other than IDL is a Framing Error.
  - For x8, x12, x16, and x32 Links, the next Token to be processed is the Symbol received in the next DW aligned Lane following the IDL Token. For example, if an IDL Token is received in Lane 4 of a x16 Link, the next Token location is Lane 8 of the same Symbol Time. However, if an IDL Token is received on Lane 4 of a x8 Link, the next Token location is Lane 0 of the following Symbol Time. It is strongly recommended that Receivers check whether the Symbols received between the IDL Token and the next Token location are also IDL and report an error for the Lane(s) that did not receive IDL, in the Lane Error Status Register. If checked, receiving a Symbol other than IDL is a Framing Error.

    Note: The only Tokens expected to be received in the same Symbol Time following an IDL Token are additional IDL Tokens or an EDS Token.

- While processing the Data Stream, Receivers must also check the Block type received by each Lane, after accounting for Lane-to-Lane de-skew, for the following conditions:
  - Receiving an Ordered Set Block on any Lane immediately following an SDS Ordered Set is a Framing Error.
  - Receiving a Block with an undefined Block type (a Sync Header of 11b or 00b) is a Framing Error. It is strongly recommended that Receivers of a multi-Lane Link report an error for any Lane that received the undefined Block type in the Lane Error Status register.
  - Receiving an Ordered Set Block on any Lane without receiving an EDS token in the preceding Block is a Framing Error. For example, receiving a SKP Ordered Set without a preceding EDS Token is a Framing Error. In addition, receiving a SKP Ordered Set followed immediately by another Ordered Set Block (including another SKP Ordered Set) within a Data Stream is a Framing Error. It is strongly recommended that if the first Symbol of the Ordered Set is SKP, Receivers of a multi-Lane Link report an error for the Lane(s) in the Lane Error Status register if the received Symbol number 1 through 4N does not match the corresponding Symbol in Table 4-16: .

- Receiving a Data Block on any Lane when the previous block contained an EDS Token is a Framing Error. It is strongly recommended that Receivers of a multi-Lane Link report an error for the Lane(s) that received the Data Block in the Lane Error Status register.
- Receivers are permitted to check for different Ordered Sets on different Lanes. For example, Lane 0 receives a SKP Ordered Set and Lane 1 receives an EIOS. If checked, receiving different Ordered Sets is a Framing Error.

4.2.2.3.4 Recovery from Framing Errors

If a receiver detects a Framing Error while processing the Data Stream, it must:

- Report a Receiver Error as described in Section 4.2.4.7.
- Stop processing the Data Stream. Processing of a new Data Stream is initiated when the next SDS Ordered Set is received as previously described.
- Initiate the error recovery process as described in Section 4.2.4.7. If the LTSSM state is L0, direct the LTSSM to Recovery. If the LTSSM state is Configuration.Complete or Configuration.Idle when the Framing Error is detected, the error recovery process is satisfied by either a transition from Configuration.Idle to Recovery.RcvrLock due to the specified timeout, or a directed transition from L0 to Recovery. If the LTSSM state is Recovery.RcvrCfg or Recovery.Idle when the Framing Error is detected, the error recovery process is satisfied by either a transition from Recovery.Idle to Recovery.RcvrLock due to the specified timeout, or a directed transition from L0 to Recovery. If the LTSSM substate is either Recovery.RcvrLock or Configuration.Linkwidth.Start, the error recovery process is satisfied upon exit from these substates and no direction of the LTSSM to Recovery is required;
  - Note: The framing error recovery mechanism is not expected to directly cause any Data Link Layer initiated recovery action such as NAK.

IMPLEMENTATION NOTE

Time Spent in Recovery Due to Detection of a Framing Error

When using 128b/130b encoding, all Framing Errors require Link recovery. It is expected that implementations will require less than 1 microsecond to recover from a Framing Error as measured from the time that both Ports have entered the Recovery state.

4.2.2.4 *Scrambling*

Each Lane of the transmitter in a multi-Lane Link may implement a separate LFSR for scrambling. Each Lane of the receiver in a multi-Lane Link may implement a separate LFSR for descrambling. Implementations may choose to implement fewer LFSRs, but must achieve the same functionality as independent LFSRs.

The LFSR uses the following polynomial: $G(X) = X^{23} + X^{21} + X^{16} + X^{8} + X^{5} + X^{2} + 1$ and is demonstrated in Figure 4-18.

The scrambling rules are as follows:

- The two bits of the Sync Header are not scrambled and do not advance the LFSR.
- All 16 Symbols of an Electrical Idle Exit Ordered Set (EIEOS) bypass scrambling. The scrambling LFSR is initialized after the last Symbol of an EIEOS is transmitted, and the descrambling LFSR is initialized after the last Symbol of an EIEOS is received.
- TS1 and TS2 Ordered Sets:
  - Symbol 0 of a TS1 or TS2 Ordered Set bypasses scrambling.
  - Symbols 1-13 are scrambled.
  - Symbols 14 and 15 bypass scrambling if required for DC Balance, but they are scrambled if not required for DC Balance.
- All 16 Symbols of a Fast Training Sequence (FTS) Ordered Set bypass scrambling.
- All 16 Symbols of a Start of Data Stream (SDS) Ordered Set bypass scrambling.
- All 16 Symbols of an Electrical Idle Ordered Set (EIOS) bypass scrambling.
- All Symbols of a SKP Ordered Set bypass scrambling.
- Transmitters advance their LFSR for all Symbols of all Ordered Sets except for the SKP Ordered Set. The LFSR is not advanced for any Symbols of a SKP Ordered Set.
- Receivers evaluate Symbol 0 of Ordered Set Blocks to determine whether to advance their LFSR. If Symbol 0 of the Block is SKP (see Section 4.2.7.2), then the LFSR is not advanced for any Symbol of the Block. Otherwise, the LFSR is advanced for all Symbols of the Block.
- All 16 Symbols of a Data Block are scrambled and advance the scrambler.
- For Symbols that need to be scrambled, the least significant bit is scrambled first and the most significant bit is scrambled last.
- The seed value of the LFSR is dependent on the Lane number assigned to the Lane when the Link first entered Configuration.Idle (i.e., having gone through Polling from Detect with LinkUp = 0b).
  - The seed values for Lane number modulo 8 are:
    - 0: 1DBFBCh
    - 1: 0607BBh
    - 2: 1EC760h
    - 3: 18C0DBh
    - 4: 010F12h
    - 5: 19CFC9h
    - 6: 0277CEh
    - 7: 1BB807h

IMPLEMENTATION NOTE

Scrambling Pseudo-code

The pseudo-code for the scrambler along with examples is provided in Section C.2 of Appendix C.

- The seed value of the LFSR does not change while LinkUp=1. Link reconfiguration through the LTSSM Configuration state does not modify the initial Lane number assignment as long as the LinkUp remains 1 (even though the Lane assignment may change during Configuration).
- Scrambling cannot be disabled in Configuration.Complete when using 128b/130b encoding.
- A Loopback Slave must not descramble or scramble the looped-back bit stream.

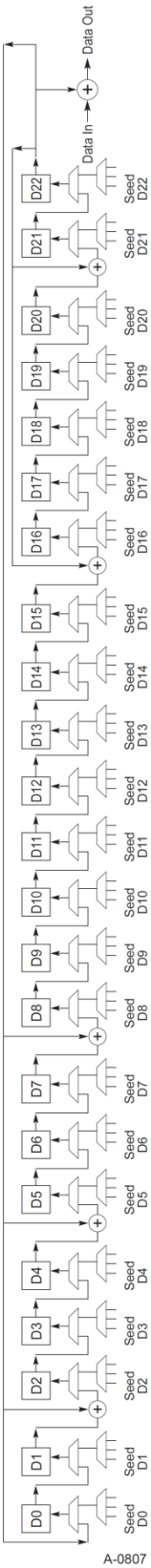


Figure 4-18: LFSR with Scrambling Polynomial in 8.0 GT/s and Above Data Rate

IMPLEMENTATION NOTE

LFSR Implementation with a Shared LFSR

Implementations may choose to implement one LFSR and take different tap points as shown in Figure 4-19, which is equivalent to the individual LFSR per-lane with different seeds, as shown in Figure 4-18. It should also be noted that the tap equations of four Lanes are the XOR of the tap equations of two neighboring Lanes. For example, Lane 0 can be obtained by XORing the output of Lanes 1 and 7; Lane 2 is the XOR of Lanes 1 and 3; Lane 4 is the XOR of Lanes 3 and 5; and Lane 6 is the XOR of Lanes 5 and 7. This can be used to help reduce the gate count at the expense of potential delay due to the XOR results of the two Lanes.

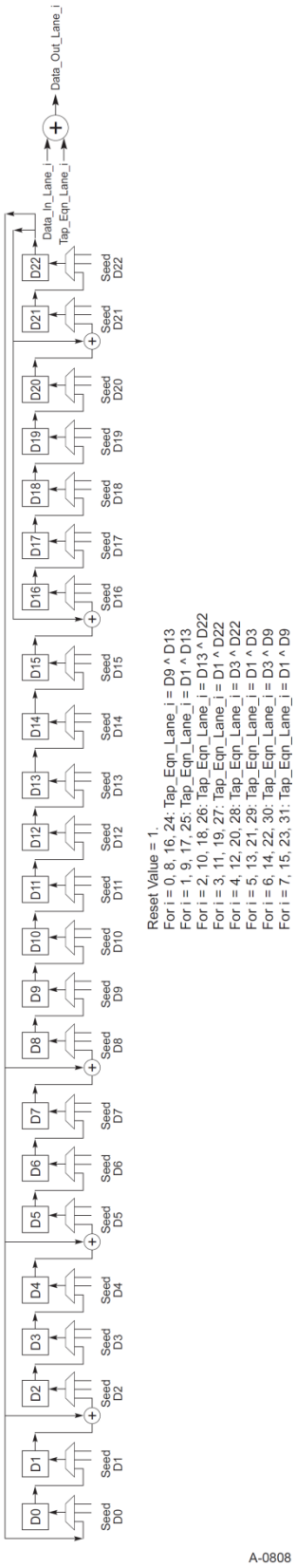


Figure 4-19: Alternate Implementation of the LFSR for Descrambling

4.2.2.5 *Loopback with 128b/130b Code*

When using 128b/130b encoding, Loopback Masters must transmit Blocks with the defined 01b and 10b Sync Headers. However, they are not required to transmit an SDS Ordered Set when transitioning from Ordered Set Blocks to Data Blocks, nor are they required to transmit an EDS Token when transitioning from Data Blocks to Ordered Set Blocks. Masters must transmit SKP Ordered Sets periodically as defined in Section 4.2.7, and they must be capable of processing received (looped-back) SKP Ordered Sets of varying length. Masters are permitted to transmit Electrical Idle Exit Ordered Sets (EIEOS) as defined in Section 4.2.2.2.1. Masters are permitted to transmit any payload in Data Blocks and Ordered Set Blocks that they expect to be looped-back. If the Loopback Master transmits an Ordered Set Block whose first symbol matches the first symbol of SKP OS, EIEOS, or EIOS, that Ordered Set Block must be a complete and valid SKP OS, EIEOS, or EIOS.

When using 128b/130b encoding, Loopback Slaves must retransmit all bits received without modification, except for SKP Ordered Sets which can be adjusted as needed for clock compensation. If clock compensation is required, slaves must add or remove 4 SKP Symbols per Ordered Set. The modified SKP Ordered Set must meet the definition of Section 4.2.7.2 (i.e., it must have between 4 to 20 SKP Symbols followed by the SKP_END Symbol and the three Symbols that follow it as transmitted by the Loopback Master). If a slave is unable to obtain Block alignment or it is misaligned, it may be unable to perform clock compensation and therefore unable to loop-back all bits received. In this case, it is permitted to add or remove Symbols as necessary to continue operation. Slaves must not check for a received SDS Ordered Set when a transition from Ordered Set Blocks to Data Blocks is detected, and they must not check for a received EDS Token when a transition from Data Blocks to Ordered Set Blocks is detected.

4.2.3 Link Equalization Procedure for 8.0 GT/s and Higher Data Rates

The Link equalization procedure enables components to adjust the Transmitter and the Receiver setup of each Lane to improve the signal quality and meet the requirements specified in Chapter 8, when operating at 8.0 GT/s and higher data rates. All the Lanes that are associated with the LTSSM (i.e., those Lanes that are currently operational or may be operational in the future due to Link Upconfigure) must participate in the Equalization procedure. The procedure must be executed during the first data rate change to 8.0 GT/s as well as the first change to all data rates greater than 8.0 GT/s. Components must arrive at the appropriate Transmitter setup for all the operating conditions and data rates that they will encounter in the future when LinkUp=1b. Components must not require that the equalization procedure be repeated at any data rate for reliable operation, although there is provision to repeat the procedure. Components must store the Transmitter setups that were agreed to during the equalization procedures and use them for future operation at 8.0 GT/s and higher data rates. Components are permitted to fine-tune their Receiver setup even after the equalization procedure is complete as long as doing so does not cause the Link to be unreliable (i.e., does not meet the requirements in Chapter 8) or go to Recovery.

The equalization procedure can be initiated either autonomously or by software. It is strongly recommended that components use the autonomous mechanism. However, a component that chooses not to participate in the autonomous mechanism must have its associated software ensure that the software based mechanism is applied.

The autonomous mechanism is executed if both components advertise that they are capable of at least the 8.0 GT/s data rate (via the TS1 and TS2 Ordered Sets) during the initial Link negotiation (when LinkUp is set to 1b). If both components advertised support for 8.0 GT/s and 16.0 GT/s, the Downstream Port may choose to only perform the 8.0 GT/s equalization procedure using the autonomous mechanism. If both components advertised support for 16.0 GT/s but only the 8.0 GT/s equalization procedure is performed using the autonomous mechanism, the software based mechanism must be executed in order to perform the 16.0 GT/s equalization procedure. After entering L0, irrespective of the current Link speed, neither component must transmit any DLLP if the equalization procedure must be performed and until the equalization procedure completes. The equalization procedure is considered complete once the Transmitter and Receiver setup of each Lane has been adjusted for each common data rate supported above 5.0 GT/s for which the Downstream Port intends to perform equalization using the autonomous mechanism. The Downstream Port is required to make the transition from L0 to Recovery to change the data rate to 8.0 GT/s and perform the equalization procedure; the Upstream Port is permitted, but not required, to make the transition from L0 to Recovery autonomously. The Downstream Port must not advertise 16.0 GT/s support in Recovery if it entered Recovery with the intention of performing an 8.0 GT/s equalization procedure. This applies to the autonomous equalization procedure as well as the software initiated equalization procedure, or any subsequent equalization redo. The Downstream Port must not advertise 16.0 GT/s support in Recovery until the 8.0 GT/s equalization procedure has been successfully executed. The 8.0 GT/s equalization procedure is deemed to have been successfully executed if the Equalization 8.0 GT/s Phase 3 Successful bit and Equalization 8.0 GT/s Complete bit of the Link Status 2 register are both set to 1b. Immediately following the transition from Recovery to L0, after the initial data rate change to 8.0 GT/s, the Downstream Port is required to transition from L0 to Recovery, advertise 16.0 GT/s data rate support, change the data rate to 16.0 GT/s and perform the 16.0 GT/s equalization procedure if both components advertised that they are capable of 16.0 GT/s during the initial Link negotiation, neither component detected problems with its 8.0 GT/s equalization settings and it intends to perform a 16.0 GT/s equalization procedure using the autonomous mechanism. If the Downstream Port detected 8.0 GT/s equalization problems or the Upstream Port made an 8.0 GT/s equalization redo request (by setting the Request Equalization bit to 1b and the Equalization Request Data Rate to 0b in the TS2s it sent in Recovery.RcvrCfg) the Downstream Port may redo 8.0 GT/s equalization prior to proceeding to the 16.0 GT/s equalization procedure. The number of 8.0 GT/s equalization redos initiated prior to the 16.0 GT/s equalization is implementation specific but must be finite. If at the conclusion of the initial or subsequent 8.0 GT/s equalization process and the execution of an implementation specific number of equalization redo's, the Link is not able to operate reliably at the 8.0 GT/s data rate, then it must revert back to 2.5 GT/s or 5.0 GT/s Data Rate. Speed changes to the 16.0 GT/s data rate and subsequent 16.0 GT/s equalization process must not be attempted regardless of whether both Link partners ever advertised support for 16.0 GT/s unless a subsequent equalization redo at the 8.0 GT/s data rate is successful.

Components using the autonomous mechanism must not initiate any autonomous Link width downsizing until the equalization procedure completes. An Upstream Port must not transmit any DLLP until it receives a DLLP from the Downstream Port. If the Downstream Port performs equalization again, it must not transmit any DLLP until it completes the equalization procedure. A Downstream Port may perform equalization again based on its own needs or based on the request from the Upstream Port, if it can meet its system requirements. Executing equalization multiple times may interfere with software determination of Link and device status, as described in Section 6.6.

IMPLEMENTATION NOTE

DLLP Blocking During Autonomous Equalization

When using the autonomous mechanism for equalization at 8.0 GT/s or higher data rates, the Downstream Port is required to block the transmission of DLLPs until equalization has completed, and the Upstream Port is required to block the transmission of DLLPs until a DLLP is received from the Downstream Port. If both components advertise that they are capable of the 16.0 GT/s data rate but the Downstream Port only uses the autonomous mechanism for equalization at 8.0 GT/s, the Downstream Port is only required to block DLLP transmission until 8.0 GT/s equalization has completed. If the Downstream Port delays entering Recovery from L0 while DLLP transmission is blocked, either the L0 Inferred Electrical Idle timeout (see Section 4.2.4.3) or the DLLP timeout (see Section 2.6.1.2) may expire in the Upstream or Downstream Ports. If either of these two timeouts occurs, it will result in the initiation of an entry to Recovery to perform Link retraining. Neither of these two timeouts is a reportable error condition, and the resulting Link retraining has no impact on proper Link operation.

When using the software based mechanism, software must guarantee that there will be no side-effects for transactions in flight (e.g., no timeout), if any, due to the Link undergoing the equalization procedure. Software can write 1b to the Perform Equalization bit in the Link Control 3 register, followed by a write to the Target Link Speed field in the Link Control 2 register to enable the Link to run at 8.0 GT/s or higher, followed by a write of 1b to the Retrain Link bit in the Link Control register of the Downstream Port to perform equalization. Software must not enable the Link to run at 16.0 GT/s during a software initiated equalization procedure if the 8.0 GT/s equalization procedure has not been successfully executed. The 8.0 GT/s equalization procedure is deemed to have been successfully executed if the Equalization 8.0 GT/s Phase 3 Successful bit and Equalization 8.0 GT/s Complete bit of the Link Status 2 register are both set to 1b. Software may set the Hardware Autonomous Width Disable of the Link Control register in both components or use some other mechanism to ensure that the Link is in its full functional width prior to setting the Perform Equalization bit in the Downstream Port. The component that had initiated the autonomous width downsizing is responsible to upconfigure the Link to go to its full functional width by initiating the transition to Recovery and Configuration within 1 ms of the Hardware Autonomous Width Disable bit being set to 1b. If an Upstream Port does not advertise the 8.0 GT/s data rate, the 16.0 GT/s data rate, or both the 8.0 GT/s and 16.0 GT/s data rates initially and did not participate in the autonomous equalization mechanism for the non-advertised rates, its associated software must ensure there will be no side-effects for transactions in flight, if any, during equalization, before it instructs the Upstream Port to go to Recovery and advertise the previously non-advertised data rates and initiate a speed change. The Downstream Port subsequently initiates the equalization procedure during the initial speed change to the data rate advertised by the Upstream Port when it transitions to Recovery. If the Upstream Port advertises support for both 8.0 GT/s and 16.0 GT/s data rates after advertising support for less than 8.0 GT/s initially, the Downstream Port must advertise 8.0 GT/s support only and perform an 8.0 GT/s speed change and equalization procedure prior to performing speed changes and equalization procedures at the higher data rate. Upon completion of the 8.0 GT/s speed change and equalization, the Downstream Port must advertise its next highest data rate supported and, if the same rate is supported by the Upstream Port, transition from L0 to Recovery in order to perform a speed change to the higher

data rate and initiate equalization at that speed. The 16.0 GT/s equalization procedure must be initiated when operating at 8.0 GT/s.

Upstream Ports are required to check for equalization setting problems in the Recovery.RcvrLock state (see Section 4.2.6.4.1). However, both Downstream and Upstream Ports are permitted to use implementation-specific methods to detect equalization problems at any time. A Port that detects a problem with its 8.0 GT/s data rate equalization settings must set the Link Equalization Request 8.0 GT/s bit in its Link Status 2 register to 1b, and a Port that detects a problem with its 16.0 GT/s data rate equalization settings must set the Link Equalization Request 16.0 GT/s bit in its 16.0 GT/s Status register to 1b. In addition to setting the appropriate Link Equalization Request bit, an Upstream Port must initiate a transition to Recovery (if necessary) and request equalization at the appropriate data rate in the Recovery.RcvrCfg state by setting the Request Equalization bit of its transmitted TS2 Ordered Sets to 1b and the Equalization Request Data Rate bit to the data rate of the detected problem. If it requests equalization, it must request equalization for each detected problem only once. When requesting equalization, the Upstream Port is also permitted, but not required, to set the Quiesce Guarantee bit to 1b to inform the Downstream Port that an equalization process initiated within 1 ms will not cause any side-effects to its operation.

When a Downstream Port receives an equalization request from an Upstream Port (when it is in the Recovery.RcvrCfg state and receives 8 consecutive TS2 Ordered Sets with the Request Equalization bit set to 1b), it must either initiate an equalization process at the requested data rate (as defined by the received Equalization Request Data Rate bit) within 1 ms of completing the next Recovery to L0 transition, or it must set the appropriate Link Equalization Request 8.0 GT/s in its Link Status 2 register or Link Equalization Request 16.0 GT/s bit in its 16.0 GT/s Statusregister. It should initiate an equalization process only if it can guarantee that executing the equalization process will not cause any side-effects to either its operation or the Upstream Port's operation. The Downstream Port is permitted, but not required, to use the received Quiesce Guarantee bit to determine the Upstream Port's ability to execute an equalization process without side-effects.

If a Downstream Port wants to initiate an equalization process and can guarantee that it will not cause side-effects to its own operation but is unable to directly determine whether the equalization process will cause side-effects to the Upstream Port's operation, then it is permitted to request that the Upstream Port initiate an equalization request. The Downstream Port does so by transitioning to Recovery and in the Recovery.RcvrCfg state setting the Request Equalization bit of its transmitted TS2 Ordered Sets to 1b, the Equalization Request Data Rate bit to the desired data rate, and the Quiesce Guarantee bit to 1b. When an Upstream Port receives such an equalization request from a Downstream Port (when it is in the Recovery.RcvrCfg state and receives 8 consecutive TS2 Ordered Sets with the Request Equalization and Quiesce Guarantee bits set to 1b), it is permitted, but not required, to quiesce its operation and prepare to execute an equalization process at the data rate requested by the Downstream Port, and then request equalization at that same data rate (using the method described previously for reporting equalization setting problems) and with the Quiesce Guarantee bit set to 1b. There is no time limit on how long the Upstream Port can take to respond, but it should attempt to do so as quickly as possible. If a Downstream Port makes a request and receives such a response from the Upstream Port, then it must either initiate an equalization process at the agreed-upon data rate within 1 ms of completing the next Recovery to L0 transition if it can still guarantee that executing the equalization process will not cause any side-effects to its operation, or it must set the appropriate Link Equalization Request 8.0 GT/s in its Link Status 2 register or Link Equalization Request 16.0 GT/s bit in its 16.0 GT/s Status register.

IMPLEMENTATION NOTE

Using Quiesce Guarantee Mechanism

Side-effects due to executing equalization after the Data Link Layer is in DL_Active can occur at the Port, Device, or system level. For example, the time required to execute the equalization process could cause a Completion Timeout error to occur – possibly in a different system component. The Quiesce Guarantee information can help Ports decide whether to execute a requested equalization or not.

A component may operate at a lower data rate after reporting its equalization problems, either by timing out through Recovery.Speed or by initiating a data rate change to a lower data rate. Any data rate change required to perform the equalization procedure is exempt from the 200 ms requirement in Section 6.11. If the Downstream Port wants to redo equalization at 8.0 GT/s and the current data rate is either 2.5 GT/s or 5.0 GT/s, the Downstream Port must request speed change through Equalization (EQ) TS1 Ordered Sets in Recovery.RcvrLock to inform the Upstream Port that it intends to redo equalization. An Upstream Port should advertise 8.0 GT/s data rate in Recovery if it receives EQ TS1 Ordered Sets with speed change bit set to 1b and if it intends to operate at 8.0 GT/s data rate in the future. If the Downstream Port wants to redo equalization at 16.0 GT/s and the current data rate is either 2.5 GT/s or 5.0 GT/s, the Downstream Port must first initiate a speed change to 8.0 GT/s (the 8.0 GT/s equalization procedure will not be executed unless necessary). If the Downstream Port wants to redo equalization at 16.0 GT/s and the current data rate is 8.0 GT/s, the Downstream Port must request speed change through TS1 Ordered Sets in Recovery.RcvrLock with the Equalization Redo bit set to 1b to inform the Upstream Port that it intends to redo equalization. An Upstream Port should advertise 16.0 GT/s data rate in Recovery if it receives TS1 Ordered Sets with speed change bit set to 1b, Equalization Redo bit set to 1b and it intends to operate at 16.0 GT/s data rate in the future. The equalization procedure can be performed at most once in each trip through the Recovery state.

The equalization procedure consists of up to four Phases, as described below. When operating at 8.0 GT/s or higher data rates, the Phase information is transmitted using the Equalization Control (EC) field in the TS1 Ordered Sets.

Phase 0: When negotiating to 8.0 GT/s, the Downstream Port communicates the 8.0 GT/s Transmitter preset values and the 8.0 GT/s Receiver preset hints for each Lane to the Upstream Port using 8b/10b encoding. These values are communicated using the EQ TS2 Ordered Sets (defined in Section 4.2.4.1) in Recovery.RcvrCfg, when a data rate change to 8.0 GT/s has been negotiated, prior to transitioning to 8.0 GT/s data rate. These preset values are derived from the Upstream Port 8.0 GT/s Transmitter Preset and Upstream Port 8.0 GT/s Receiver Preset Hint fields of each Lane's Equalization Control register. After the data rate change to 8.0 GT/s, the Upstream Port transmits TS1 Ordered Sets with the preset values it received. The preset values must be within the operable range defined in Section 8.3.3.3 if reduced swing will be used by the Transmitter.

When negotiating to 16.0 GT/s, the Downstream Port communicates the 16.0 GT/s Transmitter preset values for each Lane to the Upstream Port using 128b/130b encoding. These values are communicated using 8GT EQ TS2 Ordered Sets (defined in Section 4.2.4.1) in Recovery.RcvrCfg, when a data rate change to 16.0 GT/s has been negotiated, prior to transitioning to 16.0 GT/s data rate. These preset values are derived from the Upstream Port

16.0 GT/s Transmitter Preset field of the 16.0 GT/s Lane Equalization Control Register entry corresponding to the Lane. Optionally, the Upstream Port communicates initial 16.0 GT/s Transmitter preset settings to the Downstream Port using the 8GT EQ TS2 Ordered Sets sent in Recovery.RcvrCfg, when a data rate change to 16.0 GT/s has been negotiated, prior to transitioning to 16.0 GT/s data rate. These preset values are determined by implementation specific means. After the data rate change to 16.0 GT/s, the Upstream Port transmits TS1 Ordered Sets with the preset values it received. If the Downstream Port received preset values in Recovery.RcvrCfg, after the data rate change to 16.0 GT/s, it transmits TS1 Ordered Sets with those preset values. The preset values must be within the operable range defined in Section 8.3.3.3 if reduced swing will be used by the Transmitter.

Phase 1: Both components make the Link operational enough at the current data rate to be able to exchange TS1 Ordered Sets to complete the remaining phases for the fine-tuning of their Transmitter/Receiver pairs. It is expected that the Link will operate at a BER of less than 10^{-4} before the component is ready to move on to the next Phase.

The Downstream Port initiates Phase 1 by transmitting TS1 Ordered Sets with EC=01b (indicating Phase 1) to the Upstream Port using the preset values received from the Upstream Port in Recovery.RcvrCfg if the current data rate of operation is 16.0 GT/s and eight consecutive 8GT EQ TS2 Ordered Sets were received in Recovery.RcvrCfg. Otherwise, the Downstream Port uses the preset values in the Downstream Port Transmitter Preset field of each Lane's Equalization Control register applicable to the current data rate of operation. The Upstream Port, after adjusting its Receiver, if necessary, to ensure that it can progress with the equalization process, receives these TS1 Ordered Sets and transitions to Phase 1 (where it transmits TS1 Ordered Sets with EC=01b). The Downstream Port ensures that it can reliably receive the bit stream from the Upstream Port to continue through the rest of the Phases when it receives TS1 Ordered Sets from the Upstream Port with EC=01b before it moves on to Phase 2.

Phase 2: In this Phase the Upstream Port adjusts the Transmitter setting of the Downstream Port along with its own Receiver setting, independently, on each Lane, to ensure it receives the bit stream compliant with the requirements in Chapter 8 (e.g., all operational Downstream Lanes have a BER is less than 10^{-12}). The Downstream Port initiates the move to Phase 2 by transmitting TS1 Ordered Sets with EC=10b to the Upstream Port. The Downstream Port advertises the Transmitter coefficients and the preset it is using per the rules below in Phase 1 for preset only and in Phase 2 for preset and coefficients. The Upstream Port receives these Ordered Sets and may request different coefficient or preset settings and continue to evaluate each setting until it arrives at the best setting for operating the Downstream Lanes. After the Upstream Port has completed this Phase, it moves the Link to Phase 3 by transmitting TS1 Ordered Sets with EC=11b to the Downstream Port.

Phase 3: In this Phase the Downstream Port adjusts the Transmitter setting of the Upstream Port along with its own Receiver setting, independently, on each Lane, using a handshake and evaluation process similar to Phase 2 with the exception that EC=11b. The Downstream Port signals the end of Phase 3 (and the equalization procedure) by transmitting TS1 Ordered Sets with EC=00b.

The algorithm used by a component to adjust the transmitter of its Link partner and the evaluation of that Transmitter set-up with its Receiver set-up is implementation-specific. A component may request changes to any number of Lanes and can request different settings for each Lane. Each

requested setting can be a preset or a set of coefficients that meets the requirements defined in Section 4.2.3.1. Each component is responsible for ensuring that at the end of the fine-tuning (Phase 2 for Upstream Ports and Phase 3 for Downstream Ports), its Link partner has the Transmitter setting in each Lane that will cause the Link to meet the requirements in Chapter 8.

A Link partner receiving the request to adjust its Transmitter must evaluate the request and act on it. If a valid preset value is requested and the Transmitter is operating in full-swing mode, it must be reflected in the Transmitter set-up and subsequently in the preset and coefficient fields of the TS1 Ordered Set that the Link partner transmits. If a preset value is requested, the Transmitter is operating in reduced-swing mode, and the requested preset is supported as defined in Section 8.3.3.3 it must be reflected in the Transmitter set-up and subsequently in the preset and coefficient fields of the TS1 Ordered Set that the Link partner transmits. Transmitters operating in reduced-swing mode are permitted to reject preset requests that are not supported as defined in Section 8.3.3.3. A request for adjusting the coefficients may be accepted or rejected. If the set of coefficients requested for a Lane is accepted, it must be reflected in the Transmitter set-up and subsequently in the transmitted TS1 Ordered Sets. If the set of coefficients requested for a Lane is rejected, the Transmitter set-up is not changed, but the transmitted TS1 Ordered Sets must reflect the requested coefficients along with the Reject Coefficient bit set to 1b. In either case of responding to a coefficient request, the preset field of the transmitted TS1 Ordered Sets is not changed from the last preset value that was transmitted. A request for adjusting the coefficients may be rejected by the Link partner only if the set of coefficients requested is not compliant with the rules defined in Section 4.2.3.1.

When performing equalization of a crosslink, the component that played the role of the Downstream Port during the earlier crosslink initialization at the lower data rate also assumes the responsibility of the Downstream Port for equalization.

If a Lane is directed to use a Reserved or unsupported Transmitter preset value in Polling.Compliance, Loopback, or Phase 0 or Phase 1 of Recovery.Equalization, then the Lane is permitted to use any supported Transmitter preset setting in an implementation-specific manner. The Reserved or unsupported Transmitter preset value is transmitted in any subsequent Compliance Patterns or Ordered Sets, and not the implementation-specific preset value chosen by the Lane. For example, if a Lane of an Upstream Port is directed to use Transmitter preset value 1111b (Reserved) with the EQ TS2 Ordered Sets it receives in Recovery.RcvrCfg, it is permitted to use any supported Transmitter preset value for its transmitter setting after changing the data rate to 8.0 GT/s, but it must transmit 1111b as its Transmitter preset value in the TS1 Ordered Sets it transmits in Phase 0 and Phase 1 of Recovery.Equalization.

In the Loopback state, the Loopback Master is responsible for communicating the Transmitter and Receiver settings it wants the Slave to use through the EQ TS1 Ordered Sets it transmits in the 2.5 GT/s or 5.0 GT/s data rate, and the preset or coefficient settings it wants the device under test to operate under in the TS1 Ordered Sets it transmits in the 8.0 GT/s or higher data rate. Similarly, if the Polling.Compliance state for 8.0 GT/s or higher Data Rates is entered through TS1 Ordered Sets, the entity that is performing the test is required to send the appropriate EQ TS1 Ordered Sets and coefficients for the device under test to operate with, according to the mechanism defined in Section 4.2.6.2.

IMPLEMENTATION NOTE

Equalization Example

The following diagram is an example illustrating how two devices may complete the equalization procedure. If the maximum common data rate supported by both Ports is 8.0 GT/s, the equalization procedure is complete at the conclusion of the 8.0 GT/s equalization procedure. If the maximum common data rate supported by both Ports is 16.0 GT/s, the 8.0 GT/s equalization procedure is followed by the 16.0 GT/s equalization procedure. If either the 8.0 GT/s or 16.0 GT/s equalization procedure is repeated and is performed while the Link is in 8.0 GT/s data rate (for the 8.0 GT/s equalization) or in 16.0 GT/s (for the 16.0 GT/s equalization), Phase 0 may be skipped since there is no need for the Link to go back to 2.5 GT/s or 5.0 GT/s (for the 8.0 GT/s equalization) or 8.0 GT/s (for the 16.0 GT/s equalization) to resend the same EQ TS2 Ordered Sets to convey the presets. A Downstream Port may choose to skip Phase 2 and Phase 3 if it determines that fine-tuning of the Transmitter is not needed based on the channel and components in the platform.

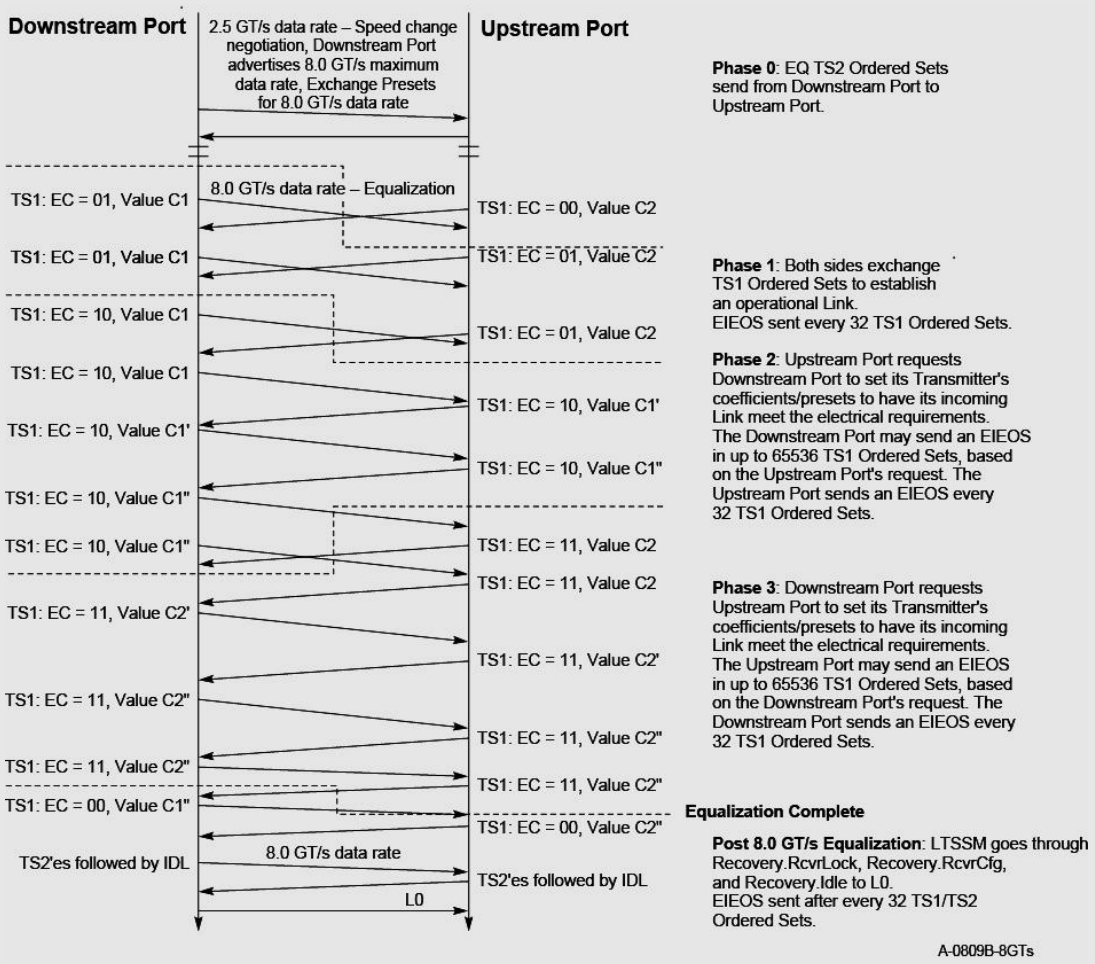


Figure 4-20: 8.0 GT/s Equalization Flow

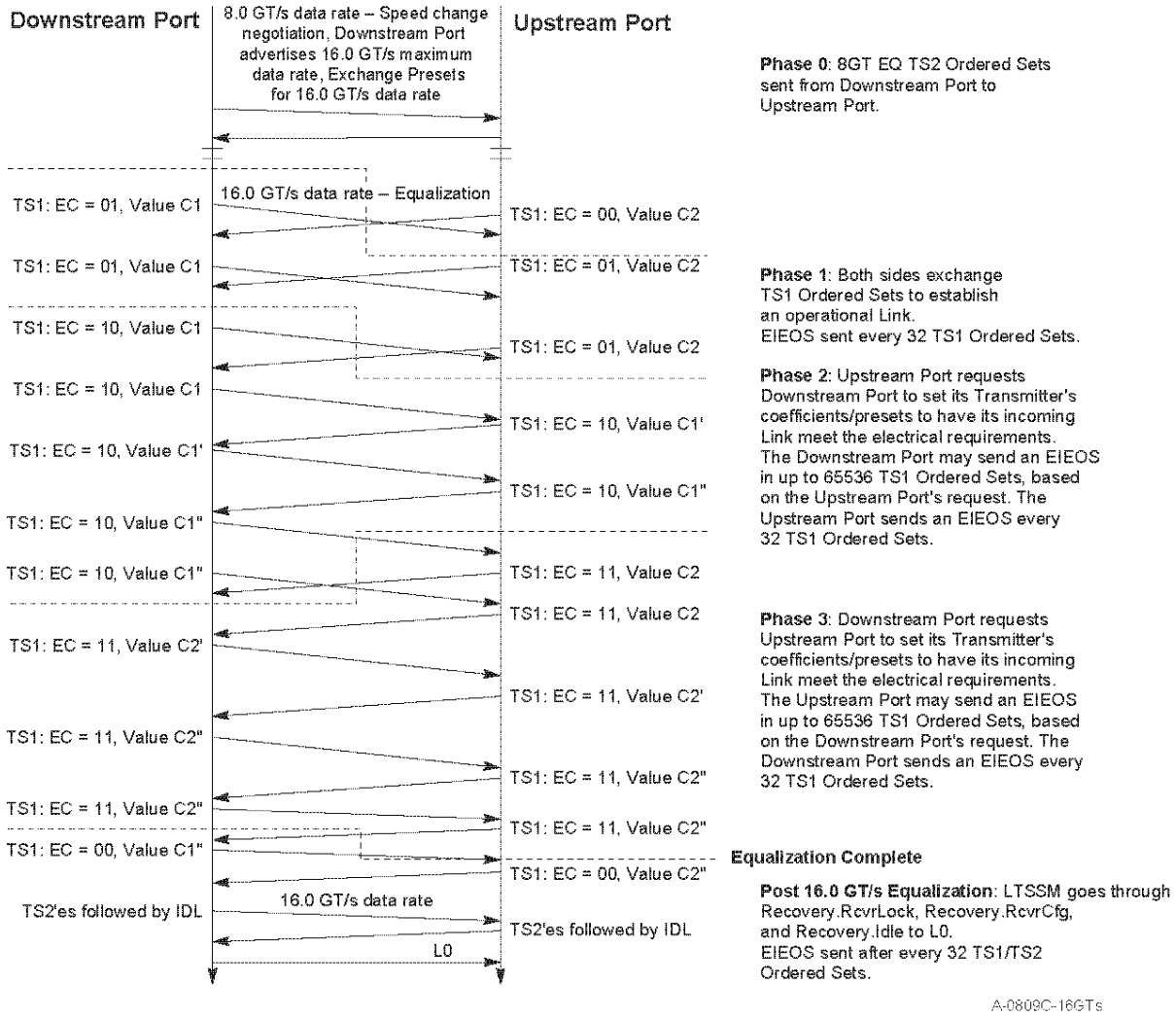


Figure 4-21: 16.0 GT/s Equalization Flow

4.2.3.1 *Rules for Transmitter Coefficients*

The explanation of the coefficients and the FIR filter it represents are provided in Section 8.3.3.1. The following rules apply to both the advertised as well as requested coefficient settings.

1. C_{-1} and C_{+1} are the coefficients used in the FIR equation and represent the pre-cursor and post-cursor, respectively. The pre-cursor and post-cursor values communicated in the TS1 Ordered Sets represent their absolute values. C_0 represents the cursor coefficient setting and is a positive entity.
2. The sum of the absolute values of the coefficients defines the FS (Full Swing; FS = $|C_{-1}|+C_0+|C_{+1}|$). FS is advertised to the Link partner in Phase 1. The Transmitter FS range is defined below:
   - FS ∈ {24, …, 63} (i.e., FS must have a value from 24 through 63) for full swing mode.
   - FS ∈ {12, …, 63} for reduced swing mode.
3. A Transmitter advertises its LF (Low Frequency) value during Phase 1. This corresponds to the minimum differential voltage that can be generated by the Transmitter which is LF/FS times the Transmitters maximum differential voltage. The Transmitter must ensure that when equation c) below equals LF it must meet the electrical requirements defined in Section 8.3.3.9 for $V_{TX\text{-}EIEOS\text{-}FS}$ and $V_{TX\text{-}EIEOS\text{-}RS}$.
4. The following rules must be satisfied before a set of coefficients can be requested of the Link partner's Transmitter. Upon reception of an update request for TX coefficient settings, a Port must verify that the new request meets the following conditions and reject the request if any of following conditions are violated:

   a) $|C_{-1}|$ <= Floor (**FS**/4)

   b) $|C_{-1}|+C_0+|C_{+1}|$ = **FS** (Do not allow peak power to change with adaptation.)

   c) $C_0-|C_{-1}|-|C_{+1}|$ >= **LF**

4.2.3.2 *Encoding of Presets*

Definition of the Transmitter and Receiver Preset Hints appears in Chapter 8. The encoding for the Transmitter Preset and Receiver Preset Hint are provided in Table 4-3 and Table 4-4. Receiver Preset Hints are optional and only defined for the 8.0 GT/s data rate.

Table 4-3: Transmitter Preset Encoding

Encoding	Preset Number in Table 4-16
0000b	P0
0001b	P1
0010b	P2
0011b	P3
0100b	P4
0101b	P5
0110b	P6
0111b	P7
1000b	P8
1001b	P9
1010b	P10
1011b through 1111b	Reserved

Table 4-4: Receiver Preset Hint Encoding for 8.0 GT/s

Encoding	Receiver Preset Value
000b	-6 dB
001b	-7 dB
010b	-8 dB
011b	-9 dB
100b	-10 dB
101b	-11 dB
110b	-12 dB
111b	Reserved

4.2.4 Link Initialization and Training

This section defines the Physical Layer control process that configures and initializes each Link for normal operation. This section covers the following features:

- configuring and initializing the Link
- supporting normal packet transfers
- supported state transitions when recovering from Link errors

- restarting a Port from low power states.

The following are discovered and determined during the training process:

- Link width
- Link data rate
- Lane reversal
- Lane polarity

Training does:

- Link data rate negotiation.
- Bit lock per Lane
- Lane polarity
- Symbol lock or Block alignment per Lane
- Lane ordering within a Link
- Link width negotiation
- Lane-to-Lane de-skew within a multi-Lane Link.

4.2.4.1 *Training Sequences*

Training sequences are composed of Ordered Sets used for initializing bit alignment, Symbol alignment and to exchange Physical Layer parameters. When the data rate is 2.5 GT/s or 5.0 GT/s, Ordered Sets are never scrambled but are always 8b/10b encoded. When the data rate is 8.0 GT/s or higher, the 128b/130b encoding is used and Symbols may or may not be scrambled, according to the rules in Section 4.2.2.4.

Training sequences (TS1 or TS2) are transmitted consecutively and can only be interrupted by SKP Ordered Sets (see Section 4.2.7) or, for data rates other than 2.5 GT/s, EIEOS Ordered Sets (see Section 4.2.4.2).

When 8.0 GT/s or higher data rates are supported, a TS1 (or TS2) Ordered Set using 8b/10b encoding (i.e., 2.5 or 5.0 GT/s data rate) can be either a standard TS1 (or TS2) Ordered Set (i.e., Symbol 6 is D10.2 for a TS1 Ordered Set or D5.2 for a TS2 Ordered Set) or an EQ TS1 (or EQ TS2) (i.e., Symbol 6 bit 7 is 1b). The ability to transmit EQ TS1 Ordered Sets is implementation-specific. Ports supporting 8.0 GT/s or higher data rates must accept either TS1 (or TS2) type in the LTSSM states unless explicitly required to look for a specific type. Ports that do not support the 8.0 GT/s data rate are permitted, but not required, to accept EQ TS1 (or TS2) Ordered Sets.

When the 16.0 GT/s data rate is supported, a TS2 using 128b/130b encoding (i.e. 8.0 or 16.0 GT/s data rate) can be either a standard TS2 Ordered Set (i.e., Symbol 7 is 45h) or an 8GT EQ TS2 (i.e., Symbol 7 bit 7 is 1b). Ports supporting the 16.0 GT/s data rate must accept either TS2 type in the LTSSM states unless explicitly required to look for a specific type. Ports that do not support the 16.0 GT/s data rate are permitted, but not required, to accept 8GT EQ TS2 Ordered Sets.

When using 8b/10b encoding, TS1 or TS2 Ordered Sets are considered consecutive only if Symbol 6 matches the Symbol 6 of the previous TS1 or TS2 Ordered Set.

When using 128b/130b encoding, TS1 or TS2 Ordered Sets are considered consecutive only if Symbols 6-9 match Symbols 6-9 of the previous TS1 or TS2 Ordered Set, with Reserved bits treated as described below.

Reserved bits in TS1 and TS2 Ordered Sets must be handled as follows:

- The Transmitter must transmit 0s for Reserved bits.
- The Receiver:
  - must not determine that a TS1 or TS2 Ordered Set is invalid based on the received value of Reserved bits
  - must use the received value of Reserved bits for the purpose of a parity computation if the Reserved bits are included in a parity calculation
  - may optionally compare the received value of Reserved bits within Symbols that are explicitly called out as being required to be identical in TS1 or TS2 Ordered Sets to determine if they are consecutive
  - must not otherwise take any functional action based on the value of any received Reserved bits

When using 128b/130b encoding, Transmitters are required to track the running DC Balance of the bits transmitted on the wire (after scrambling) that constitute the TS1 and TS2 Ordered Sets only. The running DC Balance is the difference between the number of 1s transmitted and the number of 0s transmitted. Each Lane must track its running DC Balance independently and be capable of tracking a difference of at least 511 bits in either direction: 511 more 1s than 0s, and 511 more 0s than 1s. Any counters used must saturate at their limit (not roll-over) and continue to track reductions after their limit is reached. For example, a counter that can track a difference of 511 bits will saturate at 511 if a difference of 513 is detected, and then change to 509 if the difference is reduced by 2 in the future.

The running DC Balance is set to 0 by two events: 1) The Transmitter exiting Electrical Idle; 2) Transmission of an EIEOS following a Data Block.

For every TS1 or TS2 Ordered Set transmitted, Transmitters must evaluate the running DC Balance and transmit one of the DC Balance Symbols defined for Symbols 14 and 15 as defined by the algorithm below. If the number of 1s needs to be reduced, the DC Balance Symbols 20h (for Symbol 14) and 08h (for Symbol 15) are transmitted. If the number of 0s needs to be reduced, the DC Balance Symbols DFh (for Symbol 14) and F7h (for Symbol 15) are transmitted. If no change is required, the appropriate TS1 or TS2 Identifier Symbol is transmitted. Any DC Balance Symbols transmitted for Symbols 14 or 15 bypass scrambling, while TS1 and TS2 Identifier Symbols follow the standard scrambling rules. The following algorithm must be used to control the DC Balance:

- If the running DC Balance is > 31 at the end of Symbol 11 of the TS Ordered Set, transmit DFh for Symbol 14 and F7h for Symbol 15 to reduce the number of 0s, or 20h for Symbol 14 and 08h for Symbol 15 to reduce the number of 1s.
- Else, if the running DC Balance is > 15 at the end of Symbol 11 of the TS Ordered Set, transmit F7h for Symbol 15 to reduce the number of 0s, or 08h for Symbol 15 to reduce the number of 1s. Transmit the normal TS Identifier Symbol (scrambled) for Symbol 14.
- Else, transmit the normal TS Identifier Symbol (scrambled) for Symbols 14 and 15.

Receivers are permitted, but not required, to check Symbols 14 and 15 for the following values when determining whether a TS1 or TS2 Ordered Set is valid: The appropriate TS Identifier Symbol after de-scrambling, or a valid DC Balance Symbol of DFh or 20h before de-scrambling for Symbol 14, or a valid DC Balance Symbol of F7h or 08h before de-scrambling for Symbol 15.

IMPLEMENTATION NOTE

Sync Header and DC Balance

Block Sync Header bits and the first Symbol of TS1 and TS2 Ordered Sets do not affect the running DC Balance, because they have equal number of 1s and 0s.

The Training control bits for Hot Reset, Disable Link, and Enable Loopback are mutually exclusive, only one of these bits is permitted to be set at a time as well as transmitted on all Lanes in a configured (all Lanes that were in L0) or possible (all Lanes in Configuration) Link. If more than one of the Hot Reset, Disable Link, or Enable Loopback bits are Set at the same time, the Link behavior is undefined.

The TS1 Ordered Set's Retimer Equalization Extend bit is always set to 0b when transmitted by an Upstream Port or Downstream Port. Retimers set the bit to 1b as described in Section 4.3.7.2.

Table 4-5: TS1 Ordered Set

Symbol Number	Description
0	When operating at 2.5 or 5.0 GT/s: COM (K28.5) for Symbol alignment. When operating at 8.0 GT/s or above: Encoded as 1Eh (TS1 Ordered Set).
1	Link Number. Ports that do not support 8.0 GT/s or above: 0-255, PAD. Downstream Ports that support 8.0 GT/s or above: 0-31, PAD. Upstream Ports that support 8.0 GT/s or above: 0-255, PAD. When operating at 2.5 or 5.0 GT/s: PAD is encoded as K23.7. When operating at 8.0 GT/s or above: PAD is encoded as F7h.
2	Lane Number within Link. When operating at 2.5 or 5.0 GT/s: 0-31, PAD. PAD is encoded as K23.7. When operating at 8.0 GT/s or above: 0-31, PAD. PAD is encoded as F7h.
3	N_FTS. The number of Fast Training Sequences required by the Receiver: 0-255.

Symbol Number	Description
4	Data Rate Identifier Bit 0 – Reserved for future Data Rate. Bit 1 – 2.5 GT/s Data Rate Supported. Must be set to 1b. Bit 2 – 5.0 GT/s Data Rate Supported. Must be set to 1b if Bit 3 is 1b. See Section 8.2. Bit 3 – 8.0 GT/s Data Rate Supported. Must be set to 1b if Bit 4 is 1b. Bit 4 – 16.0 GT/s Data Rate Supported. Bit 5 – Reserved for future Data Rate. Bit 6 – Autonomous Change/Selectable De-emphasis. Downstream Ports: This bit is defined for use in the following LTSSM states: Polling.Active, Configuration.Linkwidth.Start, and Loopback.Entry. In all other LTSSM states, it is Reserved. Upstream Ports: This bit is defined for use in the following LTSSM states: Polling.Active, Configuration, Recovery, and Loopback.Entry. In all other LTSSM states, it is Reserved. Bit 7 – speed_change. This bit can be set to 1b only in the Recovery.RcvrLock LTSSM state. In all other LTSSM states, it is Reserved.
5	Training Control Bit 0 – Hot Reset Bit 0 = 0b, Deassert Bit 0 = 1b, Assert Bit 1 – Disable Link Bit 1 = 0b, Deassert Bit 1 = 1b, Assert Bit 2 – Loopback Bit 2 = 0b, Deassert Bit 2 = 1b, Assert Bit 3 – Disable Scrambling in 2.5 GT/s and 5.0 GT/s data rates; Reserved in other data rates Bit 3 = 0b, Deassert Bit 3 = 1b, Assert Bit 4 – Compliance Receive Bit 4 = 0b, Deassert Bit 4 = 1b, Assert Ports that support 5.0 GT/s and above data rate(s) must implement the Compliance Receive bit. Ports that support only 2.5 GT/s data rate may optionally implement the Compliance Receive bit. If not implemented, the bit is Reserved. Bit 5:7 – Reserved

Symbol Number	Description
6	When operating at 2.5 or 5.0 GT/s: Standard TS1 Ordered Sets encode this Symbol as a TS1 Identifier, D10.2 (4Ah). EQ TS1 Ordered Sets encode this Symbol as follows: Bit 2:0 – Receiver Preset Hint. See Section 4.2.3.2. Bit 6:3 – Transmitter Preset. See Section 4.2.3.2. Bit 7 – Set to 1b. When operating at 8.0 GT/s or 16.0 GT/s: Bit 1:0 – Equalization Control (EC). This field is only used in the Recovery.Equalization and Loopback LTSSM states. See Section 4.2.6.4.2 and Section 4.2.6.10. In all other LTSSM states, it must be set to 00b. Bit 2 – Reset EIEOS Interval Count. This bit is defined for use in the Recovery.Equalization LTSSM state. See Section 4.2.6.4.2 and Section 4.2.4.2. In all other LTSSM states, it is Reserved. Bit 6:3 – Transmitter Preset. See Section 4.2.3 and Section 4.2.6. Bit 7 – Use Preset/Equalization Redo. This bit is defined for use in the Recovery.Equalization, Recovery.RcvrLock and Loopback LTSSM states. See Section 4.2.6.4.1, Section 4.2.6.4.2 and Section 4.2.6.10. In all other LTSSM states, it is Reserved.
7	When operating at 2.5 or 5.0 GT/s: TS1 Identifier. Encoded as D10.2 (4Ah). When operating at 8.0 GT/s or 16.0 GT/s: Bit 5:0 – FS when the EC field of Symbol 6 is 01b (see Section 4.2.3.1). Otherwise, Pre-cursor Coefficient for the current data rate of operation. Bit 6 – Reserved. Bit 7 – Retimer Equalization Extend. This bit is defined for use in the Recovery.Equalization LTSSM state when operating at 16.0 GT/s. In all other LTSSM states and when operating at 8.0 GT/s, it is Reserved.
8	When operating at 2.5 or 5.0 GT/s: TS1 Identifier. Encoded as D10.2 (4Ah). When operating at 8.0 GT/s or 16.0 GT/s: Bit 5:0 – LF when the EC field of Symbol 6 is 01b (see Section 4.2.3.1). Otherwise, Cursor Coefficient for the current data rate of operation. Bit 7:6 – Reserved.
9	When operating at 2.5 or 5.0 GT/s: TS1 Identifier. Encoded as D10.2 (4Ah). When operating at 8.0 GT/s or 16.0 GT/s: Bit 5:0 – Post-cursor Coefficient for the current data rate of operation. Bit 6 – Reject Coefficient Values. This bit can only be set to 1b in specific Phases of the Recovery.Equalization LTSSM State. See Section 4.2.6.4.2. In all other LTSSM states, it must be set to 0b. Bit 7 – Parity (P). This bit is the even parity of all bits of Symbols 6, 7, and 8 and bits 6:0 of Symbol 9. Receivers must calculate the parity of the received bits and compare it to the received Parity bit. Received TS1 Ordered Sets are valid only if the calculated and received Parity match.

Symbol Number	Description
10 – 13	When operating at 2.5 or 5.0 GT/s: TS1 Identifier. Encoded as D10.2 (4Ah). When operating at 8.0 GT/s or above: TS1 Identifier. Encoded as 4Ah.
14 – 15	When operating at 2.5 or 5.0 GT/s: TS1 Identifier. Encoded as D10.2 (4Ah). When operating at 8.0 GT/s or above: TS1 Identifier (encoded as 4Ah) or a DC Balance Symbol.

Table 4-6: TS2 Ordered Set

Symbol Number	Description
0	When operating at 2.5 or 5.0 GT/s: COM (K28.5) for Symbol alignment. When operating at 8.0 GT/s or above: Encoded as 2Dh (TS2 Ordered Set).
1	Link Number. Ports that do not support 8.0 GT/s or above: 0-255, PAD. Downstream Ports that support 8.0 GT/s or above: 0-31, PAD. Upstream Ports that support 8.0 GT/s or above: 0-255, PAD. When operating at 2.5 or 5.0 GT/s: PAD is encoded as K23.7. When operating at 8.0 GT/s or above: PAD is encoded as F7h.
2	Lane Number within Link. When operating at 2.5 or 5.0 GT/s: 0-31, PAD. PAD is encoded as K23.7. When operating at 8.0 GT/s or above: 0-31, PAD. PAD is encoded as F7h.
3	N_FTS. The number of Fast Training Sequences required by the Receiver: 0-255.
4	Data Rate Identifier Bit 0 – Reserved for future Data Rate Bit 1 – 2.5 GT/s Data Rate Supported. Must be set to 1b. Bit 2 – 5.0 GT/s Data Rate Supported. Must be set to 1b if Bit 3 is 1b. See Section 8.2. Bit 3 – 8.0 GT/s Data Rate Supported. Must be set to 1b if Bit 4 is 1b. Bit 4 – 16.0 GT/s Data Rate Supported Bit 5 – Reserved for future Data Rate Bit 6 – Autonomous Change/Selectable De-emphasis/Link Upconfigure Capability. This bit is defined for use in the following LTSSM states: Polling.Configuration, Configuration.Complete, and Recovery. In all other LTSSM states, it is Reserved. Bit 7 – speed_change. This bit can be set to 1b only in the Recovery.RcvrCfg LTSSM state. In all other LTSSM states, it is Reserved.

Symbol Number	Description
5	Training Control Bit 0 – Hot Reset Bit 0 = 0b, Deassert Bit 0 = 1b, Assert Bit 1 – Disable Link Bit 1 = 0b, Deassert Bit 1 = 1b, Assert Bit 2 – Loopback Bit 2 = 0b, Deassert Bit 2 = 1b, Assert Bit 3 – Disable Scrambling in 2.5 GT/s and 5.0 GT/s data rates; Reserved in other data rates Bit 3 = 0b, Deassert Bit 3 = 1b, Assert Bit 4 – Retimer Present in 2.5 GT/s data rate. Reserved in other data rates. Bit 4 = 0b, No Retimers present Bit 4 = 1b, One or more Retimers present Bit 5 – Two Retimers Present in 2.5 GT/s data rate. Reserved in other data rates. Ports that support 16.0 GT/s data rate or higher must implement this bit. Ports that support only 8.0 GT/s data rate or lower are permitted to implement this bit. Bit 5 = 0b, Zero or one Retimers present Bit 5 = 1b, Two or more Retimers present Bit 6:7 – Reserved
6	When operating at 2.5 or 5.0 GT/s: Standard TS2 Ordered Sets encode this Symbol as a TS2 Identifier, D5.2 (45h). EQ TS2 Ordered Sets encode this Symbol as follows: Bit 2:0 – 8.0 GT/ s Receiver Preset Hint. See Section 4.2.3.2. Bit 6:3 – 8.0 GT/ s Transmitter Preset. See Section 4.2.3.2. Bit 7 –Set to 1b. When operating at 8.0 GT/s or 16.0 GT/ s: Bit 4:0 – Reserved. Bit 5 – Equalization Request Data Rate. A value of 0b indicates 8.0 GT/s Data Rate and 1b indicates 16.0 GT/s Data Rate. This bit is defined for use in the Recovery.RcvrCfg LTSSM state. In all other LTSSM states, it is Reserved. Bit 6 – Quiesce Guarantee. This bit is defined for use in the Recovery.RcvrCfg LTSSM state. In all other LTSSM states, it is Reserved. Bit 7 – Request Equalization. This bit is defined for use in the Recovery.RcvrCfg LTSSM state. In all other LTSSM states, it is Reserved.

Symbol Number	Description
7	When operating at 2.5 or 5.0 GT/s: TS2 Identifier. Encoded as D5.2 (45h). When operating at 8.0 GT/s or above: Standard TS2 Ordered Sets encode this Symbol as a TS2 Identifier, 45h. 8GT EQ TS2 Ordered Sets encode this Symbol as follows: Bit 2:0 – Reserved Bit 6:3 – 16.0 GT/s Transmitter Preset. See Section 4.2.3.2. Bit 7 – Set to 1b. This definition is only valid in the Recovery.RcvrCfg LTSSM state when Preset values are being communicated.
8 – 13	When operating at 2.5 or 5.0 GT/s: TS2 Identifier. Encoded as D5.2 (45h). When operating at 8.0 GT/s or above: TS2 Identifier. Encoded as 45h.
14-15	When operating at 2.5 or 5.0 GT/s: TS2 Identifier. Encoded as D5.2 (45h). When operating at 8.0 GT/s or above: TS2 Identifier (encoded as 45h) or a DC Balance Symbol.

4.2.4.2 *Electrical Idle Sequences*

Before a Transmitter enters Electrical Idle, it must always send an Electrical Idle Ordered Set Sequence (EIOSQ), unless otherwise specified. An Electrical Idle Ordered Set Sequence (EIOSQ) is defined as one EIOS if the current Data Rate is 2.5 GT/s, 8.0 GT/s or 16.0 GT/s Data Rate, or two consecutive EIOSs if the current Data Rate is 5.0 GT/s.

When using 8b/10b encoding, an EIOS is a K28.5 (COM) followed by three K28.3 (IDL) Symbols. Transmitters must transmit all Symbols of an EIOS. An EIOS is received when the COM and two of the three IDL Symbols are received. When using 128b/130b encoding, an EIOS is an Ordered Set block, as defined in Table 4-8. Transmitters must transmit all Symbols of an EIOS if additional EIOSs are to be transmitted following it. Transmitters must transmit Symbols 0-13 of an EIOS, but are permitted to terminate the EIOS anywhere in Symbols 14 or 15, when transitioning to Electrical Idle after it. An EIOS is considered received when Symbols 0-3 of an Ordered Set Block match the definition of an EIOS.

IMPLEMENTATION NOTE

Truncation of EIOS Ordered Set

Truncation in the last EIOS is allowed to help implementations where a transmitter may terminate on an internal clock boundary that may not align on a Symbol boundary due to 128b/130b encoding. Truncation is okay since Receivers will just look at the first four Symbols to conclude it is an EIOS.

After transmitting the last Symbol of the last Electrical Idle Ordered Set, the Transmitter must be in a valid Electrical Idle state as specified by $T_{TX\text{-}IDLE\text{-}SET\text{-}TO\text{-}IDLE}$ (see Table 8-8).

Table 4-7: Electrical Idle Ordered Set (EIOS) for 2.5 GT/s and 5.0 GT/s Data Rates

Symbol Number	Encoded Values	Description
0	K28.5	COM for Symbol alignment
1	K28.3	IDL
2	K28.3	IDL
3	K28.3	IDL

Table 4-8: Electrical Idle Ordered Set (EIOS) for 8.0 GT/s and Above Data Rates

Symbol Numbers	Value	Description
0-15	66h	EIOS Identifier and Payload

Table 4-9: Electrical Idle Exit Ordered Set (EIEOS) for 5.0 GT/s Data Rate

Symbol Number	Encoded Values	Description
0	K28.5	COM for Symbol alignment
1-14	K28.7	EIE – K Symbol with low frequency components for helping achieve exit from Electrical Idle
15	D10.2	TS1 Identifier (See Note 1)

Notes:

1. This symbol is not scrambled. Previous versions of this specification were less clear and some implementations may have incorrectly scrambled this symbol. It is recommended that devices be tolerant of receiving EIEOS in which this symbol is scrambled.

Table 4-10: Electrical Idle Exit Ordered Set (EIEOS) for 8.0 GT/s Data Rates

Symbol Numbers	Value	Description
0, 2, 4, 6, 8, 10, 12, 14	00h	Symbol 0: EIEOS Identifier A low frequency pattern that alternates between eight 0s and eight 1s.
1, 3, 5, 7, 9, 11, 13, 15	FFh	A low frequency pattern that alternates between eight 0s and eight 1s.

Table 4-11: Electrical Idle Exit Ordered Set (EIEOS) for 16.0 GT/s Data Rate

Symbol Numbers	Value	Description
0, 1, 4, 5, 8, 9, 12, 13	00h	Symbol 0: EIEOS Identifier A low frequency pattern that alternates between sixteen 0s and sixteen 1s.
2, 3, 6, 7, 10, 11, 14, 15	FFh	A low frequency pattern that alternates between sixteen 0s and sixteen 1s.

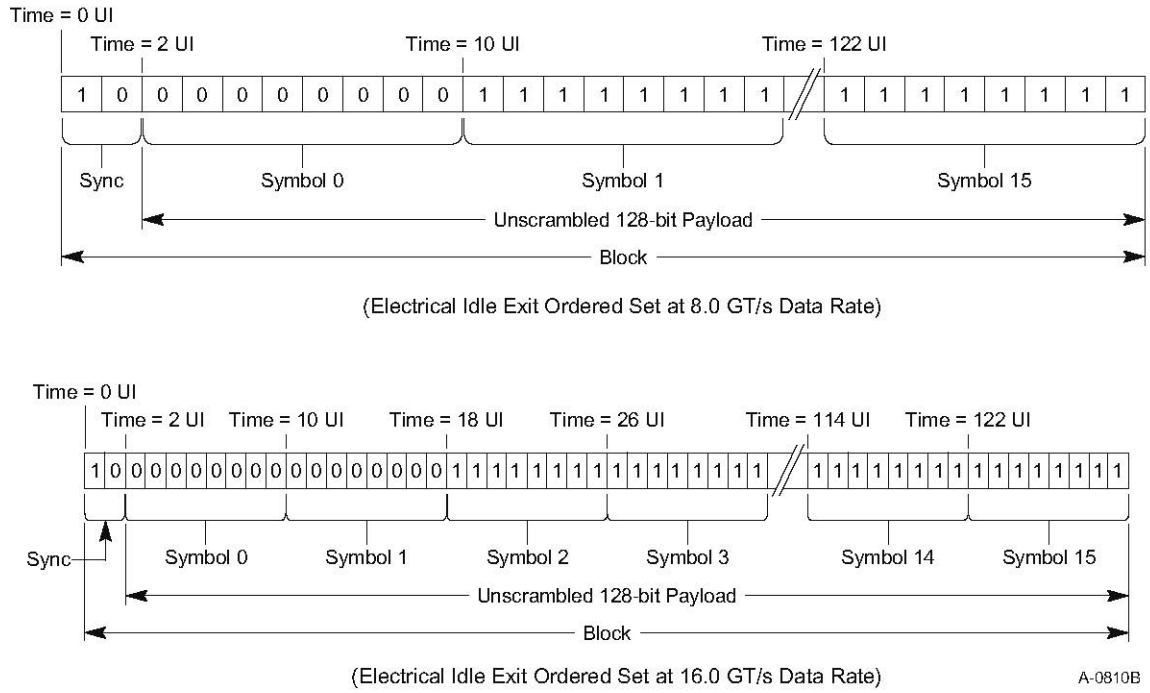


Figure 4-22: Electrical Idle Exit Ordered Set for 8.0 GT/s and Above Data Rates

The Electrical Idle Exit Ordered Set (EIEOS) is transmitted only when operating at speeds other than 2.5 GT/s. It is a low frequency pattern transmitted periodically to help ensure that receiver Electrical Idle exit circuitry can detect an exit from Electrical Idle. When using 128b/130b encoding, it is also used for Block Alignment as described in Section 4.2.2.2.1.

When using 8b/10b encoding and operating at 5.0 GT/s, an EIEOS, as defined in Table 4-9, is transmitted in the following situations:

- Before the first TS1 Ordered Set after entering the LTSSM Configuration.Linkwidth.Start state.
- Before the first TS1 Ordered Set after entering the LTSSM Recovery.RcvrLock state.
- After every 32 TS1 or TS2 Ordered Sets are transmitted in the LTSSM Configuration.Linkwidth.Start, Recovery.RcvrLock, and Recovery.RcvrCfg states. The TS1/TS2 count is set to 0 when:
  - An EIEOS is transmitted.
  - The first TS2 Ordered Set is received while in the LTSSM Recovery.RcvrCfg state.

When using 128b/130b encoding, an EIEOS, as defined in Table 4-10 and Figure 4-22, is transmitted in the following situations:

- Before the first TS1 Ordered Set after entering the LTSSM Configuration.Linkwidth.Start substate.
- Before the first TS1 Ordered Set after entering the LTSSM Recovery.RcvrLock substate.

- Immediately following an EDS Framing Token when ending a Data Stream and not transmitting an EIOS and not entering the LTSSM Recovery.RcvrLock substate.
- After every 32 TS1 or TS2 Ordered Sets are transmitted in all LTSSM states which require transmission of TS1 or TS2 Ordered Sets. The TS1/TS2 count is set to 0 when:
  - An EIEOS is transmitted.
  - The first TS2 Ordered Set is received while in the LTSSM Recovery.RcvrCfg state.
  - The first TS2 Ordered Set is received while in the LTSSM Configuration.Complete state.
  - A Downstream Port is in Phase 2 of the LTSSM Recovery.Equalization state and two consecutive TS1 Ordered Sets are received on any Lane with the Reset EIEOS Interval Count bit set.
  - An Upstream Port is in Phase 3 of the LTSSM Recovery.Equalization state and two consecutive TS1 Ordered Sets are received on any Lane with the Reset EIEOS Interval Count bit set.
- After every 65,536 TS1 Ordered Sets are transmitted in the LTSSM Recovery.Equalization state if the Reset EIEOS Interval Count bit has prevented it from being transmitted for that interval. Implementations are permitted to satisfy this requirement by transmitting an EIEOS within two TS1 Ordered Sets of when the scrambling LFSR matches its seed value.
- As part of an FTS Ordered Set, Compliance Pattern, or Modified Compliance pattern as described in the relevant sections.

Example: An LTSSM enters Recovery.RcvrLock from L0 in 5.0 GT/s data rate. It transmits an EIEOS followed by TS1 Ordered Sets. It transmits 32 TS1 Ordered Sets following which it transmits the second EIEOS. Subsequently it sends two more TS1 Ordered Sets and enters Recovery.RcvrCfg where it transmits the third EIEOS after transmitting 30 TS2 Ordered Sets. It transmits 31 more TS2 Ordered Sets (after the first 30 TS2 Ordered Sets) in Recovery.RcvrCfg when it receives a TS2 Ordered Set. Since it receives its first TS2 Ordered Set, it will reset its EIEOS interval count to 0 and keep transmitting another 16 TS2 Ordered Sets before transitioning to Recovery.Idle. Thus, it did not send an EIEOS in the midst of the last 47 TS2 Ordered Sets since the EIEOS interval count got reset to 0b. From Recovery.Idle, the LTSSM transitions to Configuration.Linkwidth.Start and transmits an EIEOS after which it starts transmitting the TS1 Ordered Sets.

While operating in speeds other than 2.5 GT/s, an implementation is permitted to not rely on the output of the Electrical Idle detection circuitry except when receiving the EIEOS during certain LTSSM states or during the receipt of the FTS prepended by the four consecutive EIE Symbols (see Section 4.2.4.5) at the Receiver during Rx L0s or the Modified Compliance Pattern in Polling.Compliance when the circuitry is required to signal an exit from Electrical Idle.

4.2.4.3 *Inferring Electrical Idle*

A device is permitted in all speeds of operation to infer Electrical Idle instead of detecting Electrical Idle using analog circuitry. Table 4-12 summarizes the conditions to infer Electrical Idle in the various substates.

Table 4-12: Electrical Idle Inference Conditions

State	2.5 GT/s	5.0 GT/s	8.0 GT/s	16.0 GT/s
L0	Absence of Flow Control Update DLLP[46] or alternatively a SKP Ordered Set in a 128 µs window	Absence of Flow Control Update DLLP[46] or alternatively a SKP Ordered Set in a 128 µs window	Absence of Flow Control Update DLLP[46] or alternatively a SKP Ordered Set in a 128 µs window	Absence of Flow Control Update DLLP[46] or alternatively a SKP Ordered Set in a 128 µs window
Recovery.RcvrCfg	Absence of a TS1 or TS2 Ordered Set in a 1280 UI interval	Absence of a TS1 or TS2 Ordered Set in a 1280 UI interval	Absence of a TS1 or TS2 Ordered Set in a 4 ms window	Absence of a TS1 or TS2 Ordered Set in a 4 ms window
Recovery.Speed when successful_speed_negotiation = 1b	Absence of a TS1 or TS2 Ordered Set in a 1280 UI interval	Absence of a TS1 or TS2 Ordered Set in a 1280 UI interval	Absence of a TS1 or TS2 Ordered Set in a 4680 UI interval	Absence of a TS1 or TS2 Ordered Set in a 4680 UI interval
Recovery.Speed when successful_speed_negotiation = 0b	Absence of an exit from Electrical Idle in a 2000 UI interval	Absence of an exit from Electrical Idle in a 16000 UI interval	Absence of an exit from Electrical Idle in a 16000 UI interval	Absence of an exit from Electrical Idle in a 16000 UI interval
Loopback.Active (as slave)	Absence of an exit from Electrical Idle in a 128 µs window	N/A	N/A	N/A

The Electrical Idle exit condition must not be determined based on inference of Electrical Idle condition. For area efficiency, an implementation is permitted to choose to implement a common timeout counter per LTSSM and look for the Electrical Idle inference condition within the common timeout window determined by the common counter for each of the Lanes the LTSSM controls instead of having a timeout counter per Lane.

[46] A Flow Control Update DLLP is either an UpdateFC as defined in this specification or an MRUpdateFC as defined in the *Multi-Root I/O Virtualization and Sharing Specification* (MR-IOV).

IMPLEMENTATION NOTE

Inference of Electrical Idle

In the L0 state, one or more Flow Control Update DLLPs are expected to be received in a 128 µs window. Also in L0, one or more SKP Ordered Sets are expected to be received in a 128 µs window. As a simplification, it is permitted to use either one (or both) of these indicators to infer Electrical Idle. Hence, the absence of a Flow Control Update DLLP and/or a SKP Ordered Set in any 128 µs window can be inferred as Electrical Idle. In Recovery.RcvrCfg as well as Recovery.Speed with successful speed negotiation, the Receiver should receive TS1 or TS2 Ordered Sets continuously with the exception of the EIEOS and the SKP Ordered Set. Hence, the absence of a TS1 or TS2 Ordered Set in the interval specified above must be treated as Electrical Idle for components that implement the inference mechanism. In the event that the device enters Recovery.Speed with successful_speed_negotiation = 0b, there is a possibility that the device had failed to receive Symbols. Hence, the Electrical Idle inference is done as an absence of exit from Electrical Idle. In data rates other than 2.5 GT/s, Electrical Idle exit is guaranteed only on receipt of an EIEOS. Hence, the window is set to 16000 UI for detecting an exit from Electrical Idle in 5.0 GT/s and above data rates. In 2.5 GT/s data rate, Electrical Idle exit must be detected with every Symbol received. Hence, absence of Electrical Idle exit in a 2000 UI window constitutes an Electrical Idle condition.

4.2.4.4 *Lane Polarity Inversion*

During the training sequence in Polling, the Receiver looks at Symbols 6-15 of the TS1 and TS2 Ordered Sets as the indicator of Lane polarity inversion (D+ and D- are swapped). If Lane polarity inversion occurs, the TS1 Symbols 6-15 received will be D21.5 as opposed to the expected D10.2. Similarly, if Lane polarity inversion occurs, Symbols 6-15 of the TS2 Ordered Set will be D26.5 as opposed to the expected D5.2. This provides the clear indication of Lane polarity inversion.

If polarity inversion is detected the Receiver must invert the received data. The Transmitter must never invert the transmitted data. Support for Lane Polarity Inversion is required on all PCI Express Receivers across all Lanes independently.

4.2.4.5 *Fast Training Sequence (FTS)*

Fast Training Sequence (FTS) is the mechanism that is used for bit and Symbol lock when transitioning from L0s to L0. The FTS is used by the Receiver to detect the exit from Electrical Idle and align the Receiver's bit and Symbol receive circuitry to the incoming data. Refer to Section 4.2.5 for a description of L0 and L0s.

- At 2.5 GT/s and 5.0 GT/s data rates:
  A single FTS is comprised of one K28.5 (COM) Symbol followed by three K28.1 Symbols. The maximum number of FTSs (N_FTS) that a component can request is 255, providing a bit time lock of 4 * 255 * 10 * UI. If the data rate is 5.0 GT/s, four consecutive EIE Symbols are transmitted at valid signal levels prior to transmitting the first FTS. These Symbols will help the Receiver detect exit from Electrical Idle. An implementation that does not guarantee proper signaling levels for up to the allowable time on the Transmitter pins (see Section 4.2.4.5) since exiting Electrical Idle condition is required to prepend its first FTS by extra EIE Symbols so that the Receiver can receive at least four EIE Symbols at valid signal levels. Implementations must not transmit more than eight EIE Symbols prior to transmitting the first FTS. A component is permitted to advertise different N_FTS rates at different speeds. At 5.0 GT/s, a component may choose to advertise an appropriate N_FTS number considering that it will receive the four EIE Symbols. 4096 FTSs must be sent when the Extended Synch bit is set in order to provide external Link monitoring tools with enough time to achieve bit and framing synchronization. SKP Ordered Sets must be scheduled and transmitted between FTSs as necessary to meet the definitions in Section 4.2.7 with the exception that no SKP Ordered Sets can be transmitted during the first N_FTS FTSs. A single SKP Ordered Set is always sent after the last FTS is transmitted. It is permitted for this SKP Ordered Set to affect or not affect the scheduling of subsequent SKP Ordered Sets for Clock Tolerance Compensation by the Transmitter as described in Section 4.2.7. Note that it is possible that two SKP Ordered Sets can be transmitted back to back (one SKP Ordered Set to signify the completion of the 4096 FTSs and one scheduled and transmitted to meet the definitions described in Section 4.2.7).

- At 8.0 GT/s and above data rates:
  A single FTS is a 130-bit unscrambled Ordered Set Block, as shown in Table 4-13. The maximum number of FTSs (N_FTS) that a component can request is 255, providing a bit time lock of 130 * 255 UI (130 * 263 UI if including the periodic EIEOS). A component is permitted to advertise different N_FTS values at different speeds. On exit from L0s, the transmitter first transmits an EIEOS block which will help the receiver detect exit from Electrical Idle due to its low frequency content. After that first EIEOS, the transmitter must send the required number of FTS (4096 when the Extended Synch bit is Set; otherwise N_FTS), with an EIEOS transmitted after every 32 FTS. The FTS sequence will enable the receiver obtain bit lock (and optionally to do Block alignment). When the Extended Synch bit is Set, SKP Ordered Sets must be scheduled and transmitted between FTSs and EIEOS as necessary to meet the definitions in Section 4.2.7. The last FTS Ordered Set of the FTS sequence, if any (no FTS Ordered Sets are sent if N_FTS is equal to zero), is followed by a final EIEOS that will help the receiver acquire Block alignment. Implementations are permitted to send two EIEOS back to back following the last FTS Ordered Set if the N_FTS is a multiple of 32. The EIEOS resets the scrambler in both the Transmitter as well as the Receiver. Following the final EIEOS, an SDS Ordered Set is transmitted to help the receiver perform de-skew and to indicate the transition from Ordered

Sets to Data Stream. After the SDS Ordered Set is transmitted, a Data Block must be transmitted.

IMPLEMENTATION NOTE

Scrambling LFSR During FTS Transmission in 128b/130b Encoding

Since the scrambler is reset on the last EIEOS, and none of the ordered set in the FTS sequence is scrambled, it does not matter whether implementations choose to advance the scrambler or not during the time FTS is received.

Table 4-13: FTS for 8.0 GT/s and Above Data Rates

Symbol Number	Value
0	55h
1	47h
2	4Eh
3	C7h
4	CCh
5	C6h
6	C9h
7	25h
8	6Eh
9	ECh
10	88h
11	7Fh
12	80h
13	8Dh
14	8Bh
15	8Eh

N_FTS defines the number of FTSs that must be transmitted when transitioning from L0s to L0. At the 2.5 GT/s data rate, the value that can be requested by a component corresponds to a Symbol lock time of 16 ns (N_FTS set to 0b and one SKP Ordered Set) to ~4 μs (N_FTS set to 255), except when the Extended Synch bit is Set, which requires the transmission of 4096 FTSs resulting in a bit lock time of 64 μs. For 8.0 GT/s and above data rates, when the Extended Synch bit is Set, the transmitter is required to send 4096 FTS Ordered Set Blocks. Note that the N_FTS value reported by a component may change; for example, due to software modifying the value in the Common Clock Configuration bit (Section 7.5.3.7).

If the N_FTS period of time expires before the Receiver obtains bit lock, Symbol lock or Block alignment, and Lane-to-Lane de-skew on all Lanes of the configured Link, the Receiver must transition to the Recovery state. This sequence is detailed in the LTSSM in Section 4.2.5.

4.2.4.6 *Start of Data Stream Ordered Set*

The Start of Data Stream (SDS) Ordered Set, described in Table 4-14, is defined only for 128b/130b encoding. It is transmitted in the Configuration.Idle, Recovery.Idle, and Tx_L0s.FTS LTSSM states to define the transition from Ordered Set Blocks to a Data Stream, and Loopback Masters are permitted to transmit it as described in Section 4.2.2.5. It must not be transmitted at any other time. While not in the Loopback state, the Block following an SDS Ordered Set must be a Data Block, and the first Symbol of that Data Block is the first Symbol of the Data Stream.

Table 4-14: SDS Ordered Set (for 8.0 GT/s and Above Data Rate)

Symbol Number	Value	Description
0	E1	SDS Ordered Set Identifier
1-15	55h	Body of SDS Ordered Set

4.2.4.7 *Link Error Recovery*

- Link Errors, when operating with 8b/10b encoding are:
  - 8b/10b decode errors, Framing Errors, loss of Symbol lock, Elasticity Buffer Overflow/Underflow, or loss of Lane-to-Lane de-skew.
  - 8b/10b decode errors must be checked and trigger a Receiver Error in specified LTSSM states (see Table 4-15), which is a reported error associated with the Port (see Section 6.2). Triggering a Receiver Error on any or all of Framing Error, Loss of Symbol Lock, Lane De-skew Error, and Elasticity Buffer Overflow/Underflow is optional.
- Link Errors, when operating with 128b/130b encoding, are:
  - Framing Errors, loss of Block Alignment, Elasticity Buffer Overflow/Underflow, or loss of Lane-to-Lane de-skew.
  - Framing errors must be checked and trigger a Receiver Error in the LTSSM states specified in Table 4-15. The Receiver Error is a reported error associated with the Port (see Section 6.2). Triggering a Receiver Error on any of all of loss of Block Alignment, Elasticity Buffer Overflow/Underflow, and loss of Lane-to-Lane de-skew is optional.
- On a configured Link, which is in L0, error recovery will at a minimum be managed in a Layer above the Physical Layer (as described in Section 3.6) by directing the Link to transition to Recovery.
  - Note: Link Errors may also result in the Physical Layer initiating a LTSSM state transition from L0 to Recovery.

- All LTSSM states other than L0 make progress[47] when Link Errors occur.
  - When operating with 8b/10b encoding, Link Errors that occur in LTSSM states other than L0 must not result in the Physical Layer initiating an LTSSM state transition.
  - When operating with 128b/130b encoding and not processing a Data Stream, Link Errors that occur in LTSSM states other than L0 must not result in the Physical Layer initiating an LTSSM state transition.
- When operating with 8b/10b encoding, if a Lane detects an implementation specific number of 8b/10b errors, Symbol lock must be verified or re-established as soon as possible.[48]

4.2.4.8 *Reset*

Reset is described from a system point of view in Section 6.6.

4.2.4.8.1 Fundamental Reset

- Fundamental Reset applies only when Main power is present.
- Fundamental Reset does not apply when no power or Aux power is present.

When Fundamental Reset is asserted:

- The Receiver terminations are required to meet $Z_{RX\text{-}HIGH\text{-}IMP\text{-}DC\text{-}POS}$ and $Z_{RX\text{-}HIGH\text{-}IMP\text{-}DC\text{-}NEG}$ (see Table 8-11).
- The Transmitter is required only to meet $I_{TX\text{-}SHORT}$ (see Table 8-3).
- The Transmitter holds a constant DC common mode voltage.[49]

When Fundamental Reset is deasserted:

- The Port LTSSM (see Section 4.2.5) is initialized (see Section 6.6.1 for additional requirements).

4.2.4.8.2 Hot Reset

Hot Reset is a protocol reset defined in Section 4.2.5.11.

---

[47] In this context, progress is defined as the LTSSM not remaining indefinitely in one state with the possible exception of Detect, Hot Reset, or Disabled.

[48] The method to verify and re-establish Symbol lock is implementation specific.

[49] The common mode being driven is not required to meet the Absolute Delta Between DC Common Mode during L0 and Electrical Idle ($V_{TX\text{-}CM\text{-}DC\text{-}ACTIVE\text{-}IDLE\text{-}DELTA}$) specification (see Table 8-7).

4.2.4.9 *Link Data Rate Negotiation*

All devices are required to start Link initialization using a 2.5 GT/s data rate on each Lane. A field in the training sequence Ordered Set (see Section 4.2.4) is used to advertise all supported data rates. The Link trains to L0 initially in 2.5 GT/s data rate after which a data rate change occurs by going through the Recovery state.

4.2.4.10 *Link Width and Lane Sequence Negotiation*

PCI Express Links must consist of 1, 2, 4, 8, 12, 16, or 32 Lanes in parallel, referred to as x1, x2, x4, x8, x12, x16, and x32 Links, respectively. All Lanes within a Link must simultaneously transmit data based on the same frequency with a skew between Lanes not to exceed $L_{TX\text{-}SKEW}$ (Table 8-11). The negotiation process is described as a sequence of steps.

The negotiation establishes values for Link number and Lane number for each Lane that is part of a valid Link; each Lane that is not part of a valid Link exits the negotiation to become a separate Link or remains in Electrical Idle.

During Link width and Lane number negotiation, the two communicating Ports must accommodate the maximum allowed Lane-to-Lane skew as specified by $L_{RX\text{-}SKEW}$ in Table 8-11.

Optional Link negotiation behaviors include Lane reversal, variable width Links, splitting of Ports into multiple Links and the configuration of a crosslink.

Other specifications may impose other rules and restrictions that must be comprehended by components compliant to those other specifications; it is the intent of this specification to comprehend interoperability for a broad range of component capabilities.

4.2.4.10.1 Required and Optional Port Behavior

- The ability for a xN Port to form a xN Link as well as a x1 Link (where N can be 32, 16, 12, 8, 4, 2, and 1) is required.
  - Designers must connect Ports between two different components in a way that allows those components to meet the above requirement. If the Ports between components are connected in ways that are not consistent with intended usage as defined by the component's Port descriptions/data sheets, behavior is undefined.
- The ability for a xN Port to form any Link width between N and 1 is optional.
  - An example of this behavior includes a x16 Port which can only configure into only one Link, but the width of the Link can be configured to be x12, x8, x4, x2 as well as the required widths of x16 and x1.
- The ability to split a Port into two or more Links is optional.
  - An example of this behavior would be a x16 Port that may be able to configure two x8 Links, four x4 Links, or 16 x1 Links.

- Support for Lane reversal is optional.
  - If implemented, Lane reversal must be done for both the Transmitter and Receiver of a given Port for a multi-Lane Link.
  - An example of Lane reversal consists of Lane 0 of an Upstream Port attached to Lane N-1 of a Downstream Port where either the Downstream or Upstream device may reverse the Lane order to configure a xN Link.

Support for formation of a crosslink is optional. In this context, a Downstream Port connected to a Downstream Port or an Upstream Port connected to an Upstream Port is a crosslink.

Current and future electromechanical and/or form factor specifications may require the implementation of some optional features listed above. Component designers must read the specifications for the systems that the component(s) they are designing will used in to ensure compliance to those specifications.

4.2.4.11 *Lane-to-Lane De-skew*

The Receiver must compensate for the allowable skew between all Lanes within a multi-Lane Link (see Table 8-3 and Table 8-11) before delivering the data and control to the Data Link Layer.

When using 8b/10b encoding, an unambiguous Lane-to-Lane de-skew mechanism may use one or more of the following:

- The COM Symbol of a received TS1 or TS2 Ordered Set
- The COM Symbol of a received Electrical Idle Exit Ordered Set
- The COM Symbol of the first received SKP Ordered Set after an FTS sequence
- The COM Symbol of a received SKP Ordered Set during a training sequence when not using SRIS.

When using 128b/130b encoding, an unambiguous Lane-to-Lane de-skew mechanism may use one or more of the following:

- A received SDS Ordered Set
- A received Electrical Idle Exit Ordered Set except when exiting L0s
- The first received Electrical Idle Exit Ordered Set after an FTS Ordered Set when exiting L0s
- When operating at 8.0 GT/s, a received SKP Ordered Set
- When operating at 16.0 GT/s, the first received SKP Ordered Set after an FTS sequence
- When operating at 16.0 GT/s, a received SKP Ordered Set except when:
  - exiting a training sequence or
  - two SKP Ordered Sets are separated by an EDS

Other de-skew mechanisms may also be employed, provided they are unambiguous. Lane-to-Lane de-skew must be performed during Configuration, Recovery, and L0s in the LTSSM.

IMPLEMENTATION NOTE

Unambiguous Lane-to-Lane De-Skew:

The max skew at 2.5 GT/s that a receiver must be able to de-skew is 20 ns. A nominal SKP Ordered Set, i.e. one that does not have SKP Symbols added or removed by a Retimer, is 4 Symbols long, or 16 ns, at 2.5 GT/s. Generally SKP Ordered Sets are transmitted such that they are well spaced out, and no particular care is needed to use them for de-skew, i.e. they provide an unambiguous mechanism. If back-to-back SKP Ordered Sets are transmitted, an implementation that simply looks for the COM of the SKP Ordered Set to occur on each Lane at the same point in time may fail. When exiting L0s a transmitter may send back-to-back SKP Ordered Sets after the last FTS Ordered Set of the Fast Training Sequence. De-skew must be obtained in L0s, therefore the implementation must comprehend back-to-back SKP Ordered Sets when performing de-skew in this case.

Exceptions to the unambiguous mechanism in Section 4.2.4.11, occur because back-to-back Ordered Sets might be sent, i.e. EIEOS might be sent back-to-back when exiting L0s when using 128b/130b encoding. EIEOS can still be used for de-skew in this case, however the implementation must comprehend back-to-back EIEOS when performing de-skew.

When operating at 16.0 GT/s, a transmitter may send back-to-back SKP Ordered Sets at the end of a Training Sequence, e.g., TS2 Ordered Set, SKP Ordered Set, SKP Ordered Set, SDS Ordered Set. Implementations that choose to use SKP Ordered Sets for de-skew in this case are recommended to recognize that the back-to-back SKP Ordered Sets are different, i.e. Standard SKP Ordered Set followed by Control SKP Ordered Set.

4.2.4.12 *Lane vs. Link Training*

The Link initialization process builds unassociated Lanes of a Port into associated Lanes that form a Link. For Lanes to configure properly into a desired Link, the TS1 and TS2 Ordered Sets must have the appropriate fields (Symbol 3, 4, and 5) set to the same values on all Lanes.

Links are formed at the conclusion of Configuration.

- If the optional behavior of a Port being able to configure multiple Links is employed, the following observations can be made:
  - A separate LTSSM is needed for each separate Link that is desired to be configured by any given Port.
  - The LTSSM Rules are written for configuring one Link. The decision to configure Links in a serial fashion or parallel is implementation specific.

4.2.5 Link Training and Status State Machine (LTSSM) Descriptions

The LTSSM states are illustrated in Figure 4-23. These states are described in following sections.

All timeout values specified for the Link Training and Status state machine (LTSSM) are minus 0 seconds and plus 50% unless explicitly stated otherwise. All timeout values must be set to the specified values after Fundamental Reset. All counter values must be set to the specified values after Fundamental Reset.

4.2.5.1 *Detect*

The purpose of this state is to detect when a far end termination is present. This state can be entered at any time if directed.

4.2.5.2 *Polling*

The Port transmits training Ordered Sets and responds to the received training Ordered Sets. In this state, bit lock and Symbol lock are established and Lane polarity is configured.

The polling state includes Polling.Compliance (see Section 4.2.6.2.2). This state is intended for use with test equipment used to assess if the Transmitter and the interconnect present in the device under test setup is compliant with the voltage and timing specifications in Table 8-3 and Table 8-11.

The Polling.Compliance state also includes a simplified inter-operability testing scheme that is intended to be performed using a wide array of test and measurement equipment (i.e., pattern generator, oscilloscope, BERT, etc.). This portion of the Polling.Compliance state is logically entered by at least one component asserting the Compliance Receive bit (bit 4 in Symbol 5 of TS1) while not asserting the Loopback bit (bit 2 in Symbol 5 of TS1) upon entering Polling.Active. The ability to set the Compliance Receive bit is implementation specific. A provision for changing data rates to that indicated by the highest common transmitted and received Data Rate Identifiers (Symbol 4 of TS1) is also included to make this behavior scalable to various data rates.

IMPLEMENTATION NOTE

Use of Polling.Compliance

Polling.Compliance is intended for a compliance test environment and not entered during normal operation and cannot be disabled for any reason. Polling.Compliance is entered based on the physical system environment or configuration register access mechanism as described in Section 4.2.6.2.1. Any other mechanism that causes a Transmitter to output the compliance pattern is implementation specific and is beyond the scope of this specification.

4.2.5.3 *Configuration*

In Configuration, both the Transmitter and Receiver are sending and receiving data at the negotiated data rate. The Lanes of a Port configure into a Link through a width and Lane negotiation sequence. Also, Lane-to-Lane de-skew must occur, scrambling can be disabled if permitted, the N_FTS is set, and the Disable or Loopback states can be entered.

4.2.5.4 *Recovery*

In Recovery, both the Transmitter and Receiver are sending and receiving data using the configured Link and Lane number as well as the previously supported data rate(s). Recovery allows a configured Link to change the data rate of operation if desired, re-establish bit lock, Symbol lock or Block alignment, and Lane-to-Lane de-skew. Recovery is also used to set a new N_FTS value and enter the Loopback, Disabled, Hot Reset, and Configuration states.

4.2.5.5 *L0*

L0 is the normal operational state where data and control packets can be transmitted and received. All power management states are entered from this state.

4.2.5.6 *L0s*

L0s is intended as a power savings state. When operating with separate reference clocks with independent Spread Spectrum Clocking (SSC) (see Section 4.2.7), L0s is not supported and must not be advertised in the capability registers. See Section 4.3.7.3 for a definition of SSC.

L0s allows a Link to quickly enter and recover from a power conservation state without going through Recovery.

The entry to L0s occurs after receiving an EIOS.

The exit from L0s to L0 must re-establish bit lock, Symbol lock or Block alignment, and Lane-to-Lane de-skew.

A Transmitter and Receiver Lane pair on a Port are not required to both be in L0s simultaneously.

4.2.5.7 *L1*

L1 is intended as a power savings state.

The L1 state allows an additional power savings over L0s at the cost of additional resume latency.

The entry to L1 occurs after being directed by the Data Link Layer and receiving an EIOS.

4.2.5.8 *L2*

Power can be aggressively conserved in L2. Most of the Transmitter and Receiver may be shut off.[50] Main power and clocks are not guaranteed, but Aux[51] power is available.

When Beacon support is required by the associated system or form factor specification, an Upstream Port that supports the wakeup capability must be able to send; and a Downstream Port must be able to receive; a wakeup signal referred to as a Beacon.

The entry to L2 occurs after being directed by the Data Link Layer and receiving an EIOS.

---

[50] The exception is the Receiver termination, which must remain in a low impedance state.

[51] In this context, “Aux” power means a power source which can be used to drive the Beacon circuitry.

4.2.5.9 *Disabled*

The intent of the Disabled state is to allow a configured Link to be disabled until directed or Electrical Idle is exited (i.e., due to a hot removal and insertion) after entering Disabled.

Disabled uses bit 1 (Disable Link) in the Training Control field (see Table 4-5 and Table 4-6) which is sent within the TS1 and TS2 Ordered Sets.

A Link can enter Disabled if directed by a higher Layer. A Link can also reach the Disabled state by receiving two consecutive TS1 Ordered Sets with the Disable Link bit asserted (see Sections 4.2.6.3.1 and 4.2.6.4.5).

4.2.5.10 *Loopback*

Loopback is intended for test and fault isolation use. Only the entry and exit behavior is specified, all other details are implementation specific. Loopback can operate on either a per-Lane or configured Link basis.

A loopback master is the component requesting Loopback.

A loopback slave is the component looping back the data.

Loopback uses bit 2 (Loopback) in the Training Control field (see Table 4-5 and Table 4-6) which is sent within the TS1 and TS2 Ordered Sets.

The entry mechanism for a loopback master is device specific.

The loopback slave device enters Loopback whenever two consecutive TS1 Ordered Sets are received with the Loopback bit set.

IMPLEMENTATION NOTE

Use of Loopback

Once in the Loopback state, the master can send any pattern of Symbols as long as the encoding rules are followed. Once in Loopback, the concept of data scrambling is no longer relevant; what is sent out is looped back. The mechanism(s) and/or interface(s) utilized by the Data Link Layer to notify the Physical Layer to enter the Loopback state is component implementation specific and beyond the scope of this specification.

4.2.5.11 *Hot Reset*

Hot Reset uses bit 0 (Hot Reset) in the Training Control field (see Table 4-5 and Table 4-6) within the TS1 and TS2 Ordered Sets.

A Link can enter Hot Reset if directed by a higher Layer. A Link can also reach the Hot Reset state by receiving two consecutive TS1 Ordered Sets with the Hot Reset bit asserted (see Section 4.2.6.11).

4.2.6 Link Training and Status State Rules

Various Link status bits are monitored through software with the exception of LinkUp which is monitored by the Data Link Layer. Table 4-15 describes how the Link status bits must be handled throughout the LTSSM (for more information, see Section 3.2 for LinkUp; Section 7.5.3.8 for Link Speed, Link Width, and Link Training; Section 6.2 for Receiver Error; and Section 6.7 for In-Band Presence). A Receiver may also optionally report an 8b/10b Error in the Lane Error Status register when operating in 8b/10b encoding, when allowed to report the error as a Receiver Error in Table 4-15.

IMPLEMENTATION NOTE

Receiver Errors During Configuration and Recovery States

Allowing Receiver Errors to be set while in Configuration or Recovery is intended to allow implementations to report Link Errors that occur while processing packets in those states. For example, if the LTSSM transitions from L0 to Recovery while a TLP is being received, a Link Error that occurs after the LTSSM transition can be reported.

Table 4-15: Link Status Mapped to the LTSSM

LTSSM State	Link Width	Link Speed	LinkUp	Link Training	Receiver Error	In-Band Presence [52]
Detect	Undefined	Undefined	0b	0b	No action	0b
Polling	Undefined	Set to 2.5 GT/s on entry from Detect. Link speed may change on entry to Polling.Compliance.	0b	0b	No action	1b
Configuration	Set	No action	0b/1b [53]	1b	Set on 8b/10b Error. Optional: Set on Link Error when using 128b/130b encoding.	1b
Recovery	No action	Set to new speed when speed changes	1b	1b	Optionally set on Link Error.	1b
L0	No action	No action	1b	0b	Set on Link Error.	1b
L0s	No action	No action	1b	0b	No action	1b

[52] In-band refers to the fact that no sideband signals are used to calculate the presence of a powered up device on the other end of a Link.

[53] LinkUp will always be 0 if coming into Configuration via Detect -> Polling -> Configuration and LinkUp will always be 1 if coming into Configuration from any other state.

LTSSM State	Link Width	Link Speed	LinkUp	Link Training	Receiver Error	In-Band Presence 52
L1	No action	No action	1b	0b	No action	1b
L2	No action	No action	1b	0b	No action	1b
Disabled	Undefined	Undefined	0b	0b	Optional: Set on 8b/10b Error	1b
Loopback	No action	Link speed may change on entry to Loopback from Configuration.	0b	0b	No action	1b
Hot Reset	No action	No action	0b	0b	Optional: Set on 8b/10b Error	1b

The state machine rules for configuring and operating a PCI Express Link are defined in the following sections.

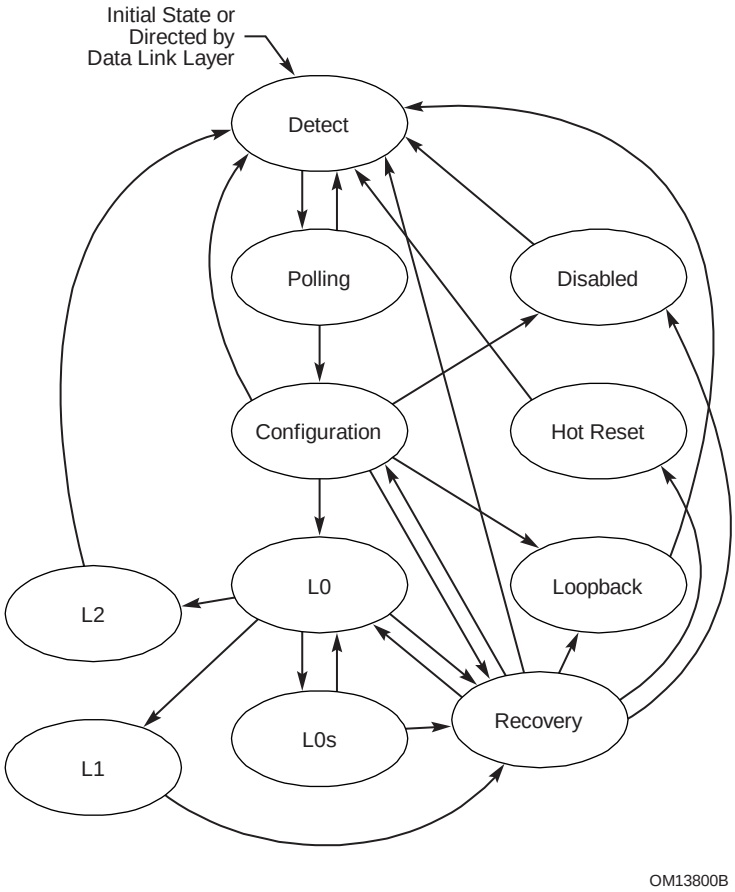


Figure 4-23: Main State Diagram for Link Training and Status State Machine

4.2.6.1 *Detect*

The Detect substate machine is shown in Figure 4-24.

4.2.6.1.1 Detect.Quiet

- Transmitter is in an Electrical Idle state.
  - The DC common mode voltage is not required to be within specification.
- 2.5 GT/s data rate is selected as the frequency of operation. If the frequency of operation was not 2.5 GT/s data rate on entry to this substate, the LTSSM must stay in this substate for at least 1 ms, during which the frequency of operation must be changed to the 2.5 GT/s data rate.
  - Note: This does not affect the advertised data rate in the TS1 and TS2 Ordered Sets.
- All Receivers must meet the the $Z_{RX\text{-}DC}$ specification for 2.5 GT/s within 1ms (see Table 8-11) of entering this substate. The LTSSM must stay in this substate until the $Z_{RX\text{-}DC}$ specification for 2.5 GT/s is met.
- LinkUp = 0b (status is cleared).
- The Equalization 8.0 GT/s Phase 1 Successful, Equalization 8.0 GT/s Phase 2 Successful, Equalization 8.0 GT/s Phase 3 Successful, and Equalization 8.0 GT/s Complete bits of the Link Status 2 register are all set to 0b. The Equalization 16.0 GT/s Phase 1 Successful, Equalization 16.0 GT/s Phase 2 Successful, Equalization 16.0 GT/s Phase 3 Successful and Equalization 16.0 GT/s Complete bits of the 16.0 GT/s Status register are all set to 0b.
- The directed_speed_change variable is reset to 0b. The upconfigure_capable variable is reset to 0b. The idle_to_rlock_transitioned variable is reset to 00h. The select_deemphasis variable must be set to either 0b or 1b based on platform specific needs for an Upstream Port and identical to the Selectable Preset/De-emphasis field in the Link Control 2 register for a Downstream Port. The equalization_done_8GT_data_rate and equalization_done_16GT_data_rate variables are reset to 0b.
  - Note that since these variables are defined with the 2.0 specification, pre-2.0 devices would not implement these variables and will always take the path as if the directed_speed_change and upconfigure_capable variables are constantly reset to 0b and the idle_to_rlock_transitioned variable is constantly set to FFh.
- The next state is Detect.Active after a 12 ms timeout or if Electrical Idle is broken on any Lane.

4.2.6.1.2 Detect.Active

- The Transmitter performs a Receiver Detection sequence on all un-configured Lanes that can form one or more Links (see Section 8.4.5.7 for more information).
- Next state is Polling if a Receiver is detected on all unconfigured Lanes.
- Next state is Detect.Quiet if a Receiver is not detected on any Lane.
- If at least one but not all un-configured Lanes detect a Receiver, then:
  1. Wait for 12 ms.
  2. The Transmitter performs a Receiver Detection sequence on all un-configured Lanes that can form one or more Links (see Section 8.4.5.7 for more information),

- The next state is Polling if exactly the same Lanes detect a Receiver as the first Receiver Detection sequence.
  - Lanes that did not detect a Receiver must:

    i) Be associated with a new LTSSM if this optional feature is supported.

    or

    ii) All Lanes that cannot be associated with an optional new LTSSM must transition to Electrical Idle.[54]
      - These Lanes must be re-associated with the LTSSM immediately after the LTSSM in progress transitions back to Detect.
      - An EIOS does not need to be sent before transitioning to Electrical Idle.
- Otherwise, the next state is Detect.Quiet.

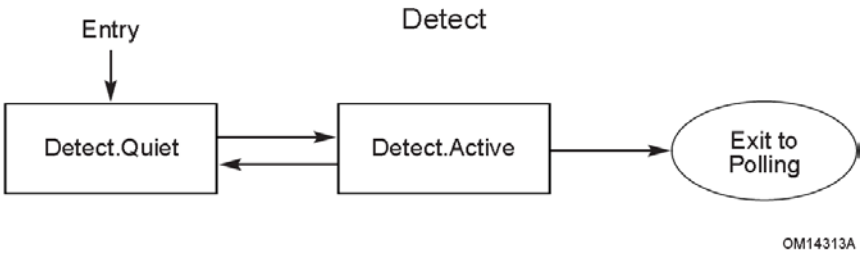


Figure 4-24: Detect Substate Machine

4.2.6.2 *Polling*

The Polling substate machine is shown in Figure 4-25.

4.2.6.2.1 Polling.Active

- Transmitter sends TS1 Ordered Sets with Lane and Link numbers set to PAD on all Lanes that detected a Receiver during Detect.
  - The Data Rate Identifier Symbol of the TS1 Ordered Sets must advertise all data rates that the Port supports, including those that it does not intend to use.
  - The Transmitter must wait for its TX common mode to settle before exiting from Electrical Idle and transmitting the TS1 Ordered Sets.
    - The Transmitter must drive patterns in the default voltage level of the Transmit Margin field within 192 ns from entry to this state. This transmit voltage level will remain in effect until Polling.Compliance or Recovery.RcvrLock is entered.

[54] The common mode being driven is not required to meet the Absolute Delta Between DC Common Mode During L0 and Electrical Idle ($V_{TX\text{-}CM\text{-}DC\text{-}ACTIVE\text{-}IDLE\text{-}DELTA}$) specification (see Table 8-7).

- Next state is Polling.Compliance if the Enter Compliance bit (bit 4) in the Link Control 2 register is 1b. If the Enter Compliance bit was set prior to entry to Polling.Active, the transition to Polling.Compliance must be immediate without sending any TS1 Ordered Sets.
- Next state is Polling.Configuration after at least 1024 TS1 Ordered Sets were transmitted, and all Lanes that detected a Receiver during Detect receive eight consecutive training sequences (or their complement) satisfying any of the following conditions:
  - TS1 with Lane and Link numbers set to PAD and the Compliance Receive bit (bit 4 of Symbol 5) is 0b.
  - TS1 with Lane and Link numbers set to PAD and the Loopback bit (bit 2 of Symbol 5) is 1b.
  - TS2 with Lane and Link numbers set to PAD.
- Otherwise, after a 24 ms timeout the next state is:
  - Polling.Configuration if,

  (i) Any Lane, which detected a Receiver during Detect, received eight consecutive training sequences (or their complement) satisfying any of the following conditions:

    1. TS1 with Lane and Link numbers set to PAD and the Compliance Receive bit (bit 4 of Symbol 5) is 0b.
    2. TS1 with Lane and Link numbers set to PAD and the Loopback bit (bit 2 of Symbol 5) is 1b.
    3. TS2 with Lane and Link numbers set to PAD.

    and a minimum of 1024 TS1 Ordered Sets are transmitted after receiving one TS1 or TS2 Ordered Set[55].

    And

  (ii) At least a predetermined set of Lanes that detected a Receiver during Detect have detected an exit from Electrical Idle at least once since entering Polling.Active.

    - Note: This may prevent one or more bad Receivers or Transmitters from holding up a valid Link from being configured, and allow for additional training in Polling.Configuration. The exact set of predetermined Lanes is implementation specific. Note that up to the 1.1 specification this predetermined set was equal to the total set of Lanes that detected a Receiver.
    - Note: Any Lane that receives eight consecutive TS1 or TS2 Ordered Sets should have detected an exit from Electrical Idle at least once since entering Polling.Active.
  - Else Polling.Compliance if either (a) or (b) is true:

  (a) not all Lanes from the predetermined set of Lanes from (ii) above have detected an exit from Electrical Idle since entering Polling.Active.

[55] Earlier versions of this specification required transmission of 1024 TS1 Ordered Sets after receiving one TS1 Ordered Set. This behavior is still permitted but the implementation will be more robust if it follows the behavior of transmitting 1024 TS1 Ordered Sets after receiving one TS1 or TS2 Ordered Set.

(b) any Lane that detected a Receiver during Detect received eight consecutive TS1 Ordered Sets (or their complement) with the Lane and Link numbers set to PAD, the Compliance Receive bit (bit 4 of Symbol 5) is 1b, and the Loopback bit (bit 2 of Symbol 5) is 0b.

- Note: If a passive test load is applied on all Lanes then the device will go to Polling.Compliance.

- Else Detect if the conditions to transition to Polling.Configuration or Polling.Compliance are not met

4.2.6.2.2 Polling.Compliance

- The Transmit Margin field of the Link Control 2 register is sampled on entry to this substate and becomes effective on the transmit package pins within 192 ns of entry to this substate and remain effective through the time the LTSSM is in this substate.
- The data rate and de-emphasis level for transmitting the compliance pattern are determined on the transition from Polling.Active to Polling.Compliance using the following algorithm.
  - If the Port is capable of transmitting at the 2.5 GT/s data rate only, the data rate for transmitting the compliance pattern is 2.5 GT/s and the de-emphasis level is -3.5 dB.
  - Else if the Port entered Polling.Compliance due to detecting eight consecutive TS1 Ordered Sets in Polling.Active with the Compliance Receive bit (bit 4 of Symbol 5) asserted and the Loopback bit (bit 2 of Symbol 5) deasserted then the data rate for transmission is that indicated by the highest common transmitted and received Data Rate Identifiers (Symbol 4 of the TS1 sequence) advertised on the eight consecutive TS1 Ordered Sets received on any Lane that detected a Receiver during Detect. The select_deemphasis variable must be set equal to the Selectable De-emphasis bit (Symbol 4 bit 6) in the eight consecutive TS1 Ordered Sets it received in Polling.Active substate. If the common data rate is 8.0 GT/s or higher, the select_preset variable on each Lane is set to the Transmitter preset value advertised in the Transmitter Preset field of the eight consecutive EQ TS1 Ordered Sets on the corresponding Lane , provided the value is not a Reserved encoding, and this value must be used by the transmitter (for 8.0 GT/s Data Rate, use of the Receiver preset hint value advertised in those eight consecutive EQ TS1 Ordered Sets is optional). If the common Data Rate is 8.0 GT/s or higher, any Lanes that did not receive eight consecutive EQ TS1 Ordered Sets with Transmitter preset information, or that received a value for a Reserved encoding, can use any supported Transmitter preset in an implementation specific manner.
  - Else if the Enter Compliance bit in the Link Control 2 register is 1b, the data rate for transmitting the compliance pattern is defined by the Target Link Speed field in the Link Control 2 register. The select_deemphasis variable is Set when the Compliance Preset/De-emphasis field in the Link Control 2 register equals 0001b if the data rate will be 5.0 GT/s. If the data rate will be 8.0 GT/s or higher, the select_preset variable on each Lane is set to, and the transmitter must operate with, the preset value provided in the Compliance Preset/De-emphasis Value (bits 15:12) in the Link Control 2 register provided the value is not a Reserved encoding.
  - Else the data rate, preset, and de-emphasis level settings are defined as follows based on the component's maximum supported data rate and the number of times Polling.Compliance has been entered with this entry criteria:

    Setting #1: Data Rate = 2.5 GT/s, De-emphasis Level = -3.5 dB

    Setting #2: Data Rate = 5.0 GT/s, De-emphasis Level = -3.5 dB

    Setting #3: Data Rate = 5.0 GT/s, De-emphasis Level = -6 dB

    Setting #4: Data Rate = 8.0 GT/s, with Transmitter Preset Encoding 0000b defined in Section 4.2.3.2

Setting #5: Data Rate = 8.0 GT/s, with Transmitter Preset Encoding 0001b defined in Section 4.2.3.2

Setting #6: Data Rate = 8.0 GT/s, with Transmitter Preset Encoding 0010b defined in Section 4.2.3.2

Setting #7: Data Rate = 8.0 GT/s, with Transmitter Preset Encoding 0011b defined in Section 4.2.3.2

Setting #8: Data Rate = 8.0 GT/s, with Transmitter Preset Encoding 0100b defined in Section 4.2.3.2

Setting #9: Data Rate = 8.0 GT/s, with Transmitter Preset Encoding 0101b defined in Section 4.2.3.2

Setting #10: Data Rate = 8.0 GT/s, with Transmitter Preset Encoding 0110b defined in Section 4.2.3.2

Setting #11: Data Rate = 8.0 GT/s, with Transmitter Preset Encoding 0111b defined in Section 4.2.3.2

Setting #12: Data Rate = 8.0 GT/s, with Transmitter Preset Encoding 1000b defined in Section 4.2.3.2

Setting #13: Data Rate = 8.0 GT/s, with Transmitter Preset Encoding 1001b defined in Section 4.2.3.2

Setting #14: Data Rate = 8.0 GT/s, with Transmitter Preset Encoding 1010b defined in Section 4.2.3.2

Setting #15: Data Rate = 16.0 GT/s, with Transmitter Preset Encoding 0000b defined in Section 4.2.3.2

Setting #16: Data Rate = 16.0 GT/s, with Transmitter Preset Encoding 0001b defined in Section 4.2.3.2

Setting #17: Data Rate = 16.0 GT/s, with Transmitter Preset Encoding 0010b defined in Section 4.2.3.2

Setting #18: Data Rate = 16.0 GT/s, with Transmitter Preset Encoding 0011b defined in Section 4.2.3.2

Setting #19: Data Rate = 16.0 GT/s, with Transmitter Preset Encoding 0100b defined in Section 4.2.3.2

Setting #20: Data Rate = 16.0 GT/s, with Transmitter Preset Encoding 0101b defined in Section 4.2.3.2

Setting #21: Data Rate = 16.0 GT/s, with Transmitter Preset Encoding 0110b defined in Section 4.2.3.2

Setting #22: Data Rate = 16.0 GT/s, with Transmitter Preset Encoding 0111b defined in Section 4.2.3.2

Setting #23: Data Rate = 16.0 GT/s, with Transmitter Preset Encoding 1000b defined in Section 4.2.3.2

Setting #24: Data Rate = 16.0 GT/s, with Transmitter Preset Encoding 1001b defined in Section 4.2.3.2

Setting #25: Data Rate = 16.0 GT/s, with Transmitter Preset Encoding 1010b defined in Section 4.2.3.2

Settings #26 to #34: Data Rate = 16.0 GT/s, with Transmitter Preset Encoding 0100b defined in Section 4.2.3.2.

Subsequent entries to Polling.Compliance repeat the above sequence. For example, the state sequence which causes a Port to transmit the Compliance pattern at a data rate of 5.0 GT/s and a de-emphasis level of -6 dB is: Polling.Active, Polling.Compliance (2.5 GT/s and -3.5 dB), Polling.Active, Polling.Compliance (5.0 GT/s and -3.5 dB), Polling.Active, Polling.Compliance (5.0 GT/s and -6 dB).

The sequence must be set to Setting #1 in the Polling.Configuration state if the Port supports 16.0 GT/s or higher Data Rates, or the Port's Receivers do not meet the $Z_{RX\text{-}DC}$ specification for 2.5 GT/s when they are operating at 8.0 GT/s or higher data rates (see Table 8-11). All Ports are permitted to set the sequence to Setting #1 in the Polling.Configuration state.

IMPLEMENTATION NOTE

Compliance Load Board Usage to Generate Compliance Patterns

It is envisioned that the compliance load (base) board may send a 100 MHz signal for about 1 ms on one leg of a differential pair at 350 mV peak-to-peak on any Lane to cycle the device to the desired speed and de-emphasis level. The device under test is required, based on its maximum supported data rate, to cycle through the following settings in order, for each entry to Polling.Compliance from Polling.Active, starting with the first setting on the first entry to Polling.Compliance after the Fundamental Reset:

- Data Rate = 2.5 GT/s, De-emphasis Level = -3.5 dB
- Data Rate = 5.0 GT/s, De-emphasis Level = -3.5 dB
- Data Rate = 5.0 GT/s, De-emphasis Level = -6 dB
- Data Rate = 8.0 GT/s, with Transmitter Preset Encoding 0000b defined in Section 4.2.3.2
- Data Rate = 8.0 GT/s, with Transmitter Preset Encoding 0001b defined in Section 4.2.3.2
- Data Rate = 8.0 GT/s, with Transmitter Preset Encoding 0010b defined in Section 4.2.3.2
- Data Rate = 8.0 GT/s, with Transmitter Preset Encoding 0011b defined in Section 4.2.3.2
- Data Rate = 8.0 GT/s, with Transmitter Preset Encoding 0100b defined in Section 4.2.3.2
- Data Rate = 8.0 GT/s, with Transmitter Preset Encoding 0101b defined in Section 4.2.3.2
- Data Rate = 8.0 GT/s, with Transmitter Preset Encoding 0110b defined in Section 4.2.3.2
- Data Rate = 8.0 GT/s, with Transmitter Preset Encoding 0111b defined in Section 4.2.3.2
- Data Rate = 8.0 GT/s, with Transmitter Preset Encoding 1000b defined in Section 4.2.3.2
- Data Rate = 8.0 GT/s, with Transmitter Preset Encoding 1001b defined in Section 4.2.3.2
- Data Rate = 8.0 GT/s, with Transmitter Preset Encoding 1010b defined in Section 4.2.3.2
- Data Rate = 16.0 GT/s, with Transmitter Preset Encoding 0000b defined in Section 4.2.3.2
- Data Rate = 16.0 GT/s, with Transmitter Preset Encoding 0001b defined in Section 4.2.3.2
- Data Rate = 16.0 GT/s, with Transmitter Preset Encoding 0010b defined in Section 4.2.3.2
- Data Rate = 16.0 GT/s, with Transmitter Preset Encoding 0011b defined in Section 4.2.3.2
- Data Rate = 16.0 GT/s, with Transmitter Preset Encoding 0100b defined in Section 4.2.3.2
- Data Rate = 16.0 GT/s, with Transmitter Preset Encoding 0101b defined in Section 4.2.3.2
- Data Rate = 16.0 GT/s, with Transmitter Preset Encoding 0110b defined in Section 4.2.3.2
- Data Rate = 16.0 GT/s, with Transmitter Preset Encoding 0111b defined in Section 4.2.3.2
- Data Rate = 16.0 GT/s, with Transmitter Preset Encoding 1000b defined in Section 4.2.3.2
- Data Rate = 16.0 GT/s, with Transmitter Preset Encoding 1001b defined in Section 4.2.3.2
- Data Rate = 16.0 GT/s, with Transmitter Preset Encoding 1010b defined in Section 4.2.3.2
- Nine instances of Data Rate = 16.0 GT/s, with Transmitter Preset Encoding 0100b defined in Section 4.2.3.2.

- If the compliance pattern data rate is not 2.5 GT/s and any TS1 Ordered Sets were transmitted in Polling.Active prior to entering Polling.Compliance, the Transmitter sends either one EIOS or two consecutive EIOSs prior to entering Electrical Idle. If the compliance pattern data rate is not 2.5 GT/s and TS1 Ordered Sets were not transmitted in Polling.Active prior to entering

Polling.Compliance, the Transmitter must enter Electrical Idle without transmitting any EIOSs. During the period of Electrical Idle, the data rate is changed to the new speed and stabilized. If the frequency of operation will be 5.0 GT/s, the de-emphasis/preset level must be set to -3.5 dB if the select_deemphasis variable is 1b else it must be set to -6 dB. If the frequency of operation will be 8.0 GT/s or higher, the Transmitter preset value must be set to the value in the select_preset variable. The period of Electrical Idle is greater than 1 ms but it is not to exceed 2 ms.

- Behavior during Polling Compliance after the data rate and de-emphasis/preset level are determined must follow the following rules:
  - If the Port entered Polling.Compliance due to detecting eight consecutive TS1 Ordered Sets in Polling.Active with the Compliance Receive bit (bit 4 of Symbol 5) asserted and the Loopback bit (bit 2 of Symbol 5) deasserted or both the Enter Compliance bit and the Enter Modified Compliance bit in the Link Control 2 register are set to 1b then the Transmitter sends out the Modified Compliance Pattern (see Section 4.2.9) at the above determined data rate with the error status Symbol set to all 0's on all Lanes that detected a Receiver during Detect.
    - If the data rate is 2.5 GT/s or 5.0 GT/s, a particular Lane's Receiver independently signifies a successful lock to the incoming Modified Compliance Pattern by looking for any one occurrence of the Modified Compliance Pattern and then setting the Pattern Lock bit (bit 8 of the 8 bit error status Symbol) in the same Lane of its own transmitted Modified Compliance Pattern.
      - The error status Symbols are not to be used for the lock process since they are undefined at any given moment.
      - An occurrence is defined above as the following sequence of 8b/10b Symbols; K28.5, D21.5, K28.5, and D10.2 or the complement of each of the individual Symbols.
      - The device under test must set the Pattern Lock bit of the Modified Compliance Pattern it transmits at the Transmitter package pin(s) after successfully locking to the incoming Modified Compliance Pattern within 1 ms of receiving the Modified Compliance Pattern at its Receiver package pin(s).
    - If the data rate is 8.0 GT/s or higher: The Error_Status field is set to 00h on entry to this substate. Each Lane sets the Pattern Lock bit independently when it achieves Block Alignment as described in Section 4.2.2.2.1. After Pattern Lock is achieved, Symbols received in Data Blocks are compared to the Idle data Symbol (00h) and each mismatched Symbol causes the Receiver Error Count field to be incremented by 1. The Receiver Error Count saturates at 127 (further mismatched Symbols do not change the Receiver Error Count). The Pattern Lock and Receiver Error Count information for each Lane is transmitted as part of the SKP Ordered Sets transmitted in that Lane's Modified Compliance Pattern. See Section 4.2.7 for more information. The device under test must set the Pattern Lock bit in the SKP Ordered Set it transmits within 4 ms of receiving the Modified Compliance Pattern at its Receiver package pin(s).

The scrambling requirements defined in Section 4.2.2.4 are applied to the received Modified Compliance Pattern. For example, the scrambling LFSR seed is set per Lane, an EIEOS initializes the LFSR and SKP Ordered Sets do not advance the LFSR.

IMPLEMENTATION NOTE

Handling Bit Slip and Block Alignment

Devices should ensure that their Receivers have stabilized before attempting to obtain Block alignment and signaling Pattern Lock. For example, if an implementation expects to see bit slips in the initial few bits, it should wait for that time to be over before settling on a Block Alignment. Devices may also want to revalidate their Block alignment prior to setting the Pattern Lock bit.

- If the data rate is 2.5 GT/s or 5.0 GT/s, once a particular Lane indicates it has locked to the incoming Modified Compliance Pattern the Receiver Error Count for that particular Lane is incremented every time a Receiver error occurs.
    - The error status Symbol uses the lower 7 bits as the Receiver Error Count field and this field will remain stuck at all 1's if the count reaches 127.
    - The Receiver must not make any assumption about the 10-bit patterns it will receive when in this substate if 8b/10b encoding is used.
- If the Enter Compliance bit in the Link Control 2 register is 0b, the next state is Detect if directed
- Else if the Enter Compliance bit was set to 1b on entry to Polling.Compliance, next state is Polling.Active if any of the following conditions apply:
    - The Enter Compliance bit in the Link Control 2 register has changed to 0b
    - The Port is an Upstream Port and an EIOS is received on any Lane. The Enter Compliance bit is reset to 0b when this condition is true.

  If the Transmitter was transmitting at a data rate other than 2.5 GT/s, or the Enter Compliance bit in the Link Control 2 register was set to 1b during entry to Polling.Compliance, the Transmitter sends eight consecutive EIOS and enters Electrical Idle prior to transitioning to Polling.Active. During the period of Electrical Idle, the data rate is changed to 2.5 GT/s and stabilized and the de-emphasis level is set to -3.5 dB. The period of Electrical Idle is greater than 1 ms but must not exceed 2 ms.
- Note: Sending multiple EIOS provides enough robustness such that the other Port detects at least one EIOS and exits Polling.Compliance substate when the configuration register mechanism was used for entry.

o Else if the Port entered Polling.Compliance due to the Enter Compliance bit of the Link Control 2 register being set to 1b and the Enter Modified Compliance bit of the Link Control 2 register being set to 0b:

(a) Transmitter sends out the compliance pattern on all Lanes that detected a Receiver during Detect at the data rate and de-emphasis/preset level determined above.

(b) Next state is Polling.Active if any of the following two conditions are true:

1.. The Enter Compliance bit in the Link Control 2 register has changed to 0b (from 1b) since entering Polling.Compliance.

2.. The Port is an Upstream Port, the Enter Compliance bit in the Link Control 2 register is set to 1b and an EIOS has been detected on any Lane. The Enter Compliance bit is reset to 0b when this condition is true.

The Transmitter sends eight consecutive EIOSs and enters Electrical Idle prior to transitioning to Polling.Active. During the period of Electrical Idle, the data rate is changed to 2.5 GT/s and stabilized. The period of Electrical Idle is greater than 1 ms but must not exceed 2 ms.

Note: Sending multiple EIOSs provides enough robustness such that the other Port detects at least one EIOS and exits Polling.

- Else:

(a) Transmitter sends out the following patterns on Lanes that detected a Receiver during Detect at the data rate and de-emphasis/preset level determined above:

- For Settings #1 to #25: Compliance pattern on all Lanes.
- For Setting #26: Jitter Measurement Pattern on all Lanes.
- For Setting #27: Jitter Measurement Pattern on Lanes 0/8/16/24 and Compliance pattern on all other Lanes.
- For Setting #28: Jitter Measurement Pattern on Lanes 1/9/17/25 and Compliance pattern on all other Lanes.
- For Setting #29: Jitter Measurement Pattern on Lanes 2/10/18/26 and Compliance pattern on all other Lanes.
- For Setting #30: Jitter Measurement Pattern on Lanes 3/11/19/27 and Compliance patter on all other Lanes.
- For Setting #31: Jitter Measurement Pattern on Lanes 4/12/20/28 and Compliance pattern on all other Lanes.
- For Setting #32: Jitter Measurement Pattern on Lanes 5/13/21/29 and Compliance pattern on all other Lanes.
- For Setting #33: Jitter Measurement Pattern on Lanes 6/14/22/30 and Compliance pattern on all other Lanes.
- For Setting #34: Jitter Measurement Pattern on Lanes 7/15/23/31 and Compliance pattern on all other Lanes.

(b) Next state is Polling.Active if an exit of Electrical Idle is detected at the Receiver of any Lane that detected a Receiver during Detect.
If the Transmitter is transmitting at a data rate other than 2.5 GT/s, the Transmitter sends eight consecutive EIOSs and enters Electrical Idle prior to transitioning to

Polling.Active. During the period of Electrical Idle, the data rate is changed to 2.5 GT/s and stabilized. The period of Electrical Idle is greater than 1 ms but must not exceed 2 ms.

4.2.6.2.3 Polling.Configuration

- Receiver must invert polarity if necessary (see Section 4.2.4.4).
- The Transmit Margin field of the Link Control 2 register must be reset to 000b on entry to this substate.
- The Transmitter's Polling.Compliance sequence setting is updated, if required, as described in Section 4.2.6.2.2.
- Transmitter sends TS2 Ordered Sets with Link and Lane numbers set to PAD on all Lanes that detected a Receiver during Detect.
  - The Data Rate Identifier Symbol of the TS2 Ordered Sets must advertise all data rates that the Port supports, including those that it does not intend to use.
- The next state is Configuration after eight consecutive TS2 Ordered Sets, with Link and Lane numbers set to PAD, are received on any Lanes that detected a Receiver during Detect, and 16 TS2 Ordered Sets are transmitted after receiving one TS2 Ordered Set.
- Otherwise, next state is Detect after a 48 ms timeout.

4.2.6.2.4 Polling.Speed

This state is unreachable given that the Link comes up to L0 in 2.5 GT/s data rate only and changes speed by entering Recovery.

IMPLEMENTATION NOTE

Support for Higher Data Rates than 2.5 GT/s

A Link will initially train to the L0 state at the 2.5 GT/s data rate even if both sides are capable of operating at a data rate greater than 2.5 GT/s. Supported higher data rates are advertised in the TS Ordered Sets. The other side's speed capability is registered during the Configuration.Complete substate. Based on the highest supported common data rate, either side can initiate a change in speed from the L0 state by transitioning to Recovery.

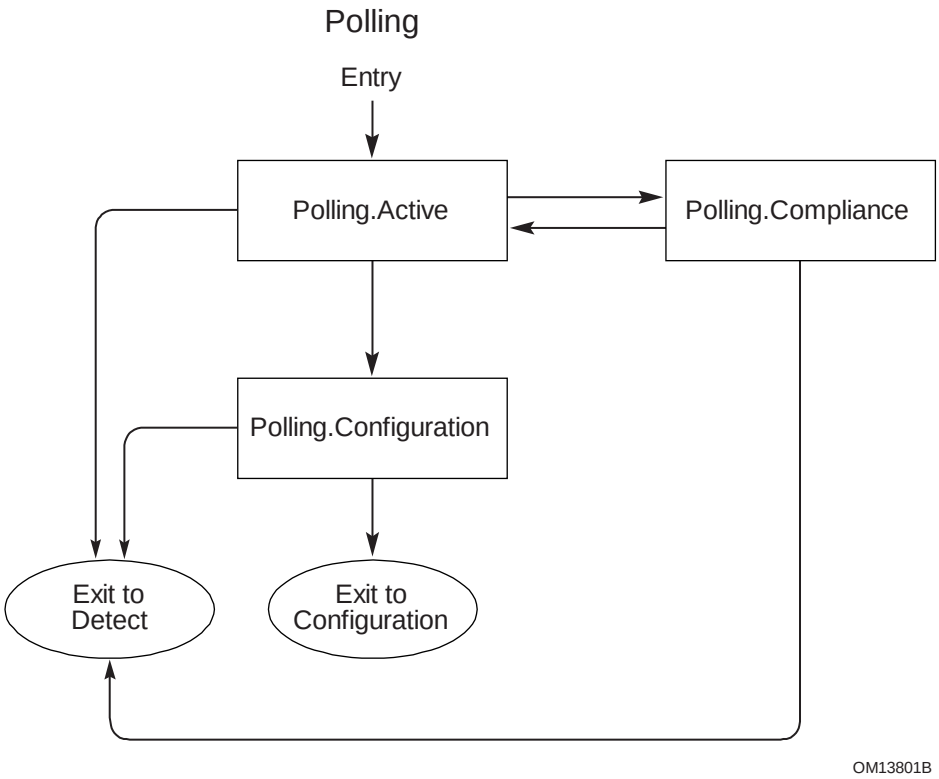


Figure 4-25: Polling Substate Machine

4.2.6.3 *Configuration*

The Configuration substate machine is shown in Figure 4-26.

4.2.6.3.1 Configuration.Linkwidth.Start

4.2.6.3.1.1 Downstream Lanes

- Next state is Disabled if directed.
  - Note: "if directed" applies to a Downstream Port that is instructed by a higher Layer to assert the Disable Link bit (TS1 and TS2) on all Lanes that detected a Receiver during Detect.
- Next state is Loopback if directed to this state, and the Transmitter is capable of being a loopback master, which is determined by implementation specific means.
  - Note: "if directed" applies to a Port that is instructed by a higher Layer to assert the Loopback bit (TS1 and TS2) on all Lanes that detected a Receiver during Detect.
- In the optional case where a crosslink is supported, the next state is Disabled after all Lanes that are transmitting TS1 Ordered Sets receive two consecutive TS1 Ordered Sets with the Disable Link bit asserted.
- Next state is Loopback after all Lanes that are transmitting TS1 Ordered Sets, that are also receiving Ordered Sets, receive the Loopback bit asserted in two consecutive TS1 Ordered Sets.

Note that the device receiving the Ordered Set with the Loopback bit set becomes the loopback slave.

- The Transmitter sends TS1 Ordered Sets with selected Link numbers and sets Lane numbers to PAD on all the active Downstream Lanes if LinkUp is 0b or if the LTSSM is not initiating upconfiguration of the Link width. In addition, if upconfigure_capable is set to 1b, and the LTSSM is not initiating upconfiguration of the Link width, the LTSSM sends TS1 Ordered Sets with the selected Link number and sets the Lane number to PAD on each inactive Lane after it detected an exit from Electrical Idle since entering Recovery and has subsequently received two consecutive TS1 Ordered Sets with the Link and Lane numbers each set to PAD while in this substate.
  - On transition to this substate from Polling, any Lane that detected a Receiver during Detect is considered an active Lane.
  - On transition to this substate from Recovery, any Lane that is part of the configured Link the previous time through Configuration.Complete is considered an active Lane.
  - The Data Rate Identifier Symbol of the TS1 Ordered Sets must advertise all data rates that the Port supports, including those that it does not intend to use.

- If LinkUp is 1b and the LTSSM is initiating upconfiguration of the Link width, initially it transmits TS1 Ordered Sets with both the Link and Lane numbers set to PAD on the current set of active Lanes; the inactive Lanes it intends to activate; and those Lanes where it detected an exit from Electrical Idle since entering Recovery and has received two consecutive TS1 Ordered Sets with the Link and Lane numbers each set to PAD. The LTSSM transmits TS1 Ordered Sets with the selected Link number and the Lane number set to PAD when each of the Lanes transmitting TS1 Ordered Sets receives two consecutive TS1 Ordered Sets with the Link and Lane numbers each set to PAD or 1 ms has expired since entering this substate.
    - After activating any inactive Lane, the Transmitter must wait for its TX common mode to settle before exiting from Electrical Idle and transmitting the TS1 Ordered Sets.
    - Link numbers are only permitted to be different for groups of Lanes capable of being a unique Link.
    - Note: An example of Link number assignments is a set of eight Downstream Lanes capable of negotiating to become one x8 Port when connected to one component or two x4 Ports when connected to two different components. The Downstream Lanes send out TS1 Ordered Sets with the Link number set to N on four Lanes and Link number set to N+1 on the other four Lanes. The Lane numbers are all set to PAD.
- If any Lanes first received at least one or more TS1 Ordered Sets with a Link and Lane number set to PAD, the next state is Configuration.Linkwidth.Accept immediately after any of those same Downstream Lanes receive two consecutive TS1 Ordered Sets with a non-PAD Link number that matches any of the transmitted Link numbers, and with a Lane number set to PAD.
    - If the crosslink configuration is not supported, the condition of first receiving a Link and Lane number set to PAD is always true.
- Else: Optionally, if LinkUp is 0b and if crosslinks are supported, then all Downstream Lanes that detected a Receiver during Detect must first transmit 16 to 32 TS1 Ordered Sets with a non-PAD Link number and PAD Lane number and after this occurs if any Downstream Lanes receive two consecutive TS1 Ordered Sets with a Link number different than PAD and a Lane Number set to PAD, the Downstream Lanes are now designated as Upstream Lanes and a new random crosslink timeout is chosen (see $T_{crosslink}$ in Table 8-3). The next state is Configuration.Linkwidth.Start as Upstream Lanes.
    - Note: This supports the optional crosslink where both sides may try to act as a Downstream Port. This is resolved by making both Ports become Upstream and assigning a random timeout until one side of the Link becomes a Downstream Port and the other side remains an Upstream Port. This timeout must be random even when hooking up two of the same devices so as to eventually break any possible deadlock.
    - If crosslinks are supported, receiving a sequence of TS1 Ordered Sets with a Link number of PAD followed by a Link number of non-PAD that matches the transmitted Link number is only valid when not interrupted by the reception of a TS2 Ordered Set.

IMPLEMENTATION NOTE

Crosslink Initialization

In the case where the Downstream Lanes are connected to both Downstream Lanes (crosslink) and Upstream Lanes, the Port with the Downstream Lanes may continue with a single LTSSM as described in this section or optionally, split into multiple LTSSMs.

- The next state is Detect after a 24 ms timeout.

4.2.6.3.1.2 Upstream Lanes

- In the optional case where crosslinks are supported the next state is Disabled if directed.
  - Note: "if directed" only applies to an optional crosslink Port that is instructed by a higher Layer to assert the Disable Link bit (TS1 and TS2) on all Lanes that detected a Receiver during Detect.
- Next state is Loopback if directed to this state, and the Transmitter is capable of being a loopback master, which is determined by implementation specific means.
  - Note: "if directed" applies to a Port that is instructed by a higher Layer to assert the Loopback bit (TS1 and TS2) on all Lanes that detected a Receiver during Detect.
- Next state is Disabled after any Lanes that are transmitting TS1 Ordered Sets receive two consecutive TS1 Ordered Sets with the Disable Link bit asserted.
  - In the optional case where a crosslink is supported, the next state is Disabled only after all Lanes that are transmitting TS1 Ordered Sets, that are also receiving TS1 Ordered Sets, receive the Disable Link bit asserted in two consecutive TS1 Ordered Sets.
- Next state is Loopback after all Lanes that are transmitting TS1 Ordered Sets, that are also receiving TS1 Ordered Sets, receive the Loopback bit asserted in two consecutive TS1 Ordered Sets.
  - Note: The device receiving the Ordered Set with the Loopback bit set becomes the loopback slave.
- The Transmitter sends out TS1 Ordered Sets with Link numbers and Lane numbers set to PAD on all the active Upstream Lanes; the inactive Lanes it is initiating to upconfigure the Link width; and if upconfigure_capable is set to 1b, on each of the inactive Lanes where it detected an exit from Electrical Idle since entering Recovery and has subsequently received two consecutive TS1 Ordered Sets with Link and Lane numbers, each set to PAD, in this substate.
  - On transition to this substate from Polling, any Lane that detected a Receiver during Detect is considered an active Lane.
  - On transition to this substate from Recovery, any Lane that is part of the configured Link the previous time through Configuration.Complete is considered an active Lane.
  - On transition to this substate from Recovery, if the transition is not caused by LTSSM timeout, the Transmitter must set the Autonomous Change (Symbol 4 bit 6) to 1b in the TS1 Ordered Sets that it sends while in the Configuration state if the Transmitter intends to change the Link width for autonomous reasons.

- The Data Rate Identifier Symbol of the TS1 Ordered Sets must advertise all data rates that the Port supports, including those that it does not intend to use.

- If any Lane receives two consecutive TS1 Ordered Sets with Link numbers that are different than PAD and Lane number set to PAD, a single Link number is selected and Lane number set to PAD are transmitted on all Lanes that both detected a Receiver and also received two consecutive TS1 Ordered Sets with Link numbers that are different than PAD and Lane number set to PAD. Any left over Lanes that detected a Receiver during Detect must transmit TS1 Ordered Sets with the Link and Lane number set to PAD. The next state is Configuration.Linkwidth.Accept:
  - If the LTSSM is initiating upconfiguration of the Link width, it waits until it receives two consecutive TS1 Ordered Sets with a non-PAD Link Number and a PAD Lane number on all the inactive Lanes it wants to activate, or, 1 ms after entry to this substate, it receives two consecutive TS1 Ordered Sets on any Lane with a non-PAD Link number and PAD Lane number, whichever occurs earlier, before transmitting TS1 Ordered Sets with selected Link number and Lane number set to PAD.
  - It is recommended that any possible multi-Lane Link that received an error in a TS1 Ordered Set or lost 128b/130b Block Alignment on a subset of the received Lanes; delay the evaluation listed above by an additional two, or more, TS1 Ordered Sets when using 8b/10b encoding, or by an additional 34, or more, TS1 Ordered Sets when using 128b/130b encoding, but must not exceed 1 ms, so as not to prematurely configure a smaller Link than possible.
    - After activating any inactive Lane, the Transmitter must wait for its TX common mode to settle before exiting Electrical Idle and transmitting the TS1 Ordered Sets.

- Optionally, if LinkUp is 0b and if crosslinks are supported, then all Upstream Lanes that detected a Receiver during Detect must first transmit 16–32 TS1 Ordered Sets with a PAD Link number and PAD Lane number and after this occurs and if any Upstream Lanes first receive two consecutive TS1 Ordered Sets with Link and Lane numbers set to PAD, then:
  - The Transmitter continues to send out TS1 Ordered Sets with Link numbers and Lane numbers set to PAD.
  - If any Lanes receive two consecutive TS1 Ordered Sets with Link numbers that are different than PAD and Lane number set to PAD, a single Link number is selected and Lane number set to PAD are transmitted on all Lanes that both detected a Receiver and also received two consecutive TS1 Ordered Sets with Link numbers that are different than PAD and Lane number set to PAD. Any left over Lanes that detected a Receiver during Detect must transmit TS1 Ordered Sets with the Link and Lane number set to PAD. The next state is Configuration.Linkwidth.Accept.
    - It is recommended that any possible multi-Lane Link that received an error in a TS1 Ordered Set or lost 128b/130b Block Alignment on a subset of the received Lanes; delay the evaluation listed above by an additional two, or more, TS1 Ordered Sets when using 8b/10b encoding, or by an additional 34, or more, TS1 Ordered Sets when using 128b/130b encoding, but must not exceed 1 ms, so as not to prematurely configure a smaller Link than possible.

- Otherwise, after a $T_{crosslink}$ timeout, 16 to 32 TS2 Ordered Sets with PAD Link numbers and PAD Lane numbers are sent. The Upstream Lanes become Downstream Lanes and the next state is Configuration.Linkwidth.Start as Downstream Lanes.
  - Note: This optional behavior is required for crosslink behavior where two Ports may start off with Upstream Ports, and one will eventually take the lead as a Downstream Port.

- The next state is Detect after a 24 ms timeout.

4.2.6.3.2 Configuration.Linkwidth.Accept

4.2.6.3.2.1 Downstream Lanes

- If a configured Link can be formed with at least one group of Lanes that received two consecutive TS1 Ordered Sets with the same received Link number (non-PAD and matching one that was transmitted by the Downstream Lanes), TS1 Ordered Sets are transmitted with the same Link number and unique non-PAD Lane numbers are assigned to all these same Lanes. The next state is Configuration.Lanenum.Wait.
  - The assigned non-PAD Lane numbers must range from 0 to n-1, be assigned sequentially to the same grouping of Lanes that are receiving the same Link number, and Downstream Lanes which are not receiving TS1 Ordered Sets must not disrupt the initial sequential numbering of the widest possible Link. Any left over Lanes must transmit TS1 Ordered Sets with the Link and Lane number set to PAD.
  - It is recommended that any possible multi-Lane Link that received an error in a TS1 Ordered Set or lost 128b/130b Block Alignment on a subset of the received Lanes delay the evaluation listed above by an additional two, or more, TS1 Ordered Sets when using 8b/10b encoding, or by an additional 34, or more, TS1 Ordered Sets when using 128b/130b encoding, but must not exceed 1 ms, so as not to prematurely configure a smaller Link than possible.

- The next state is Detect after a 2 ms timeout or if no Link can be configured or if all Lanes receive two consecutive TS1 Ordered Sets with Link and Lane numbers set to PAD.

4.2.6.3.2.2 Upstream Lanes

- If a configured Link can be formed using Lanes that transmitted a non-PAD Link number which are receiving two consecutive TS1 Ordered Sets with the same non-PAD Link number and any non-PAD Lane number, TS1 Ordered Sets are transmitted with the same non-PAD Link number and Lane numbers that, if possible, match the received Lane numbers or are different, if necessary, (i.e., Lane reversed). The next state is Configuration.Lanenum.Wait.
  - The newly assigned Lane numbers must range from 0 to m-1, be assigned sequentially only to some continuous grouping of Lanes that are receiving non-PAD Lane numbers (i.e., Lanes which are not receiving any TS1 Ordered Sets always disrupt a continuous grouping and must not be included in this grouping), must include either Lane 0 or Lane n-1 (largest received Lane number), and m-1 must be equal to or smaller than the largest

received Lane number (n-1). Remaining Lanes must transmit TS1 Ordered Sets with Link and Lane numbers set to PAD.

- It is recommended that any possible multi-Lane Link that received an error in a TS1 Ordered Set or lost 128b/130b Block Alignment on a subset of the received Lanes delay the evaluation listed above by an additional two, or more, TS1 Ordered Sets when using 8b/10b encoding, or by an additional 34, or more, TS1 Ordered Sets when using 128b/130b encoding, but must not exceed 1 ms, so as not to prematurely configure a smaller Link than possible.

- The next state is Detect after a 2 ms timeout or if no Link can be configured or if all Lanes receive two consecutive TS1 Ordered Sets with Link and Lane numbers set to PAD.

IMPLEMENTATION NOTE

Example Cases

Notable examples related to the configuration of Downstream Lanes:

1. A x8 Downstream Port, which can be divided into two x4 Links, sends two different Link numbers on to two x4 Upstream Ports. The Upstream Ports respond simultaneously by picking the two Link numbers. The Downstream Port will have to choose one of these sets of Link numbers to configure as a Link, and leave the other for a secondary LTSSM to configure (which will ultimately happen in Configuration.Complete).
2. A x16 Downstream Port, which can be divided into two x8 Links, is hooked up to a x12 Upstream Port that can be configured as a x12 Link or a x8 and a x4 Link. During Configuration.Linkwidth.Start the Upstream Port returned the same Link number on all 12 Lanes. The Downstream Port would then return the same received Link number and assign Lane numbers on the eight Lanes that can form a x8 Link with the remaining four Lanes transmitting a Lane number and a Link number set to PAD.
3. A x8 Downstream Port where only seven Lanes are receiving TS1 Ordered Sets with the same received Link number (non-PAD and matching one that was transmitted by the Downstream Lanes) and an eighth Lane, which is in the middle or adjacent to those same Lanes, is not receiving a TS1 Ordered Set. In this case, the eighth Lane is treated the same as the other seven Lanes and Lane numbering for a x8 Lane should occur as described above.

Notable examples related to the configuration of Upstream Lanes:

A x8 Upstream Port is presented with Lane numbers that are backward from the preferred numbering. If the optional behavior of Lane reversal is supported by the Upstream Port, the Upstream Port transmits the same Lane numbers back to the Downstream Port. Otherwise the opposite Lane numbers are transmitted back to the Downstream Port, and it will be up to the Downstream Port to optionally fix the Lane ordering or exit Configuration.

Optional Lane reversal behavior is required to configure a Link where the Lane numbers are reversed and the Downstream Port does not support Lane reversal. Specifically, the Upstream Port Lane reversal will accommodate the scenario where the default Upstream sequential Lane numbering (0 to n-1) is receiving a reversed Downstream sequential Lane number (n-1 to 0).

A x8 Upstream Port is not receiving TS1 Ordered Sets on the Upstream Port Lane 0:

a. In the case where the Upstream Port can only support a x8 or x1 Link and the Upstream Port can support Lane reversal. The Upstream Port will assign a Lane 0 to only the received Lane 7 (received Lane number n-1) and the remaining seven Lanes must transmit TS1 Ordered Sets with Link and Lane numbers set to PAD.
b. In the case where the Upstream Port can only support a x8 or x1 Link and the Upstream Port cannot support Lane reversal. No Link can be formed and the Upstream Port will eventually timeout after 2 ms and exit to Detect.

An optional x8 Upstream crosslink Port, which can be divided into two x4 Links, is attached to two x4 Downstream Ports that present the same Link number, and each x4 Downstream Port presents Lane numbers simultaneously that were each numbered 0 to 3. The Upstream Port will have to choose one of these sets of Lane numbers to configure as a Link, and leave the other for a second pass through Configuration.

4.2.6.3.3 Configuration.Lanenum.Accept

4.2.6.3.3.1 Downstream Lanes

- If two consecutive TS1 Ordered Sets are received with non-PAD Link and non-PAD Lane numbers that match all the non-PAD Link and non-PAD Lane numbers (or reversed Lane numbers if Lane reversal is optionally supported) that are being transmitted in Downstream Lane TS1 Ordered Sets, the next state is Configuration.Complete. Note that Retimers are permitted to delay the transition to Configuration.Complete, as described in Section 4.3.8.
  - The Link Bandwidth Management Status and Link Autonomous Bandwidth Status bits of the Link Status register must be updated as follows on a Link bandwidth change if the current transition to Configuration state was from the Recovery state:
    - (a) If the bandwidth change was initiated by the Downstream Port due to reliability issues, the Link Bandwidth Management Status bit is Set.
    - (b) Else if the bandwidth change was not initiated by the Downstream Port and the Autonomous Change bit (Symbol 4 bit 6) in two consecutive received TS1 Ordered Sets is 0b, the Link Bandwidth Management Status bit is Set.
    - (c) Else the Link Autonomous Bandwidth Status bit is Set.
  - The condition of Reversed Lane numbers is defined strictly as the Downstream Lane 0 receiving a TS1 Ordered Set with a Lane number equal to n-1 and the Downstream Lane n-1 receiving a TS1 Ordered Set with a Lane number equal to 0.
  - It is recommended that any possible multi-Lane Link that received an error in a TS1 Ordered Set or lost 128b/130b Block Alignment on a subset of the received Lanes delay the evaluation listed above by an additional two, or more, TS1 Ordered Sets when using 8b/10b encoding, or by an additional 34, or more, TS1 Ordered Sets when using 128b/130b encoding, but must not exceed 1 ms, so as not to prematurely configure a smaller Link than possible.
- If a configured Link can be formed with any subset of the Lanes that receive two consecutive TS1 Ordered Sets with the same transmitted non-PAD Link numbers and any non-PAD Lane numbers, TS1 Ordered Sets are transmitted with the same non-PAD Link numbers and new Lane numbers assigned and the next state is Configuration.Lanenum.Wait.
  - The newly assigned transmitted Lane numbers must range from 0 to m-1, be assigned sequentially only to some continuous grouping of the Lanes that are receiving non-PAD Lane numbers (i.e., Lanes which are not receiving any TS1 Ordered Sets always disrupt a continuous grouping and must not be included in this grouping), must include either Lane 0 or Lane n-1 (largest received Lane number), and m-1 must be equal to or smaller than the largest received Lane number (n-1). Any left over Lanes must transmit TS1 Ordered Sets with the Link and Lane number set to PAD.
  - It is recommended that any possible multi-Lane Link that received an error in a TS1 Ordered Set or lost 128b/130b Block Alignment on a subset of the received Lanes delay the evaluation listed above by an additional two, or more, TS1 Ordered Sets when using 8b/10b encoding, or by an additional 34, or more, TS1 Ordered Sets when using

128b/130b encoding, but must not exceed 1 ms, so as not to prematurely configure a smaller Link than possible.

- The next state is Detect if no Link can be configured or if all Lanes receive two consecutive TS1 Ordered Sets with Link and Lane numbers set to PAD.

4.2.6.3.3.2 Upstream Lanes

- If two consecutive TS2 Ordered Sets are received with non-PAD Link and non-PAD Lane numbers that match all non-PAD Link and non-PAD Lane numbers that are being transmitted in Upstream Lane TS1 Ordered Sets, the next state is Configuration.Complete. Note that Retimers are permitted to delay the transition to Configuration.Complete, as described in Section 4.3.8.
- If a configured Link can be formed with any subset of the Lanes that receive two consecutive TS1 Ordered Sets with the same transmitted non-PAD Link numbers and any non-PAD Lane numbers, TS1 Ordered Sets are transmitted with the same non-PAD Link numbers and new Lane numbers assigned and the next state is Configuration.Lanenum.Wait.
  - The newly assigned transmitted Lane numbers must range from 0 to m-1, be assigned sequentially only to some continuous grouping of Lanes that are receiving non-PAD Lane numbers (i.e., Lanes which are not receiving any TS1 Ordered Sets always disrupt a continuous grouping and must not be included in this grouping), must include either Lane 0 or Lane n-1 (largest received Lane number), and m-1 must be equal to or smaller than the largest received Lane number (n-1). Any left over Lanes must transmit TS1 Ordered Sets with the Link and Lane number set to PAD.
  - It is recommended that any possible multi-Lane Link that received an error in a TS1 Ordered Set or lost 128b/130b Block Alignment on a subset of the received Lanes delay the evaluation listed above by an additional two, or more, TS1 Ordered Sets when using 8b/10b encoding, or by an additional 34, or more, TS1 Ordered Sets when using 128b/130b encoding, but must not exceed 1 ms, so as not to pre-maturely configure a smaller Link than possible.
- The next state is Detect if no Link can be configured or if all Lanes receive two consecutive TS1 Ordered Sets with Link and Lane numbers set to PAD.

4.2.6.3.4 Configuration.Lanenum.Wait

4.2.6.3.4.1 Downstream Lanes

- The next state is Configuration.Lanenum.Accept if any of the Lanes that detected a Receiver during Detect receive two consecutive TS1 Ordered Sets which have a Lane number different from when the Lane first entered Configuration.Lanenum.Wait, and not all the Lanes' Link numbers are set to PAD or two consecutive TS1 Ordered Sets have been received on all Lanes, with Link and Lane numbers that match what is being transmitted on all Lanes.

  The Upstream Lanes are permitted delay up to 1 ms before transitioning to Configuration.Lanenum.Accept.

  The reason for delaying up to 1 ms before transitioning is to prevent received errors or skew between Lanes affecting the final configured Link width.

  The condition of requiring reception of any Lane number different from when the Lane(s) first entered Configuration.Lanenum.Wait is necessary in order to allow the two Ports to settle on an agreed upon Link width. The exact meaning of the statement "any of the Lanes receive two consecutive TS1 Ordered Sets, which have a Lane number different from when the Lane first entered Configuration.Lanenum.Wait" requires that a Lane number must have changed from when the Lanes most recently entered Configuration.Lanenum.Wait before a transition to Configuration.Lanenum.Accept can occur.

- The next state is Detect after a 2 ms timeout or if all Lanes receive two consecutive TS1 Ordered Sets with Link and Lane numbers set to PAD.

4.2.6.3.4.2 Upstream Lanes

- The next state is Configuration.Lanenum.Accept

  (a) If any of the Lanes receive two consecutive TS1 Ordered Sets that have a Lane number different from when the Lane first entered Configuration.Lanenum.Wait, and not all the Lanes' Link numbers are set to PAD

  or

  (b) If any Lane receives two consecutive TS2 Ordered Sets

- The next state is Detect after a 2 ms timeout or if all Lanes receive two consecutive TS1 Ordered Sets with Link and Lane numbers set to PAD.

4.2.6.3.5 Configuration.Complete

A device is allowed to change the supported data rates and upconfigure capability that it advertises when it enters this substate, but it must not change those values while in this substate.

4.2.6.3.5.1 Downstream Lanes

- TS2 Ordered Sets are transmitted using Link and Lane numbers that match the received TS1 Ordered Set Link and Lane numbers.

- The Upconfigure Capability bit of the TS2 Ordered Sets is permitted to be set to 1b to indicate that the Port is capable of supporting a x1 Link on the currently assigned Lane 0 and up-configuring the Link while LinkUp = 1b. Advertising this capability is optional.

- N_FTS must be noted for use in L0s when leaving this state.
- When using 8b/10b encoding, Lane-to-Lane de-skew must be completed when leaving this state.
- Scrambling is disabled if all configured Lanes have the Disable Scrambling bit asserted in two consecutively received TS2 Ordered Sets.
  - The Port that is sending the Disable Scrambling bit on all of the configured Lanes must also disable scrambling. Scrambling can only be disabled when using 8b/10b encoding.
- The next state is Configuration.Idle immediately after all Lanes that are transmitting TS2 Ordered Sets receive eight consecutive TS2 Ordered Sets with matching Lane and Link numbers (non-PAD) and identical data rate identifiers (including identical Link Upconfigure Capability (Symbol 4 bit 6)), and 16 TS2 Ordered Sets are sent after receiving one TS2 Ordered Set. Implementations with the Retimer Presence Detect Supported bit of the Link Capabilities 2 register set to 1b must also receive the eight consecutive TS2 Ordered Sets with identical Retimer Present (Symbol 5 bit 4) when the data rate is 2.5 GT/s. Implementations with Two Retimers Presence Detect Supported bit of the Link Capabilities 2 register set to 1b must also receive the eight consecutive TS2 Ordered Sets with identical Retimer Present (Symbol 5 bits 5:4) when the data rate is 2.5 GT/s.
  - If the data rate of operation is 2.5 GT/s:
    - If the Retimer Presence Detect Supported bit of the Link Capabilities 2 register is set to 1b and any Configured Lane received the Retimer Present bit set to 1b in the eight consecutively received TS2 Ordered Sets, then the Retimer Presence Detected bit must be set to 1b in the Link Status 2 Register otherwise the Retimer Presence Detected bit must be set to 0b in the Link Status 2 Register.
    - If the Two Retimers Presence Detect Supported bit of the Link Capabilities 2 register is set to 1b and any configured Lane received the Two Retimers Present bit set to 1b in the eight consecutively received TS2 Ordered Sets then the Two Retimers Presence Detected bit must be set to 1b in the Link Status 2 Register, otherwise the Two Retimers Presence Detected bit must be set to 0b.
  - If the device supports greater than 2.5 GT/s data rate, it must record the data rate identifier received on any configured Lane of the Link. This will override any previously recorded value. A variable to track speed change in recovery state, changed_speed_recovery, is reset to 0b.
  - If the device sends TS2 Ordered Sets with the Link Upconfigure Capability (Symbol 4 bit 6) set to 1b, and receives eight consecutive TS2 Ordered Sets with the Link Upconfigure Capability bit set to 1b, the variable upconfigure_capable is set to 1b, else it is reset to 0b.

- All remaining Lanes that are not part of the configured Link are no longer associated with the LTSSM in progress and must:

  i. Be associated with a new LTSSM if this optional feature is supported.

  or

  ii. All Lanes that cannot be associated with an optional new LTSSM must transition to Electrical Idle.[56] Those Lanes that formed a Link up to the L0 state, and LinkUp has been 1b since then, but are not a part of the currently configured Link, must be associated with the same LTSSM if the LTSSM advertises Link width upconfigure capability. It is recommended that the Receiver terminations of these Lanes be left on. If they are not left on, they must be turned on when the LTSSM enters the Recovery.RcvrCfg substate until it reaches the Configuration.Complete substate if upconfigure_capable is set to 1b to allow for potential Link width upconfiguration. Any Lane that was not part of the LTSSM during the initial Link training through L0 cannot become a part of the LTSSM as part of the Link width upconfiguration process.

  - In the case of an optional crosslink, the Receiver terminations are required to meet $Z_{RX\text{-}HIGH\text{-}IMP\text{-}DC\text{-}POS}$ and $Z_{RX\text{-}HIGH\text{-}IMP\text{-}DC\text{-}NEG}$ (see Table 8-11).
  - These Lanes must be re-associated with the LTSSM immediately after the LTSSM in progress transitions back to Detect.
  - An EIOS does not need to be sent before transitioning to Electrical Idle, and the transition to Electrical Idle does not need to occur on a Symbol or Ordered Set boundary.

- After a 2 ms timeout:

  - The next state is Detect if the current data rate is 2.5 GT/s or 5.0 GT/s.
  - The next state is Configuration.Idle if the idle_to_rlock_transitioned variable is less than FFh and the current data rate is 8.0 GT/s or higher.

  i. The changed_speed_recovery variable is reset to 0b.

  ii. Lanes that are not part of the configured Link are no longer associated with the LTSSM in progress and must meet requirement (i) or (ii) specified above for the non-timeout transition to Configuration.Idle.

  iii. The upconfigure_capable variable is permitted, but not required, to be updated if at least one Lane received eight consecutive TS2 Ordered Sets with matching Lane and Link numbers (non-PAD). If updated, the upconfigure_capable variable is set to 1b when the transmitted and received Link Upconfigure Capability bits are 1b, else it is reset to 0b.

- Else the next state is Detect.

---

[56] The common mode being driven does not need to meet the Absolute Delta Between DC Common Mode During L0 and Electrical Idle ($V_{TX\text{-}CM\text{-}DC\text{-}ACTIVE\text{-}IDLE\text{-}DELTA}$) specification (see Table 8-7).

4.2.6.3.5.2 Upstream Lanes

- TS2 Ordered Sets are transmitted using Link and Lane numbers that match the received TS2 Link and Lane numbers.
  - The Upconfigure Capability bit of the TS2 Ordered Sets is permitted to be set to 1b to indicate that the Port is capable of supporting a x1 Link on the currently assigned Lane 0 and up-configuring the Link while LinkUp = 1b. Advertising this capability is optional.
- N_FTS must be noted for use in L0s when leaving this state.
- When using 8b/10b encoding, Lane-to-Lane de-skew must be completed when leaving this state.
- Scrambling is disabled if all configured Lanes have the Disable Scrambling bit asserted in two consecutively received TS2 Ordered Sets.
  - The Port that is sending the Disable Scrambling bit on all of the configured Lanes must also disable scrambling. Scrambling can only be disabled when using 8b/10b encoding.
- The next state is Configuration.Idle immediately after all Lanes that are transmitting TS2 Ordered Sets receive eight consecutive TS2 Ordered Sets with matching Lane and Link numbers (non-PAD) and identical data rate identifiers (including identical Link Upconfigure Capability (Symbol 4 bit 6)), and 16 consecutive TS2 Ordered Sets are sent after receiving one TS2 Ordered Set. Implementations with the Retimer Presence Detect Supported bit of the Link Capabilities 2 register set to 1b must also receive the eight consecutive TS2 Ordered Sets with identical Retimer Present (Symbol 5 bit 4) when the data rate is 2.5 GT/s. Implementations with Two Retimers Presence Detect Supported bit of the Link Capabilities 2 register set to 1b must also receive the eight consecutive TS2 Ordered Sets with identical Retimer Present (Symbol 5 bits 5:4) when the data rate is 2.5 GT/s.
  - If the data rate of operation is 2.5 GT/s:
    - If the Retimer Presence Detect Supported bit of the Link Capabilities 2 register is set to 1b and any Configured Lane received the Retimer Present bit set to 1b in the eight consecutively received TS2 Ordered Sets, then the Retimer Presence Detected bit must be set to 1b in the Link Status 2 Register otherwise the Retimer Presence Detected bit must be set to 0b in the Link Status 2 Register.
    - If the Two Retimers Presence Detect Supported bit of the Link Capabilities 2 register is set to 1b and any configured Lane received the Two Retimers Present bit set to 1b in the eight consecutively received TS2 Ordered Sets then the Two 35 Retimers Presence Detected bit must be set to 1b in the Link Status 2 Register, otherwise the Two Retimers Presence Detected bit must be set to 0b.
  - If the device supports greater than 2.5 GT/s data rate, it must record the data rate identifier received on any configured Lane of the Link. This will override any previously recorded value. A variable to track speed change in recovery state, changed_speed_recovery, is reset to 0b.
  - If the device sends TS2 Ordered Sets with the Link Upconfigure Capability (Symbol 4 bit 6) set to 1b, as well as receives eight consecutive TS2 Ordered Sets with the Link

Upconfigure Capability bit set to 1b, the variable upconfigure_capable is set to 1b, else it is reset to 0b.

- All remaining Lanes that are not part of the configured Link are no longer associated with the LTSSM in progress and must:

  i. Optionally be associated with a new crosslink LTSSM if this feature is supported.

  or

  ii. All remaining Lanes that are not associated with a new crosslink LTSSM must transition to Electrical Idle,[57] and Receiver terminations are required to meet $Z_{RX\text{-}HIGH\text{-}IMP\text{-}DC\text{-}POS}$ and $Z_{RX\text{-}HIGH\text{-}IMP\text{-}DC\text{-}NEG}$ (see Table 8-11). Those Lanes that formed a Link up to the L0 state, and LinkUp has been 1b since then, but are not a part of the currently configured Link, must be associated with the same LTSSM if the LTSSM advertises Link width upconfigure capability. It is recommended that the Receiver terminations of these Lanes be left on. If they are not left on, they must be turned on when the LTSSM enters the Recovery.RcvrCfg substate until it reaches the Configuration.Complete substate if upconfigure_capable is set to 1b to allow for potential Link width upconfiguration. Any Lane that was not part of the LTSSM during the initial Link training through L0 cannot become a part of the LTSSM as part of the Link width upconfiguration process.

  - These Lanes must be re-associated with the LTSSM immediately after the LTSSM in progress transitions back to Detect.
  - EIOS does not need to be sent before transitioning to Electrical Idle, and the transition to Electrical Idle does not need to occur on a Symbol or Ordered Set boundary.

- After a 2 ms timeout:
  - The next state is Detect if the current data rate is 2.5 GT/s or 5.0 GT/s.

The next state is Configuration.Idle if the idle_to_rlock_transitioned variable is less than FFh and the current data rate is 8.0 GT/s or higher.

i. The changed_speed_recovery variable is reset to 0b.

ii. Lanes that are not part of the configured Link are no longer associated with the LTSSM in progress and must meet requirement (i) or (ii) specified above for the non-timeout transition to Configuration.Idle.

iii. The upconfigure_capable variable is permitted, but not required, to be updated if at least one Lane received eight consecutive TS2 Ordered Sets with matching Lane and Link numbers (non-PAD). If updated, the upconfigure_capable variable is set to 1b when the transmitted and received Link Upconfigure Capability bits are 1b, else it is reset to 0b.

Else the next state is Detect.

[57] The common mode being driven does not need to meet the Absolute Delta Between DC Common Mode During L0 and Electrical Idle ($V_{TX\text{-}CM\text{-}DC\text{-}ACTIVE\text{-}IDLE\text{-}DELTA}$) specification (see Table 8-7).

4.2.6.3.6 Configuration.Idle

- When using 8b/10b encoding, the Transmitter sends Idle data Symbols on all configured Lanes.
- When using 128b/130b encoding:
  - If the data rate is 8.0 GT/s, the Transmitter sends one SDS Ordered Set on all configured Lanes to start a Data Stream and then sends Idle data Symbols on all configured Lanes. The first Idle data Symbol transmitted on Lane 0 is the first Symbol of the Data Stream.
  - If the data rate is 16.0 GT/s, the Transmitter sends one Control SKP Ordered Set followed immediately by one SDS Ordered Set on all configured Lanes to start a Data Stream and then sends Idle data Symbols on all configured Lanes. The first Idle data Symbol transmitted on Lane 0 is the first Symbol of the Data Stream.
- Receiver waits for Idle data.
- LinkUp = 1b
- When using 8b/10b encoding, the next state is L0 if eight consecutive Symbol Times of Idle data are received on all configured Lanes and 16 Idle data Symbols are sent after receiving one Idle data Symbol.
  - If software has written a 1b to the Retrain Link bit in the Link Control register since the last transition to L0 from Recovery or Configuration, the Downstream Port must set the Link Bandwidth Management Status bit of the Link Status register to 1b.
- When using 128b/130b encoding, next state is L0 if eight consecutive Symbol Times of Idle data are received on all configured Lanes, 16 Idle data Symbols are sent after receiving one Idle data Symbol, and this state was not entered by a timeout from Configuration.Complete.
  - The Idle data Symbols must be received in Data Blocks.
  - Lane-to-Lane de-skew must be completed before Data Stream processing starts.
  - If software has written a 1b to the Retrain Link bit in the Link Control register since the last transition to L0 from Recovery or Configuration, the Downstream Port must set the Link Bandwidth Management Status bit of the Link Status register to 1b.
  - The idle_to_rlock_transitioned variable is reset to 00h on transition to L0.

- Otherwise, after a minimum 2 ms timeout:
  - If the idle_to_rlock_transitioned variable is less than FFh, the next state is Recovery.RcvrLock.
  - On transition to Recovery.RcvrLock:
    - If the data rate is 8.0 GT/s or higher, the idle_to_rlock_transitioned variable is incremented by 1.
    - If the data rate is 2.5 GT/s or 5.0 GT/s, the idle_to_rlock_transitioned variable is set to FFh.
  - Else the next state is Detect.

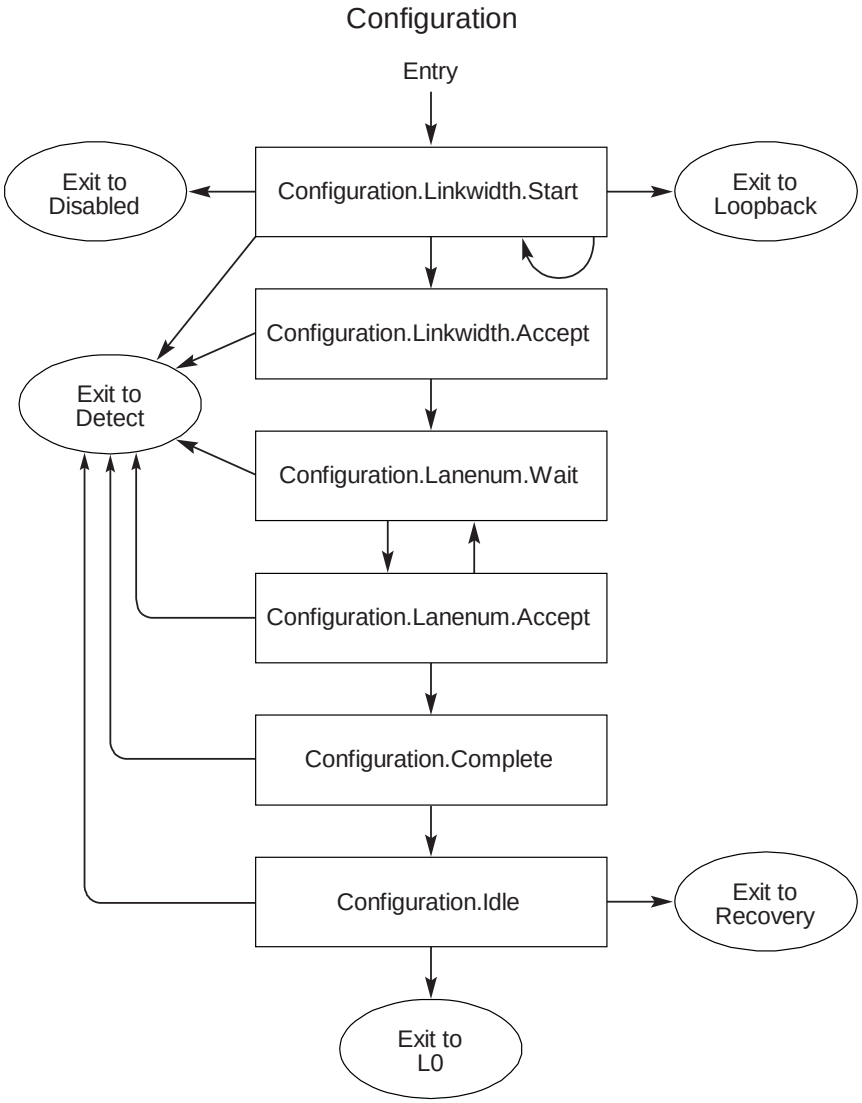


Figure 4-26: Configuration Substate Machine

4.2.6.4 *Recovery*

The Recovery substate machine is shown in Figure 4-27

4.2.6.4.1 Recovery.RcvrLock

If the Link is operating at a data rate of 8.0 GT/s or higher, a Receiver must consider any TS1 or TS2 Ordered Set to be received only after it obtains Block Alignment in that Lane. If entry to this substate is from L1 or Recovery.Speed or L0s, the Block Alignment must be obtained after exiting Electrical Idle condition. If entry to this substate is from L0, the Block Alignment must be obtained after the end of the last Data Stream.

- If the data rate of operation is 8.0 GT/s or higher:
  - If the start_equalization_w_preset variable is set to 1b:
    - An Upstream Port operating at 8.0 GT/s must use the 8.0 GT/s Transmitter preset values it registered from the received eight consecutive EQ TS2 Ordered Sets in Recovery.RcvrCfg in its Transmitter setting as soon as it starts transmitting in 8.0 GT/s data rate and ensure that it meets the preset definition in Chapter 8. Lanes that received a Reserved or unsupported Transmitter preset value must use an implementation specific method to choose a supported Transmitter preset setting for use as soon it starts transmitting at 8.0 GT/s. The Upstream Port may optionally use the Receiver preset hint value suggested by the Downstream Port in those EQ TS2 Ordered Sets.
    - An Upstream Port operating at 16.0 GT/s must use the 16.0 GT/s Transmitter preset values it registered from the received eight consecutive 8GT EQ TS2 Ordered Sets in Recovery.RcvrCfg in its Transmitter setting as soon as it starts transmitting in 16.0 GT/s data rate and ensure that it meets the preset definition in Section 4.3. Lanes that received a Reserved or unsupported Transmitter preset value must use an implementation specific method to choose a supported Transmitter preset setting for use as soon as it starts transmitting at 16.0 GT/s.
    - A Downstream Port operating at 8.0 GT/s must use the Downstream Port 8.0 GT/s Transmitter Preset field defined in the Lane Equalization Control Register entry for each Lane as soon as it starts transmitting in 8.0 GT/s data rate and ensure that it meets the preset definition in Chapter 8. Lanes that have a Reserved or unsupported Transmitter Preset value in the Downstream Port 8.0 GT/s Transmitter Preset field of their associated 8.0 GT/s Lane Equalization Control Register entry must use an implementation specific method to choose a supported Transmitter preset setting for use as soon as it starts transmitting at 8.0 GT/s. The Downstream Port may optionally use the Downstream Port 8.0 GT/s Receiver Preset Hint field defined in the Lane Equalization Control Register entry for each of its Receivers corresponding to the Lane, if they are not Reserved values.
    - A Downstream Port operating at 16.0 GT/s must use 16.0 GT/s Transmitter preset settings as soon as it starts transmitting at the 16.0 GT/s data rate. The

16.0 GT/s Transmitter preset settings must be chosen, for each configured Lane, as follows:

1. If eight consecutive 8GT EQ TS2 Ordered Sets were received with supported 16.0 GT/s Transmitter Preset values in the most recent transition through Recovery.RcvrCfg, the Transmitter Preset value from those 8GT EQ TS2 Ordered Sets must be used.
2. Else, if the Transmitter Preset value defined in the 16.0 GT/s Downstream Port Transmitter Preset field of the 16.0 GT/s Lane Equalization Control Register entry is supported: the Transmitter Preset value from the 16.0 GT/s Lane Equalization Control Register entry must be used.
3. Else, use an implementation specific method to choose a supported Transmitter preset setting.

The Downstream Port must ensure that it meets the preset definition in Section 4.3.

- Next state is Recovery.Equalization.

- Else:
  - The Transmitter must use the coefficient settings agreed upon at the conclusion of the last equalization procedure
  - If this substate was entered from Recovery.Equalization, in the transmitted TS1 Ordered Sets, a Downstream Port must set the Pre-cursor, Cursor, and Post-cursor Coefficient fields to the current Transmitter settings, and if the last accepted request in Phase 2 of Recovery.Equalization was a preset request, it must set the Transmitter Preset field to the accepted preset of that request.
  - It is recommended that in this substate, in the transmitted TS1 Ordered Sets, all Ports set the Pre-cursor, Cursor, and Post-cursor Coefficient fields to the current Transmitter settings, and set the Transmitter Preset field to the most recent preset that the Transmitter settings were set to.
  - An Upstream Port that receives eight consecutive TS1 Ordered Sets on all configured Lanes with the following characteristics must transition to Recovery.Equalization
    - Link and Lane numbers in the received TS1 Ordered Sets match with the Link and Lane numbers in the transmitted TS1 Ordered Sets on each Lane
    - speed_change bit is equal to 0b
    - EC bits not equal to 00b

IMPLEMENTATION NOTE

Redoing Equalization

A Downstream Port may use this provision to redo some parts of the Transmitter Equalization process using software help or some other implementation specific means while ensuring no transactions are in flight on the Link to avoid any timeouts.

- Next state for a Downstream Port is Recovery.Equalization if directed and Recovery.RcvrLock was not entered from Configuration.Idle or Recovery.Idle. The Port must ensure that no more than 2 TS1 Ordered Sets with EC=00b are transmitted due to being in Recovery.RcvrLock before starting to transmit the TS1 Ordered Sets required by Recovery.Equalization.

- Transmitter sends TS1 Ordered Sets on all configured Lanes using the same Link and Lane numbers that were set after leaving Configuration. The speed_change bit (bit 7 of the Data Rate Identifier Symbol in TS1 Ordered Set) must be set to 1b if the directed_speed_change variable is set to 1b. The directed_speed_change variable is set to 1b if any configured Lane receives eight consecutive TS1 Ordered Sets with the speed_change bit set to 1b. Only those data rates greater than 2.5 GT/s should be advertised that can be supported reliably. The N_FTS value in the TS1 Ordered Set transmitted reflects the number at the current speed of operation. A device is allowed to change the supported data rates that it advertises when it enters this substate. A Downstream Port that intends to redo equalization with a data rate change from 2.5 GT/s or 5.0 GT/s to 8.0 GT/s must:
  - Send EQ TS1 Ordered Sets with the speed_change bit set to 1b and advertising the 8.0 GT/s Data Rate Identifier.
  - If the equalization redo attempt is initiated by the hardware as described in Section 4.2.3, then hardware must ensure that the Data Rate is 2.5 GT/s or 5.0 GT/s before initiating the attempt.
  - If the equalization redo attempt is initiated by the software mechanism as described in Section 4.2.3, then software must ensure that the Data Rate is 2.5 GT/s or 5.0 GT/s before initiating the attempt.

An Upstream Port must advertise 8.0 GT/s data rate support in the TS2 Ordered Sets it transmits in Recovery.RcvrCfg, and optionally in the TS1 Ordered Sets it transmits in this substate, if:

- The eight consecutive Ordered Sets it receives are EQ TS1 or EQ TS2 Ordered Sets with the speed_change bit set to 1b and 8.0 GT/s data rate support is advertised by the Downstream Port, unless the Upstream Port has determined that a problem unrelated to equalization prevents it from operating reliably at 8.0 GT/s.

A Downstream Port that intends to redo equalization with a data rate change from 8.0 GT/s to 16.0 GT/s must:

- Send TS1 Ordered Sets with the Equalization Redo bit set to 1b, the speed_change bit set to 1b, and advertising the 16.0 GT/s Data Rate Identifier.
- If the equalization redo attempt is initiated by the hardware as described in Section 4.2.3, then hardware must ensure that the Data Rate is 8.0 GT/s before initiating the attempt.

- If the equalization redo attempt is initiated by the software mechanism as described in Section 4.2.3, then software must ensure that the Data Rate is 8.0 GT/s before initiating the attempt.

An Upstream Port must advertise 16.0 GT/s data rate support in the TS2 Ordered Sets it transmits in Recovery.RcvrCfg, and optionally in the TS1 Ordered Sets it transmits in this substate, if:

- The eight consecutive Ordered Sets it receives are TS1 Ordered Sets with the Equalization Redo bit set to 1b or 8GT EQ TS2 Ordered Sets with the speed_change bit set to 1b, unless the Upstream Port has determined that a problem unrelated to 16.0 GT/s equalization prevents it from operating reliably at 16.0 GT/s.

Under other conditions, a device must not change the supported data rate values either in this substate or while in the Recovery.RcvrCfg or Recovery.Equalization substates. The successful_speed_negotiation variable is reset to 0b upon entry to this substate.

IMPLEMENTATION NOTE

Handling a Request to Advertise 8.0 GT/s Data Rate Identifier

If an Upstream Port that is not advertising 8.0 GT/s Data Rate Identifiers receives EQ TSs with 8.0 GT/s Data Rate Identifiers and with the speed_change bit set in Recovery.RcvrLock, that indicates that the Downstream Port is attempting to switch the Link Speed to 8.0 GT/s in order to perform the 8.0 GT/s Link Equalization Procedure. If for some reason the Upstream Port is unable or unwilling to switch to advertising 8.0 GT/s Data Rate Identifiers in the TS2 Ordered Sets it transmits once it transitions to Recovery.RcvrCfg, the 8.0 GT/s Link Equalization Procedure will not be performed in the current tenure in Recovery. This may cause the Downstream Port to permanently abandon its attempt to change the link speed to 8.0 GT/s and perform the 8.0 GT/s Link Equalization Procedure, resulting in an operational link speed of less than 8.0 GT/s until after the link transitions through Detect and is re-trained. It is recommended that if an Upstream Port is for some temporary reason unable or unwilling to switch to advertising 8.0 GT/s Data Rate Identifiers in the condition described above, and does not intend to prohibit the Link from operating at 8.0 GT/s, that it perform one of the following two actions below as soon as is reasonable for it to do so:

- If the Upstream Port supports the Quiesce Guarantee mechanism for performing the Link Equalization Procedure, enter Recovery and advertise 8.0 GT/s Data Rate Identifiers with the speed_change bit set to 1b in the TSs that it sends. If Recovery.Equalization is not entered after changing speed to 8.0 GT/s and before entering Recovery.RcvrCfg at 8.0 GT/s (the Downstream Port did not direct an entry to Recovery.Equalization), it should set the Request Equalization and Quiesce Guarantee bits to 1b in the TS2 Ordered Sets sent at 8.0 GT/s in Recovery.RcvrCfg in order to request the Downstream Port to initiate the Link Equalization Procedure.
- Enter Recovery and advertise 8.0 GT/s Data Rate Identifiers with the speed_change bit cleared to 0b. The Downstream Port may then later initiate a speed change to 8.0 GT/s and perform the Link Equalization Procedure, though there is no guarantee that it will do so.

The process for handling a request to advertise 16.0 GT/s Data Rate Identifier is similar to 8.0 GT/s Data Rate Identifier with 16.0 GT/s Data Rate Identifier substituting 8.0 GT/s Data Rate Idenitifier and 8GT EQ TS2s substituting EQ TSs.

An Upstream Port must set the Selectable De-emphasis bit (bit 6 of Symbol 4) of the TS1 Ordered Sets it transmits to match the desired de-emphasis level at 5.0 GT/s. The mechanism an Upstream Port may adopt to request a de-emphasis level if it chooses to do so is implementation specific. It must also be noted that since the Upstream Port's request may not reach the Downstream Port due to bit errors in the TS1 Ordered Sets, the Upstream Port may attempt to re-request the desired de-emphasis level in subsequent entries to Recovery state when speed change is requested. If the Downstream Port intends to use the Upstream Port's de-emphasis information in Recovery.RcvrCfg, then it must record the value of the Selectable De-emphasis bit received in this state.

The Transmit Margin field of the Link Control 2 register is sampled on entry to this substate and becomes effective on the transmit package pins within 192 ns of entry to this substate and

remains effective until a new value is sampled on a subsequent entry to this substate from L0, L0s, or L1.

- After activating any inactive Lane, the Transmitter must wait for its TX common mode to settle before exiting Electrical Idle and transmitting the TS1 Ordered Sets with the following exceptions.
- When exiting from the L1.2 L1 PM Substate, common mode is permitted to be established passively during L1.0, and actively during Recovery. In order to ensure common mode has been established in Recovery.RcvrLock, the Downstream Port must maintain a timer, and the Downstream Port must not send TS2 training sequences until a minimum of $T_{COMMONMODE}$ has elapsed since the Downstream Port has started both transmitting and receiving TS1 training sequences. See Section 5.5.3.3.
- Implementations must note that the voltage levels may change after an early bit lock and Symbol or Block alignment since the new Transmit Margin field becomes effective within 192 ns after the other side enter Recovery.RcvrLock. The Receiver needs to reacquire bit lock and Symbol or Block alignment under those conditions.
    a. Note: The directed_speed_change variable is set to 1b in L0 or L1 state for the side that is initiating a speed change. For the side that is not initiating a speed change, this bit is Set in this substate if the received TS Ordered Sets have the speed change bit Set. This bit is reset to 0b in the Recovery.Speed substate.
    b. A device must accept all good TLPs and DLLPs it receives after entering this substate from L0 prior to receiving the first TS Ordered Set. If operating with 128b/130b encoding, any received TLPs and DLLPs are subject to the framing rules for 128b/130b encoding in Section 4.2.2.3.

- Next state is Recovery.RcvrCfg if eight consecutive TS1 or TS2 Ordered Sets are received on all configured Lanes with the same Link and Lane numbers that match what is being transmitted on those same Lanes and the speed_change bit is equal to the directed_speed_change variable and the EC field is 00b in all the consecutive TS1 Ordered Sets if the current data rate is 8.0 GT/s or higher.
    - If the Extended Synch bit is Set, the Transmitter must send a minimum of 1024 consecutive TS1 Ordered Sets before transitioning to Recovery.RcvrCfg.
    - If this substate was entered from Recovery.Equalization, the Upstream Port must evaluate the equalization coefficients or preset received by all Lanes that receive eight TS1 Ordered Sets and note whether they are different from the final set of coefficients or preset that was accepted in Phase 2 of the equalization process. Note: Mismatches are reported in Recovery.RcvrCfg by setting the Request Equalization bit of TS2 Ordered Sets.
- Otherwise, after a 24 ms timeout:
    - Next state is Recovery.RcvrCfg if the following two conditions are true:
        - Eight consecutive TS1 or TS2 Ordered Sets are received on any configured Lane with the same Link and Lane numbers that match what is being transmitted on the same Lane and the speed_change bit equal to 1b.

    - Either the current data rate of operation is greater than 2.5 GT/s; or 5.0 GT/s or greater data rate identifiers are set in both the transmitted TS1 and the (eight consecutive) received TS1 or TS2 Ordered Sets.

- Else the next state is Recovery.Speed if the speed of operation has not changed to a mutually negotiated data rate since entering Recovery from L0 or L1 (i.e., changed_speed_recovery = 0b) and the current speed of operation is greater than 2.5 GT/s. The new data rate to operate after leaving Recovery.Speed will be at 2.5 GT/s. Note: This indicates that the Link was unable to operate at the current data rate (greater than 2.5 GT/s) and the Link will operate at the 2.5 GT/s data rate.
- Else the next state is Recovery.Speed if the operating speed has been changed to a mutually negotiated data rate since entering Recovery from L0 or L1 (changed_speed_recovery = 1b; i.e., the arc to this substate has been taken from Recovery.Speed). The new data rate to operate after leaving Recovery.Speed is reverted back to the speed it was when Recovery was entered from L0 or L1.

  Note: This indicates that the Link was unable to operate at the new negotiated data rate and will revert back to the old data rate with which it entered Recovery from L0 or L1.
- Else the next state is Configuration and the directed_speed_change variable is reset to 0b if any of the configured Lanes that are receiving a TS1 or TS2 Ordered Set have received at least one TS1 or TS2 Ordered Set with Link and Lane numbers that match what is being transmitted on those same Lanes and the operating speed has not changed to a mutually negotiated data rate (i.e., changed_speed_recovery = 0b) since entering Recovery and at least one of the following conditions is true:
    - The directed_speed_change variable is equal to 0b and the speed_change bit on the received TS1 or TS2 Ordered Set is equal to 0b.
    - The current data rate of operation is 2.5 GT/s and 2.5 GT/s data rate is the highest commonly advertised data rate among the transmitted TS1 Ordered Sets and the received TS1 or TS2 Ordered Set(s).
- Otherwise, the next state is Detect.

IMPLEMENTATION NOTE

Example Showing Speed Change Algorithm Between 2.5 GT/s and 5.0 GT/s

Suppose a Link connects two greater than 5.0 GT/s capable components, A and B. The Link comes up to the L0 state in 2.5 GT/s data rate. Component A decides to change the speed to greater than 5.0 GT/s, sets the directed_speed_change variable to 1b and enters Recovery.RcvrLock from L0. Component A sends TS1 Ordered Sets with the speed_change bit set to 1b and advertises the 2.5 GT/s, 5.0 GT/s, and 8.0 GT/s data rates. Component B sees the first TS1 in L0 state and enters Recovery.RcvrLock state. Initially, component B sends TS1s with the speed_change set to 0b. Component B will start sending the speed_change indication in its TS1 after it receives eight consecutive TS1 Ordered Sets from component A and advertises all of the data rates it can support. Component B will enter Recovery.RcvrCfg from where it will enter Recovery.Speed. Component A will wait for eight consecutive TS1/TS2 with speed_change bit set from component B before moving to Recovery.RcvrCfg and on to Recovery.Speed. Both component A and component B enter Recovery.Speed and record 8.0 GT/s as the maximum speed they can operate with. The directed_speed_change variable will be reset to 0b when in Recovery.Speed. When they enter Recovery.RcvrLock from Recovery.Speed, they will operate at 8.0 GT/s and send TS1s with speed_change set to 0b. If both sides work well at 8.0 GT/s, they will continue on to Recovery.RcvrCfg and enter L0 through Recovery.Idle at 8.0 GT/s. However, if component B fails to achieve Symbol lock, it will timeout in Recovery.RcvrLock and enters Recovery.Speed. Component A would have moved on to Recovery.RcvrCfg but would see the Electrical Idle after receiving TS1s at 8.0 GT/s after component B enters Recovery.Speed. This will cause component A to move to Recovery.Speed. After entering Recovery.Speed for the second time, both sides will revert back to the speed they operated with prior to entering the Recovery state (2.5 GT/s). Both sides will enter L0 from Recovery in 2.5 GT/s. Component A may initiate the directed_speed_change variable for a second time, requesting 8.0 GT/s data rate in its Data Rate Identifier, go through the same steps, fail to establish the 8.0 GT/s data rate and go back to L0 in 2.5 GT/s data rate. On the third attempt, however, component A may decide to only advertise 2.5 GT/s and 5.0 GT/s data rates and successfully establish the Link at 5.0 GT/s data rate and enter L0 at that speed. However, if either side entered Detect, that side should advertise all of the data rates it can support, since there may have been a hot plug event.

4.2.6.4.2 Recovery.Equalization

Transmitter sends TS1 Ordered Sets on all configured Lanes using the same Link and Lane numbers that were set after leaving Configuration.

4.2.6.4.2.1 Downstream Lanes

Upon entry to this substate:

- Current phase is Phase 1
  - If the data rate of operation is 8.0 GT/s:
    - The Equalization 8.0 GT/s Phase 1 Successful, Equalization 8.0 GT/s Phase 2 Successful, Equalization 8.0 GT/s Phase 3 Successful, Link Equalization Request 8.0 GT/s, and Equalization 8.0 GT/s Complete bits of the Link Status 2 register and the Perform Equalization bit of the Link Control 3 register are all set to 0b
    - The equalization_done_8GT_data_rate variable is set to 1b
  - If the data rate of operation is 16.0 GT/s:
    - The Equalization 16.0 GT/s Phase 1 Successful, Equalization 16.0 GT/s Phase 2 Successful, Equalization 16.0 GT/s Phase 3 Successful, Link Equalization Request 16.0 GT/s, and Equalization 16.0 GT/s Complete bits of the 16.0 GT/s Status register and the Perform Equalization bit of the Link Control 3 registers are all set to 0b
    - The equalization_done_16GT_data_rate variable is set to 1b
- The start_equalization_w_preset variable is set to 0b

4.2.6.4.2.1.1 Phase 1 of Transmitter Equalization

- Transmitter sends TS1 Ordered Sets using the Transmitter preset settings for the current data rate of operation. In the TS1 Ordered Sets, the EC field is set to 01b, the Transmitter Preset field of each Lane is set to the value of its corresponding Transmitter preset setting for the current data rate, the FS, LF, and the Post-cursor Coefficient fields are set to values corresponding to the Lane's Transmitter preset field. The Transmitter preset settings, for each configured Lane, must be chosen as follows:

1. If eight consecutive 8GT EQ TS2 Ordered Sets were received with supported Transmitter preset values in the most recent transition through Recovery.RcvrCfg and the current data rate of operation is 16.0 GT/s, the Transmitter preset value requested in the 8GT EQ TS2 Ordered Sets must be used.
2. Else, if the Transmitter preset setting specified by the Downstream Port 8.0 GT/s Transmitter Preset field of the Lane Equalization Control Register entry (for operation at the 8.0 GT/s data rate) or the Downstream Port 16.0 GT/s Transmitter Preset field of the 16.0 GT/s Lane Equalization Control Register entry (for operation at the 16.0 GT/s data rate) is a supported value and is not a Reserved value, it must be used.

3. Else, use an implementation specific method to choose a supported Transmitter preset setting for use.

- The Downstream Port is permitted to wait for up to 500 ns after entering Phase 1 before evaluating received information for TS1 Ordered Sets if it needs the time to stabilize its Receiver logic.
- Next phase is Phase 2 if all configured Lanes receive two consecutive TS1 Ordered Sets with EC=01b and the Downstream Port wants to execute Phase 2 and Phase 3.
  - The Receiver must complete its bit lock process and then recognize Ordered Sets within 2 ms after receiving the first bit of the first valid Ordered Set on its Receiver pin.
  - If the data rate is 8.0 GT/s, the Equalization 8.0 GT/s Phase 1 Successful bit of the Link Status 2 register is set to 1b.
  - If the data rate is 16.0 GT/s, the Equalization 16.0 GT/s Phase 1 Successful bit of the 16.0 GT/s Status register is set to 1b.
  - The LF and FS values received in the two consecutive TS1 Ordered Sets must be stored for use during Phase 3, if the Downstream Port wants to adjust the Upstream Port's Transmitter coefficients.
- Else, next state is Recovery.RcvrLock if all configured Lanes receive two consecutive TS1 Ordered Sets with EC=01b and the Downstream Port does not want to execute Phase 2 and Phase 3.
  - If the data rate is 8.0 GT/s, The Equalization 8.0 GT/s Phase 1 Successful, Equalization 8.0 GT/s Phase 2 Successful, Equalization 8.0 GT/s Phase 3 Successful, and Equalization 8.0 GT/s Complete bits of the Link Status 2 register are set to 1b.
  - If the data rate is 16.0 GT/s, The Equalization 16.0 GT/s Phase 1 Successful, Equalization 16.0 GT/s Phase 2 Successful, Equalization 16.0 GT/s Phase 3 Successful, and Equalization 16.0 GT/s Complete bits of the 16.0 GT/s Status register are set to 1b.
  - Note: A transition to Recovery.RcvrLock might be used in the case where the Downstream Port determines that Phase 2 and Phase 3 are not needed based on the platform and channel characteristics.
- Else, next state is Recovery.Speed after a 24 ms timeout.
  - successful_speed_negotiation is set to 0b
  - If the data rate is 8.0 GT/s, the Equalization 8.0 GT/s Complete bit of the Link Status 2 register is set to 1b.
  - If the data rate is 16.0 GT/s, the Equalization 16.0 GT/s Complete bit of the 16.0 GT/s Status register is set to 1b.

4.2.6.4.2.1.2 Phase 2 of Transmitter Equalization

- Transmitter sends TS1 Ordered Sets with EC = 10b and the coefficient settings, set on each Lane independently, as follows:
  - If two consecutive TS1 Ordered Sets with EC=10b have been received since entering Phase 2, or two consecutive TS1 Ordered Sets with EC=10b and a preset or set of

coefficients (as specified by the Use Preset bit) different than the last two consecutive TS1 Ordered Sets with EC=10b:

- If the preset or coefficients requested in the most recent two consecutive TS1 Ordered Sets are legal and supported (see Section 4.2.3):
  - Change the transmitter settings to the requested preset or coefficients such that the new settings are effective at the Transmitter pins within 500 ns of when the end of the second TS1 Ordered Set requesting the new setting was received at the Receiver pin. The change of Transmitter settings must not cause any illegal voltage level or parameter at the Transmitter pin for more than 1 ns.
  - In the transmitted TS1 Ordered Sets, the Transmitter Preset field is set to the requested preset (for a preset request), the Pre-cursor, Cursor, and Post-cursor Coefficient fields are set to the Transmitter settings (for a preset or a coefficients request), and the Reject Coefficient Values bit is Clear.
- Else (the requested preset or coefficients are illegal or unsupported): Do not change the Transmitter settings used, but reflect the requested preset or coefficient values in the transmitted TS1 Ordered Sets and set the Reject Coefficient Values bit to 1b.

- Else: the preset and coefficients currently being used by the Transmitter.

- Next phase is Phase 3 if all configured Lanes receive two consecutive TS1 Ordered Sets with EC=11b.
  - If the data rate is 8.0 GT/s, the Equalization 8.0 GT/s Phase 2 Successful bit of the Link Status 2 register is set to 1b.
  - If the data rate is 16.0 GT/s, the Equalization 16.0 GT/s Phase 2 Successful bit of the 16.0 GT/s Status register is set to 1b.
- Else, next state is Recovery.Speed after a 32 ms timeout with a tolerance of -0 ms and +4 ms
  - successful_speed_negotiation is set to 0b.
  - If the data rate is 8.0 GT/s, The Equalization 8.0 GT/s Complete bit of the Link Status 2 register is set to 1b.
  - If the data rate is 16.0 GT/s, The Equalization 16.0 GT/s Complete bit of the 16.0 GT/s Status register is set to 1b.

4.2.6.4.2.1.3 Phase 3 of Transmitter Equalization

- Transmitter sends TS1 Ordered Sets with EC = 11b
- The Port must evaluate and arrive at the optimal settings independently on each Lane. To evaluate a new preset or coefficient setting that is legal, as per the rules in Section 4.2.3 and Chapter 8:
  - Request a new preset by setting the Transmitter Preset field to the desired value and set the Use Preset bit to 1b. Or, request a new set of coefficients by setting the Pre-cursor,

Cursor, and Post-Cursor Coefficient fields to the desired values and set the Use Preset bit to 0b. Once a request is made, it must be continuously requested for at least 1 μs or until the evaluation of the request is completed, whichever is later.

- Wait for the required time (500 ns plus the roundtrip delay including the logic delays through the Downstream Port) to ensure that, if accepted, the Upstream Port is transmitting using the requested settings. Obtain Block Alignment and then evaluate the incoming Ordered Sets. Note: The Downstream Port may simply ignore anything it receives during this waiting period as the incoming bit stream may be illegal during the transition to the requested settings. Hence the requirement to validate Block Alignment after this waiting period. If Block Alignment cannot be obtained after an implementation specific amount of time (in addition to the required waiting period specified above) it is recommended to proceed to perform receiver evaluation on the incoming bit stream regardless.
- If two consecutive TS1 Ordered Sets are received with the Transmitter Preset field (for a preset request) or the Pre-cursor, Cursor, and Post-Cursor fields (for a coefficients request) identical to what was requested and the Reject Coefficient Values bit is Clear, then the requested setting was accepted and, depending on the results of receiver evaluation, can be considered as a candidate final setting.
- If two consecutive TS1 Ordered Sets are received with the Transmitter Preset field (for a preset request) or the Pre-cursor, Cursor, and Post-Cursor fields (for a coefficients request) identical to what was requested and the Reject Coefficient Values bit is Set, then the requested setting was rejected and must not be considered as a candidate final setting.
- If, after an implementation specific amount of time following the start of receiver evaluation, no consecutive TS1s with the Transmitter Preset field (for a preset request) or the Pre-Cursor, Cursor, and Post-Cursor fields (for a coefficients request) identical to what was requested are received, then the requested setting must not be considered as a candidate final setting.
- The Downstream Port is responsible for setting the Reset EIEOS Interval Count bit in the TS1 Ordered Sets it transmits according to its evaluation criteria and requirements. The Use Preset bit of the received TS1 Ordered Sets must not be used to determine whether a request is accepted or rejected.

IMPLEMENTATION NOTE

Reset EIEOS and Coefficient/ Preset Requests

A Port may set Reset EIEOS Interval Count to 1b when it wants a longer PRBS pattern and subsequently clear it when it needs to obtain Block Alignment.

All TS1 Ordered Sets transmitted in this Phase are requests. The first request maybe a new preset or a new coefficient request or a request to maintain the current link partner transmitter settings by reflecting the settings received in the two consecutive TS1 Ordered Sets with EC=11b that cause the transition to Phase 3.

- The total amount of time spent per preset or coefficients request from transmission of the request to the completion of the evaluation must be less than 2 ms. Implementations that need a longer evaluation time at the final stage of optimization may continue requesting the same preset or coefficient setting beyond the 2 ms limit but must adhere to the 24 ms timeout in this Phase and must not take this exception more than two times. If the requester is unable to receive Ordered Sets within the timeout period, it may assume that the requested setting does not work in that Lane.
- All new preset or coefficient settings must be presented on all configured Lanes simultaneously. Any given Lane is permitted to continue to transmit the current preset or coefficients as its new value if it does not want to change the setting at that time.

- If the data rate of operation is 8.0 GT/s:
  - Next state is Recovery.RcvrLock if all configured Lanes are operating at their optimal settings.
  - The Equalization 8.0 GT/s Phase 3 Successful and Equalization 8.0 GT/s Complete bits of the Link Status 2 register are set to 1b.Else, next state is Recovery.Speed after a timeout of 24 ms with a tolerance of -0 ms and +2 ms
    - successful_speed_negotiation is set to 0b
    - The Equalization 8.0 GT/s Complete bit of the Link Status 2 register is set to 1b.
- If the data rate of operation is 16.0 GT/s:
  - Next state is Recovery.RcvrLock if all configured Lanes are operating at their optimal settings and all Lanes receive two consecutive TS1 Ordered Sets with the Retimer Equalization Extend bit set to 0b.
    - The Equalization 16.0 GT/s Phase 3 Successful and Equalization 16.0 GT/s Complete bits of the 16.0 GT/s Status register are set to 1b
  - Else, next state is Recovery.Speed after a timeout of 24 ms with a tolerance of -0 ms and +2 ms
    - successful_speed_negotiation is set to 0b
    - The Equalization 16.0 GT/s Complete bit of the 16.0 GT/s Status register is set to 1b.

4.2.6.4.2.2 Upstream Lanes

Upon entry to this substate:

- Current phase is Phase 0
  - If the data rate of operation is 8.0 GT/s:
    - The Equalization 8.0 GT/s Phase 1 Successful, Equalization 8.0 GT/s Phase 2 Successful, Equalization 8.0 GT/s Phase 3 Successful, Link Equalization Request

8.0 GT/s, and Equalization 8.0 GT/s Complete bits of the Link Status 2 register are all set to 0b

- The equalization_done_8GT_data_rate variable is set to 1b

- If the data rate of operation is 16.0 GT/s:
  - The Equalization 16.0 GT/s Phase 1 Successful, Equalization 16.0 GT/s Phase 2 Successful, Equalization 16.0 GT/s Phase 3 Successful, Link Equalization Request 16.0 GT/s, and Equalization 16.0 GT/s Complete bits of the 16.0 GT/s Status register are all set to 0b
  - The equalization_done_16GT_data_rate variable is set to 1b

- The start_equalization_w_preset variable is set to 0b.

4.2.6.4.2.2.1 Phase 0 of Transmitter Equalization

- If the current data rate of operation is 8.0 GT/s, transmitter sends TS1 Ordered Sets using the Transmitter settings specified by the Transmitter Preset field received in the EQ TS2 Ordered Sets during the most recent transition to 8.0 GT/s data rate from 2.5 GT/s or 5.0 GT/s data rate. If the current data rate of operation is 16.0 GT/s, transmitter sends TS1 Ordered Sets using the 16.0 GT/s Transmitter settings specified by the Transmitter Preset field received in the 8GT EQ TS2 Ordered Sets during the most recent transition to 16.0 GT/s data rate from 8.0 GT/s data rate. Lanes that received a Reserved or unsupported Transmitter preset value must use an implementation specific method to choose a supported Transmitter Preset for use. Any reference to Transmitter Preset field received in EQ TS2 Ordered Sets or 16.0 GT/s Transmitter Preset field in 8GT EQ TS2 Ordered Sets (depending on the Data Rate) for the remainder of the Recovery.Equalization state is in reference to the presets determined above. In the TS1 Ordered Sets, the EC field is set to 00b, the Transmitter Preset field of each Lane is set to the value it received in the Transmitter Preset field of EQ TS2 Ordered Sets or 16.0 GT/s Transmitter Preset field of 8GT EQ TS2 Ordered Sets, and the Pre-cursor Coefficient, Cursor Coefficient, and Post-cursor Coefficient fields are set to values corresponding to the Transmitter Preset field.
  - Lanes that received a Reserved or unsupported Transmitter Preset in the EQ TS2 Ordered Sets or 8GT EQ TS2 Ordered Sets (depending on the Data Rate): In the TS1 Ordered Sets, the Transmitter Preset field is set to the received Transmitter Preset, the Reject Coefficient Values bit is Set and the Coefficient fields are set to values corresponding to the implementation-specific Transmitter Preset chosen by the Lane.[58]
  - Lanes that did not receive EQ TS2 Ordered Sets or 8GT EQ TS2 Ordered Sets (depending on the Data Rate): In the TS1 Ordered Sets, the Transmitter Preset field is set to the implementation-specific Transmitter Preset chosen by the Lane, the Reject Coefficient Values bit is Clear, and the Coefficient fields are set to values corresponding

[58] An earlier version of this specification permitted the Reject Coefficient Values bit to be clear for this case. This is not recommended, but is permitted.

to the same implementation-specific Transmitter Preset chosen by the Lane and advertised in the Transmitter Preset field.[59]

- The Upstream Port is permitted to wait for up to 500 ns after entering Phase 0 before evaluating receiver information for TS1 Ordered Sets if it needs the time to stabilize its Receiver logic.
- Next phase is Phase 1 if all the configured Lanes receive two consecutive TS1 Ordered Sets with EC=01b.
  - The Receiver must complete its bit lock process and then recognize Ordered Sets within 2 ms after receiving the first bit of the first valid Ordered Set on its Receiver pin.
  - The LF and FS values received in the two consecutive TS1 Ordered Sets must be stored for use during Phase 2 if the Upstream Port wants to adjust the Downstream Port's Transmitter coefficients.
- Else, next state is Recovery.Speed after a 12 ms timeout.
  - successful_speed_negotiation is set to 0b
  - If the data rate is 8.0 GT/s, the Equalization 8.0 GT/s Complete bit of the Link Status 2 register is set to 1b.
  - If the data rate is 16.0 GT/s, the Equalization 16.0 GT/s Complete bit of the 16.0 GT/s Status register is set to 1b.

4.2.6.4.2.2.2 Phase 1 of Transmitter Equalization

- Transmitter sends TS1 Ordered Sets using the Transmitter settings determined in Phase 0. In the TS1 Ordered Sets, the EC field is set to 01b, and the FS, LF, and Post-cursor Coefficient fields of each Lane are set to values corresponding to the Lane's current Transmitter settings.
- Next phase is Phase 2 if all configured Lanes receive two consecutive TS1 Ordered Sets with EC=10b
  - If the data rate is 8.0 GT/s, the Equalization 8.0 GT/s Phase 1 Successful bit of the Link Status 2 register are set to 1b.
  - If the data rate is 16.0 GT/s, the Equalization 16.0 GT/s Phase 1 Successful bit of the 16.0 GT/s Status register is set to 1b.
- Next state is Recovery.RcvrLock if all configured Lanes receive eight consecutive TS1 Ordered Sets with EC=00b
  - o If the data rate is 8.0 GT/s, the Equalization 8.0 GT/s Phase 1 Successful and Equalization 8.0 GT/s Complete bits of the Link Status 2 register are set to 1b
  - If the data rate is 16.0 GT/s, the Equalization 16.0 GT/s Phase 1 Successful and Equalization 16.0 GT/s Complete bits of the 16.0 GT/s Status register are set to 1b.
- Else, next state is Recovery.Speed after a 12 ms timeout
  - successful_speed_negotiation is set to 0b

[59] An earlier version of this specification permitted the Transmitter Preset field to be undefined and the Reject Coefficient Values bit to be clear for this case. This is not recommended, but is permitted

- If the data rate is 8.0 GT/s, the Equalization 8.0 GT/s Complete bit of the Link Status 2 register for the current data rate of operation is set to 1b.
- If the data rate is 16.0 GT/s, the Equalization 16.0 GT/s Complete bit of the 16.0 GT/s Status register is set to 1b.

4.2.6.4.2.2.3 Phase 2 of Transmitter Equalization

- Transmitter sends TS1 Ordered Sets with EC = 10b
- The Port must evaluate and arrive at the optimal settings independently on each Lane. To evaluate a new preset or coefficient setting that is legal, as per the rules in Section 4.2.3 and Chapter 8:
  - Request a new preset by setting the Transmitter Preset field to the desired value and set the Use Preset bit to 1b. Or, request a new set of coefficients by setting the Pre-cursor, Cursor, and Post-cursor Coefficient fields to the desired values and set the Use Preset bit to 0b. Once a request is made, it must be continuously requested for at least 1 μs or until the evaluation of the request is completed, whichever is later.
  - Wait for the required time (500 ns plus the roundtrip delay including the logic delays through the Upstream Port) to ensure that, if accepted, the Downstream Port is transmitting using the requested settings. Obtain Block Alignment and then evaluate the incoming Ordered Sets. Note: The Upstream Port may simply ignore anything it receives during this waiting period as the incoming bit stream may be illegal during the transition to the requested settings. Hence the requirement to validate Block Alignment after this waiting period. If Block Alignment cannot be obtained after an implementation specific amount of time (in addition to the required waiting period specified above) it is recommended to proceed to perform receiver evaluation on the incoming bit stream regardless.
  - If two consecutive TS1 Ordered Sets are received with the Transmitter Preset field (for a preset request) or the Pre-cursor, Cursor, and Post-Cursor fields (for a coefficients request) identical to what was requested and the Reject Coefficient Values bit is Clear, then the requested setting was accepted and and, depending on the results of receiver evaluation, can be considered as a candidate final setting..
  - If two consecutive TS1 Ordered Sets are received with the Transmitter Preset field (for a preset request) or the Pre-Cursor, Cursor, and Post-Cursor fields (for a coefficients request) identical to what was requested and the Reject Coefficient Values bit is Set, then the requested setting was rejected and must not be considered as a candidate final setting.
  - If, after an implementation specific amount of time following the start of receiver evaluation, no consecutive TS1s with the Transmitter Preset field (for a preset request) or the Pre-Cursor, Cursor, and Post-Cursor fields (for a coefficients request) identical to what was requested are received, then the requested setting must not be considered as a candidate final setting.
  - The Upstream Port is responsible for setting the Reset EIEOS Interval Count bit in the TS1 Ordered Sets it transmits according to its evaluation criteria and requirements. The

Use Preset bit of the received TS1 Ordered Sets must not be used to determine whether a request is accepted or rejected.

IMPLEMENTATION NOTE

Reset EIEOS and Coefficient/Preset Requests

A Port may set Reset EIEOS Interval Count to 1b when it wants a longer PRBS pattern and subsequently clear it when it needs to obtain Block Alignment.

All TS1 Ordered Sets transmitted in this Phase are requests. The first request maybe a new preset or a new coefficient request or a request to maintain the current link partner transmitter settings by reflecting the settings received in the two consecutive TS1 Ordered Sets with EC=10b that cause the transition to Phase 2.

- The total amount of time spent per preset or coefficients request from transmission of the request to the completion of the evaluation must be less than 2 ms. Implementations that need a longer evaluation time at the final stage of optimization may continue requesting the same setting beyond the 2 ms limit but must adhere to the 24 ms timeout in this Phase and must not take this exception more than two times. If the requester is unable to receive Ordered Sets within the timeout period, it may assume that the requested setting does not work in that Lane.
- All new preset or coefficient settings must be presented on all configured Lanes simultaneously. Any given Lane is permitted to continue to transmit the current preset or coefficients as its new value if it does not want to change the setting at that time.

- If the data rate of operation is 8.0 GT/s:
  - Next phase is Phase 3 if all configured Lanes are operating at their optimal settings.
    - The Equalization 8.0 GT/s Phase 2 Successful bit of the Link Status 2 register is set to 1b
  - Else, next state is Recovery.Speed after a 24 ms timeout with a tolerance of -0 ms and +2 ms
    - successful_speed_negotiation is set to 0b
    - The Equalization 8.0 GT/s Complete bit of the Link Status 2 register is set to 1b.
- If the data rate of operation is 16.0 GT/s:
  - Next phase is Phase 3 if all configured Lanes are operating at their optimal settings and all Lanes receive two consecutive TS1 Ordered Sets with the Retimer Equalization Extend bit set to 0b.
    - The Equalization 16.0 GT/s Phase 2 Successful bit of the 16.0 GT/s Status register is set to 1b
  - Else, next state is Recovery.Speed after a 24 ms timeout with a tolerance of -0 ms and +2 ms

- successful_speed_negotiation is set to 0b
- The Equalization 16.0 GT/s Complete bit of the 16.0 GT/s Status register is set to 1b.

4.2.6.4.2.2.4 Phase 3 of Transmitter Equalization

- Transmitter sends TS1 Ordered Sets with EC = 11b and the coefficient settings, set on each configured Lane independently, as follows:
  - If two consecutive TS1 Ordered Sets with EC=11b have been received since entering Phase 3, or two consecutive TS1 Ordered Sets with EC=11b and a preset or set of coefficients (as specified by the Use Preset bit) different than the last two consecutive TS1 Ordered Sets with EC=11b:
    - If the preset or coefficients requested in the most recent two consecutive TS Ordered Sets are legal and supported (see Section 4.2.3 and Chapter 8):
      - Change the transmitter settings to the requested preset or coefficients such that the new settings are effective at the Transmitter pins within 500 ns of when the end of the second TS1 Ordered Set requesting the new setting was received at the Receiver pin. The change of Transmitter settings must not cause any illegal voltage level or parameter at the Transmitter pin for more than 1 ns.
      - In the transmitted TS1 Ordered Sets, the Transmitter Preset field is set to the requested preset (for a preset request), the Pre-cursor, Cursor, and Post-cursor Coefficient fields are set to the Transmitter settings (for a preset or a coefficients request), and the Reject Coefficient Values bit is Clear.
    - Else (the requested preset or coefficients are illegal or unsupported): Do not change the Transmitter settings used, but reflect the requested preset or coefficient values in the transmitted TS1 Ordered Sets and set the Reject Coefficient Values bit to 1b.
  - Else: the preset and coefficients currently being used by the Transmitter.
    - The Transmitter preset value initially transmitted on entry to Phase 3 can be the Transmitter preset value transmitted in Phase 0 for the same Data Rate or the Transmitter preset setting currently being used by the Transmitter.
- Next state is Recovery.RcvrLock if all configured Lanes receive two consecutive TS1 Ordered Sets with EC=00b.
  - If the data rate is 8.0 GT/s, the Equalization 8.0 GT/s Phase 3 Successful and Equalization 8.0 GT/s Complete bits of the Link Status 2 register are set to 1b.
  - If the data rate is 16.0 GT/s, the Equalization 16.0 GT/s Phase 3 Successful and Equalization 16.0 GT/s Complete bits of the 16.0 GT/s Status register are set to 1b.
- Else, next state is Recovery.Speed after a timeout of 32 ms with a tolerance of -0 ms and +4 ms
  - successful_speed_negotiation is set to 0b

- If the data rate is 8.0 GT/s, the Equalization 8.0 GT/s Complete bit of the Link Status 2 register is set to 1b.
- If the data rate is 16.0 GT/s, the Equalization 16.0 GT/s Complete bit of the 16.0 GT/s Status register is set to 1b.

4.2.6.4.3 Recovery.Speed

- The Transmitter enters Electrical Idle and stays there until the Receiver Lanes have entered Electrical Idle, and then additionally remains there for at least 800 ns on a successful speed negotiation (i.e., successful_speed_negotiation = 1b) or at least 6 µs on an unsuccessful speed negotiation (i.e., successful_speed_negotiation = 0b), but stays there no longer than an additional 1 ms. The frequency of operation is permitted to be changed to the new data rate only after the Receiver Lanes have entered Electrical Idle. If the negotiated data rate is 5.0 GT/s, and if operating in full swing mode, -6 dB de-emphasis level must be selected for operation if the select_deemphasis variable is 0b and -3.5 dB de-emphasis level must be selected for operation if the select_deemphasis variable is 1b. Note that if the link is already operating at the highest data rate supported by both Ports, Recovery.Speed is executed but the data rate is not changed.

  An EIOSQ must be sent prior to entering Electrical Idle.

  The DC common mode voltage is not required to be within specification.

  An Electrical Idle condition exists on the Lanes if an EIOS is received on any of the configured Lanes or Electrical Idle is detected/inferred as described in Section 4.2.4.3.

  - On entry to this substate following a successful speed negotiation (i.e., successful_speed_negotiation = 1b), an Electrical Idle condition may be inferred on the Receiver Lanes if a TS1 or TS2 Ordered Set has not been received in any configured Lane in a time interval specified in Table 4-12. (This covers the case where the Link is operational and both sides have successfully received TS Ordered Sets. Hence, a lack of a TS1 or TS2 Ordered Set in the specified interval can be interpreted as entry to Electrical Idle.)
  - Else on entry to this substate following an unsuccessful speed negotiation (i.e., successful_speed_negotiation = 0b) if an exit from Electrical Idle has not been detected at least once in any configured Lane in a time interval specified in Table 4-12. (This covers the case where at least one side is having trouble receiving TS Ordered Sets that was transmitted by the other agent, and hence a lack of exit from Electrical Idle in a longer interval can be treated as equivalent to entry to Electrical Idle.)

- Next state is Recovery.RcvrLock after the Transmitter Lanes are no longer required to be in Electrical Idle as described in the condition above.

  - If this substate has been entered from Recovery.RcvrCfg following a successful speed change negotiation (i.e., successful_speed_negotiation = 1b), the new data rate is changed on all the configured Lanes to the highest common data rate advertised by both sides of the Link. The changed_speed_recovery variable is set to 1b.
  - Else if this substate is being entered for a second time since entering Recovery from L0 or L1 (i.e., changed_speed_recovery = 1b), the new data rate will be the data rate at

which the LTSSM entered Recovery from L0 or L1. The changed_speed_recovery variable will be reset to 0b.

- Else the new data rate will be 2.5 GT/s. The changed_speed_recovery variable remains reset at 0b.

  Note: This represents the case where the frequency of operation in L0 was greater than 2.5 GT/s and one side could not operate at that frequency and timed out in Recovery.RcvrLock the first time it entered that substate from L0 or L1.

- Next state is Detect after a 48 ms timeout.
  - Note: This transition is not possible under normal conditions.
- The directed_speed_change variable will be reset to 0b. The new data rate must be reflected in the Current Link Speed field of the Link Status register.
  - On a Link bandwidth change, if successful_speed_negotiation is set to 1b and the Autonomous Change bit (bit 6 of Symbol 4) in the eight consecutive TS2 Ordered Sets received while in Recovery.RcvrCfg is set to 1b or the speed change was initiated by the Downstream Port for autonomous reasons (non-reliability and not due to the setting of the Link Retrain bit), the Link Autonomous Bandwidth Status bit of the Link Status register is set to 1b.
  - Else: on a Link bandwidth change, the Link Bandwidth Management Status bit of the Link Status register is set to 1b.

4.2.6.4.4 Recovery.RcvrCfg

Transmitter sends TS2 Ordered Sets on all configured Lanes using the same Link and Lane numbers that were set after leaving Configuration. The speed_change bit (bit 7 of data rate identifier Symbol in TS2 Ordered Set) must be set to 1b if the directed_speed_change variable is already set to 1b. The N_FTS value in the transmitted TS2 Ordered Sets should reflect the number at the current data rate.

The Downstream Port must transmit EQ TS2 Ordered Sets (TS2 Ordered Sets with Symbol 6 bit 7 set to 1b) on each configured Lane with the Transmitter Preset and Receiver Preset Hint fields set to the values specified by the Upstream 8.0 GT/s Port Transmitter Preset and the Upstream 8.0 GT/s Port Receiver Preset Hint fields from the corresponding Lane Equalization Control Register entry if all of the following conditions are satisfied:

a) The Downstream Port advertised 8.0 GT/s data rate support in Recovery.RcvrLock, and 8.0 GT/s data rate support has been advertised in the Configuration.Complete or Recovery.RcvrCfg substates by the Upstream Port since exiting the Detect state, and eight consecutive TS1 or TS2 Ordered Sets were received on any configured Lane prior to entry to this substate with speed_change bit set to 1b

b) The equalization_done_8GT_data_rate variable is 0b or if directed

c) The current data rate of operation is 2.5 GT/s or 5.0 GT/s

The Downstream Port must transmit 8GT EQ TS2 Ordered Sets (TS2 Ordered Sets with Symbol 7 bit 7 set to 1b) on each configured Lane with the Transmitter Preset field set to the values specified by the Upstream Port 16.0 GT/s Transmitter Preset field from the corresponding 16.0 GT/s Lane Equalization Control Register entry if all of the following conditions are satisfied:

a) The Downstream Port advertised 16.0 GT/s data rate support in Recovery.RcvrLock, and 16.0 GT/s data rate support has been advertised in the Configuration.Complete or Recovery.RcvrCfg substates by the Upstream Port since exiting the Detect state, and eight consecutive TS1 or TS2 Ordered Sets were received on any configured Lane prior to entry to this substate with speed_change bit set to 1b

b) The equalization_done_16GT_data_rate variable is 0b or if directed

c) The current data rate of operation is 8.0 GT/s

The Upstream Port may optionally transmit 8GT EQ TS2 Ordered Sets with the 16.0 GT/s Transmitter Preset fields set to implementation specific values if all of the following conditions are satisfied:

a) The Upstream Port advertised 16.0 GT/s data rate support in Recovery.RcvrLock, and 16.0 GT/s data rate support has been advertised in the Configuration.Complete or Recovery.RcvrCfg substates by the Downstream Port since exiting the Detect state, and eight consecutive TS1 or TS2 Ordered Sets were received on any configured Lane prior to entry to this substate with speed_change bit set to 1b

b) The equalization_done_16GT_data_rate variable is 0b or if directed

c) The current data rate of operation is 8.0 GT/s

When using 128b/130b encoding, Upstream and Downstream Ports use the Request Equalization, Equalization Request Data Rate, and Quiesce Guarantee bits of their transmitted TS2 Ordered Sets to communicate equalization requests as described in Section 4.2.3. When not requesting equalization, the Request Equalization, Equalization Request Data Rate, and Quiesce Guarantee bits must be set to 0b.

The start_equalization_w_preset variable is reset to 0b upon entry to this substate.

- On entry to this substate, a Downstream Port must set the select_deemphasis variable equal to the Selectable De-emphasis field in the Link Control 2 register or adopt some implementation specific mechanism to set the select_deemphasis variable, including using the value requested by the Upstream Port in the eight consecutive TS1 Ordered Sets it received. A Downstream Port advertising 5.0 GT/s data rate support must set the Selectable De-emphasis bit (Symbol 4 bit 6) of the TS2 Ordered Sets it transmits identical to the select_deemphasis variable. An Upstream Port must set its Autonomous Change bit (Symbol 4 bit 6) to 1b in the TS2 Ordered Set if it intends to change the Link bandwidth for autonomous reasons.
  - For devices that support Link width upconfigure, it is recommended that the Electrical Idle detection circuitry be activated in the set of currently inactive Lanes in this substate, the Recovery.Idle substate, and Configuration.Linkwidth.Start substates, if the directed_speed_change variable is reset to 0b. This is done so that during a Link upconfigure, the side that does not initiate the upconfiguration does not miss the first EIEOS sent by the initiator during the Configuration.Linkwidth.Start substate.
- Next state is Recovery.Speed if all of the following conditions are true:
  - One of the following conditions is satisfied:
    i. Eight consecutive TS2 Ordered Sets are received on any configured Lane with identical data rate identifiers, identical values in Symbol 6, and the

speed_change bit set to 1b and eight consecutive TS2 Ordered Sets are standard TS2 Ordered Sets if either 8b/10b or 128b/130b encoding is used

ii. Eight consecutive EQ TS2 or 8GT EQ TS2 Ordered Sets are received on all configured Lanes with identical data rate identifiers, identical value in Symbol 6, and the speed_change bit set to 1b

iii. Eight consecutive EQ TS2 or 8GT EQ TS2 Ordered Sets are received on any configured Lane with identical data rate identifiers, identical value in Symbol 6, and the speed_change bit set to 1b and 1 ms has expired since the receipt of the eight consecutive EQ Ordered Sets on any configured Lane

- Either the current data rate is greater than 2.5 GT/s or greater than 2.5 GT/s data rate identifiers are set both in the transmitted and the (eight consecutive) received TS2 Ordered Sets
- For 8b/10b encoding, at least 32 TS2 Ordered Sets, without being interrupted by any intervening EIEOS, are transmitted with the speed_change bit set to 1b after receiving one TS2 Ordered Set with the speed_change bit set to 1b in the same configured Lane. For 128b/130b encoding, at least 128 TS2 Ordered Sets are transmitted with the speed_change bit set to 1b after receiving one TS2 Ordered Set with the speed_change bit set to 1b in the same configured Lane.

The data rate(s) advertised on the received eight consecutive TS2 Ordered Sets with the speed_change bit set is noted as the data rate(s) that can be supported by the other Port. The Autonomous Change bit (Symbol 4 bit 6) in these received eight consecutive TS2 Ordered Sets is noted by the Downstream Port for possible logging in the Link Status register in Recovery.Speed substate. Upstream Ports must register the Selectable De-emphasis bit (bit 6 of Symbol 4) advertised in these eight consecutive TS2 Ordered Sets in the select_deemphasis variable. The new speed to change to in Recovery.Speed is the highest data rate that can be supported by both Ports on the Link. For an Upstream Port, if the current data rate of operation is 2.5 GT/s or 5.0 GT/s and these eight consecutive TS2 Ordered Sets are EQ TS2 Ordered Sets, it must set the start_equalization_w_preset variable to 1b and update the Upstream Port 8.0 GT/s Transmitter Preset and Upstream Port 8.0 GT/s Receiver Preset Hint fields of the Lane Equalization Control Register entry with the values received in the eight consecutive EQ TS2 Ordered Sets for the corresponding Lane. For an Upstream Port, if the current data rate of operation is 8.0 GT/s, 16.0 GT/s support is advertised by both ends, and these eight consecutive TS2 Ordered Sets are 8GT EQ TS2 Ordered Sets, it must set the start_equalization_w_preset variable to 1b and update the 16.0 GT/s Upstream Port Transmitter Preset field of the 16.0 GT/s Lane Equalization Control Register entry with the values received in the eight consecutive 8GT EQ TS2 Ordered Sets for the corresponding Lane. Any configured Lanes which do not receive EQ TS2 or 8GT EQ TS2 Ordered Sets meeting this criteria will use implementation dependent preset values when first operating at 8.0 GT/s or 16.0 GT/s prior to performing link equalization. A Downstream Port must set the start_equalization_w_preset variable to 1b if the equalization_done_8GT_data_rate variable is 0b or if 16.0 GT/s support is advertised by both ends and the equalization_done_16GT_data_rate variable is 0b or if directed. A Downstream Port must record the 16.0 GT/s Transmitter Preset settings advertised in the eight consecutive TS2 Ordered Sets received if they are 8GT EQ TS2 Ordered Sets, and 16.0 GT/s support is advertised by both ends. The variable successful_speed_negotiation is set to 1b. Note that if

the Link is already operating at the highest data rate supported by both Ports, Recovery.Speed is executed but the data rate is not changed. If 128b/130b encoding is used and the Request Equalization bit is Set in the eight consecutive TS2 Ordered Sets, the Port must handle it as an equalization request as described in Section 4.2.3.

- Next state is Recovery.Idle if the following two conditions are both true:
  - Eight consecutive TS2 Ordered Sets are received on all configured Lanes with the same Link and Lane number that match what is being transmitted on those same Lanes with identical data rate identifiers within each Lane and one of the following two sub-conditions are true:
    - the speed_change bit is 0b in the received eight consecutive TS2 Ordered Sets
    - current data rate is 2.5 GT/s and either no 5.0 GT/s, or higher, data rate identifiers are set in the received eight consecutive TS2 Ordered Sets, or no 5.0 GT/s, or higher, data rate identifiers are being transmitted in the TS2 Ordered Sets
  - 16 TS2 Ordered Sets are sent after receiving one TS2 Ordered Set without being interrupted by any intervening EIEOS. The changed_speed_recovery variable and the directed_speed_change variable are reset to 0b on entry to Recovery.Idle.
  - If the N_FTS value was changed, the new value must be used for future L0s states.
  - When using 8b/10b encoding, Lane-to-Lane de-skew must be completed before leaving Recovery.RcvrCfg.
  - The device must note the data rate identifier advertised on any configured Lane in the eight consecutive TS2 Ordered Sets described in this state transition. This will override any previously recorded value.
  - When using 128b/130b encoding and if the Request Equalization bit is Set in the eight consecutive TS2 Ordered Sets, the device must note it and follow the rules in Section 4.2.3.
- Next state is Configuration if eight consecutive TS1 Ordered Sets are received on any configured Lanes with Link or Lane numbers that do not match what is being transmitted on those same Lanes and 16 TS2 Ordered Sets are sent after receiving one TS1 Ordered Set and one of the following two conditions apply:
  - the speed_change bit is 0b on the received TS1 Ordered Sets
  - current data rate is 2.5 GT/s and either no 5.0 GT/s, or higher, data rate identifiers are set in the received eight consecutive TS1 Ordered Sets, or no 5.0 GT/s, or higher, data rate identifiers are being transmitted in the TS2 Ordered Sets

The changed_speed_recovery variable and the directed_speed_change variable are reset to 0b if the LTSSM transitions to Configuration.

  - If the N_FTS value was changed, the new value must be used for future L0s states.
- Next state is Recovery.Speed if the speed of operation has changed to a mutually negotiated data rate since entering Recovery from L0 or L1 (i.e., changed_speed_recovery = 1b) and an EIOS has been detected or an Electrical Idle condition has been inferred/detected on any of the configured Lanes and no configured Lane received a TS2 Ordered Set since entering this

substate (Recovery.RcvrCfg). The new data rate to operate after leaving Recovery.Speed will be reverted back to the speed of operation during entry to Recovery from L0 or L1.

As described in Section 4.2.4.3, an Electrical Idle condition may be inferred if a TS1 or TS2 Ordered Set has not been received in a time interval specified in Table 4-12.

- Next state is Recovery.Speed if the speed of operation has not changed to a mutually negotiated data rate since entering Recovery from L0 or L1 (i.e., changed_speed_recovery = 0b) and the current speed of operation is greater than 2.5 GT/s and an EIOS has been detected or an Electrical Idle condition has been detected/inferred on any of the configured Lanes and no configured Lane received a TS2 Ordered Set since entering this substate (Recovery.RcvrCfg). The new data rate to operate after leaving Recovery.Speed will be 2.5 GT/s.

  As described in Section 4.2.4.3, an Electrical Idle condition may be inferred if a TS1 or TS2 Ordered Set has not been received in a time interval specified in Table 4-12.

  Note: This transition implies that the other side was unable to achieve Symbol lock or Block alignment at the speed with which it was operating. Hence both sides will go back to the 2.5 GT/s speed of operation and neither device will attempt to change the speed again without exiting Recovery state. It should also be noted that even though a speed change is involved here, the changed_speed_recovery will be 0b.

- After a 48 ms timeout;
  - The next state is Detect if the current data rate is 2.5 GT/s or 5.0 GT/s.
  - The next state is Recovery.Idle if the idle_to_rlock_transitioned variable is less than FFh and the current data rate is 8.0 GT/s or higher.
    - i. The changed_speed_recovery variable and the directed_speed_change variable are reset to 0b on entry to Recovery.Idle.
  - Else the next state is Detect.

4.2.6.4.5 Recovery.Idle

- Next state is Disabled if directed.
  - Note: "if directed" applies to a Downstream or optional crosslink Port that is instructed by a higher Layer to assert the Disable Link bit (TS1 and TS2) on the Link.
- Next state is Hot Reset if directed.
  - Note: "if directed" applies to a Downstream or optional crosslink Port that is instructed by a higher Layer to assert the Hot Reset bit (TS1 and TS2) on the Link.
- Next state is Configuration if directed.
  - Note: "if directed" applies to a Port that is instructed by a higher Layer to optionally re-configure the Link (i.e., different width Link).
- Next state is Loopback if directed to this state, and the Transmitter is capable of being a loopback master, which is determined by implementation specific means.
  - Note: "if directed" applies to a Port that is instructed by a higher Layer to assert the Loopback bit (TS1 and TS2) on the Link.

- Next state is Disabled immediately after any configured Lane has the Disable Link bit asserted in two consecutively received TS1 Ordered Sets.
  - Note: This is behavior only applicable to Upstream and optional crosslink Ports.
- Next state is Hot Reset immediately after any configured Lane has the Hot Reset bit asserted in two consecutive TS1 Ordered Sets.
  - Note: This is behavior only applicable to Upstream and optional crosslink Ports.
- Next state is Configuration if two consecutive TS1 Ordered Sets are received on any configured Lane with a Lane number set to PAD.
  - Note: A Port that optionally transitions to Configuration to change the Link configuration is guaranteed to send Lane numbers set to PAD on all Lanes.
  - Note: It is recommended that the LTSSM initiate a Link width up/downsizing using this transition to reduce the time it takes to change the Link width.
- Next state is Loopback if any configured Lane has the Loopback bit asserted in two consecutive TS1 Ordered Sets.
  - Note: The device receiving the Ordered Set with the Loopback bit set becomes the loopback slave.
- When using 8b/10b encoding, the Transmitter sends Idle data on all configured Lanes.
- When using 128b/130b encoding:
  - If the data rate is 8.0 GT/s, the Transmitter sends one SDS Ordered Set on all configured Lanes to start a Data Stream and then sends Idle data Symbols on all configured Lanes. The first Idle data Symbol transmitted on Lane 0 is the first Symbol of the Data Stream.
  - If the data rate is 16.0 GT/s, the Transmitter sends one Control SKP Ordered Set followed immediately by one SDS Ordered Set on all configured Lanes to start a Data Stream and then sends Idle data Symbols on all configured Lanes. The first Idle data Symbol transmitted on Lane 0 is the first Symbol of the Data Stream.
  - If directed to other states, Idle Symbols do not have to be sent, and must not be sent with 128b/130b encoding, before transitioning to the other states (i.e., Disabled, Hot Reset, Configuration, or Loopback)

IMPLEMENTATION NOTE

EDS Usage

In 128b/130b encoding, on transition to Configuration or Loopback or Hot Reset or Disabled, an EDS must be sent if a Data Stream is active (i.e., an SDS Ordered Set has been sent). It is possible that the side that is not initiating Link Upconfigure has already transmitted SDS and transmitting Data Stream (Logical IDL) when it receives the TS1 Ordered Sets. In that situation, it will send EDS in the set of Lanes that are active before sending the TS1 Ordered Sets in Configuration.

- When using 8b/10b encoding, next state is L0 if eight consecutive Symbol Times of Idle data are received on all configured Lanes and 16 Idle data Symbols are sent after receiving one Idle data Symbol.
  - If software has written a 1b to the Retrain Link bit in the Link Control register since the last transition to L0 from Recovery or Configuration, the Downstream Port must set the Link Bandwidth Management Status bit of the Link Status register to 1b.
- When using 128b/130b encoding, next state is L0 if eight consecutive Symbol Times of Idle data are received on all configured Lanes, 16 Idle data Symbols are sent after receiving one Idle data Symbol, and this state was not entered by a timeout from Recovery.RcvrCfg
  - The Idle data Symbols must be received in Data Blocks.
  - Lane-to-Lane de-skew must be completed before Data Stream processing starts.
  - If software has written a 1b to the Retrain Link bit in the Link Control register since the last transition to L0 from Recovery or Configuration, the Downstream Port must set the Link Bandwidth Management Status bit of the Link Status register to 1b.
  - The idle_to_rlock_transitioned variable is reset to 00h on transition to L0.
- Otherwise, after a 2 ms timeout:
  - If the idle_to_rlock_transitioned variable is less than FFh, the next state is Recovery.RcvrLock.
    - If the data rate is 8.0 GT/s or higher, the idle_to_rlock_transitioned variable is incremented by 1b upon transitioning to Recovery.RcvrLock.
    - If the data rate is 5.0 GT/s (or, if supported in 2.5 GT/s), the idle_to_rlock_transitioned variable is set to FFh, upon transitioning to Recovery.RcvrLock.

- Else the next state is Detect

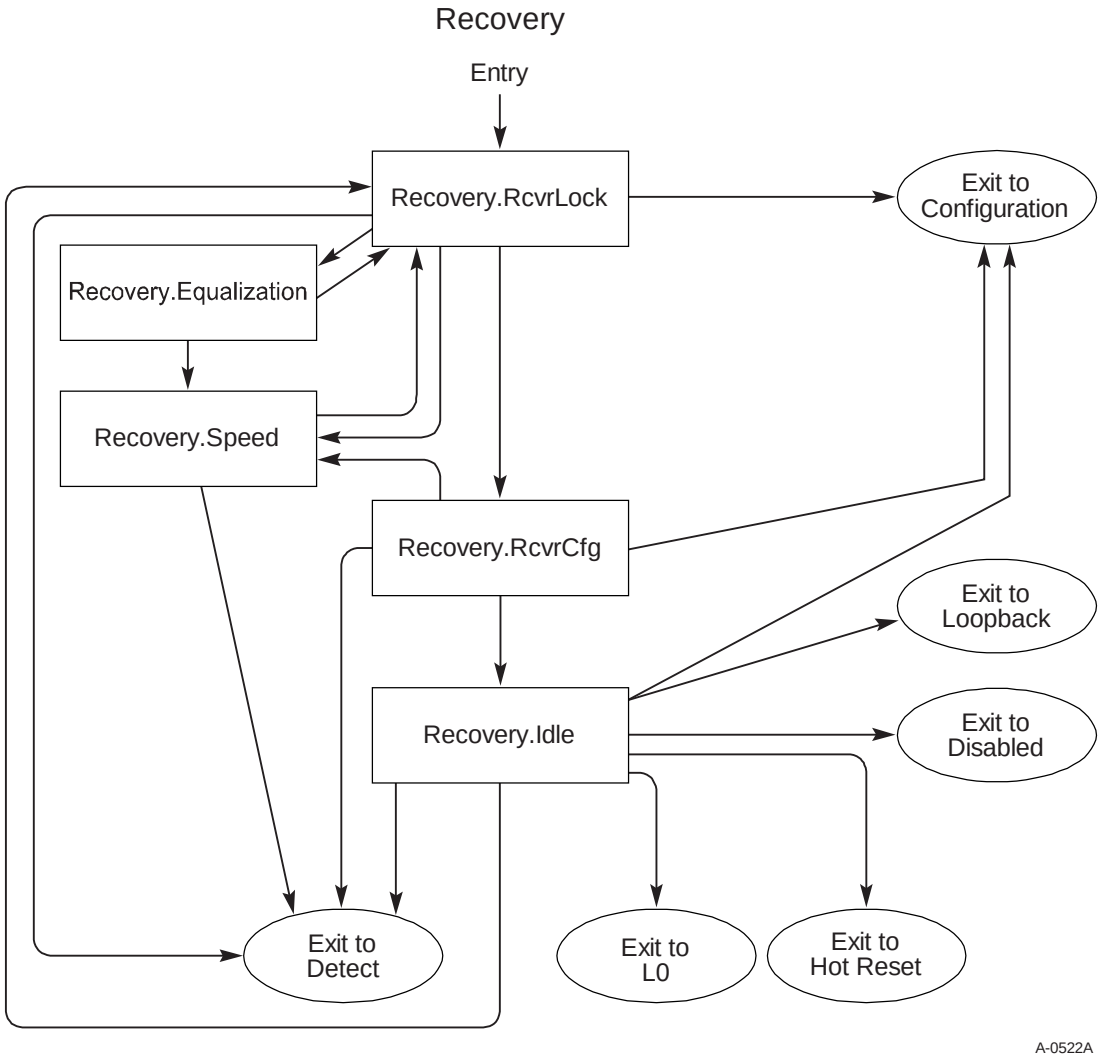


Figure 4-27: Recovery Substate Machine

4.2.6.5 *L0*

This is the normal operational state.

- LinkUp = 1b (status is set true).
  - On receipt of an STP or SDP Symbol, the idle_to_rlock_transitioned variable is reset to 00h.
- Next state is Recovery if either of the two conditions are satisfied (i) if directed to change speed (directed_speed_change variable = 1b) by a higher layer when both sides support greater than 2.5 GT/s speeds and the Link is in DL_Active state, or (ii) if directed to change speed (directed_speed_change variable = 1b) by a higher layer when both sides support 8.0 GT/s data rate to perform Transmitter Equalization at 8.0 GT/s data rate. The changed_speed_recovery bit is reset to 0b.
  - For an Upstream Port, the directed_speed_change variable must not be set to 1b if it has never recorded greater than 2.5 GT/s data rate support advertised in Configuration.Complete or Recovery.RcvrCfg substates by the Downstream Port since exiting the Detect state.
  - For a Downstream Port, the directed_speed_change variable must not be set to 1b if it has never recorded greater than 2.5 GT/s data rate support advertised in Configuration.Complete or Recovery.RcvrCfg substates by the Upstream Port since exiting the Detect state. If greater than 2.5 GT/s data rate support has been noted, the Downstream Port must set the directed_speed_change variable to 1b if the Retrain Link bit of the Link Control register is set to 1b and the Target Link Speed field in the Link Control 2 register is not equal to the current Link speed.
  - A Port supporting greater than 2.5 GT/s data rates must participate in the speed change even if the Link is not in DL_Active state if it is requested by the other side through the TS Ordered Sets.
- Next state is Recovery if directed to change Link width.
  - The upper layer must not direct a Port to increase the Link width if the other Port did not advertise the capability to upconfigure the Link width during the Configuration state or if the Link is currently operating at the maximum possible width it negotiated on initial entry to the L0 state.
  - Normally, the upper layer will not reduce width if upconfigure_capable is reset to 0b other than for reliability reasons, since the Link will not be able to go back to the original width if upconfigure_capable is 0b. A Port must not initiate reducing the Link width for reasons other than reliability if the Hardware Autonomous Width Disable bit in the Link Control register is set to 1b.
  - The decision to initiate an increase or decrease in the Link width, as allowed by the specification, is implementation specific.
- Next state is Recovery if a TS1 or TS2 Ordered Set is received on any configured Lane or an EIEOS is received on any configured Lane in 128b/130b encoding.
- Next state is Recovery if directed to this state. If Electrical Idle is detected/inferred on all Lanes without receiving an EIOS on any Lane, the Port may transition to the Recovery state or may

remain in L0. In the event that the Port is in L0 and the Electrical Idle condition occurs without receiving an EIOS, errors may occur and the Port may be directed to transition to Recovery.

- As described in Section 4.2.4.3, an Electrical Idle condition may be inferred on all Lanes under any one of the following conditions: (i) absence of a Flow Control Update DLLP in any 128 µs window, (ii) absence of a SKP Ordered Set in any of the configured Lanes in any 128 µs window, or (iii) absence of either a Flow Control Update DLLP or a SKP Ordered Set in any of the configured Lanes in any 128 µs window.
- Note: "if directed" applies to a Port that is instructed by a higher Layer to transition to Recovery including the Retrain Link bit in the Link Control register being set.
- The Transmitter may complete any TLP or DLLP in progress.

❑ Next state is L0s for only the Transmitter if directed to this state and the Transmitter implements L0s. See Section 4.2.6.6.2.

- Note: "if directed" applies to a Port that is instructed by a higher Layer to initiate L0s (see Section 5.4.1.1.1).
- Note: This is a point where the TX and RX may diverge into different LTSSM states.

❑ Next state is L0s for only the Receiver if an EIOS is received on any Lane, the Receiver implements L0s, and the Port is not directed to L1 or L2 states by any higher layers. See Section 4.2.6.6.1.

- Note: This is a point where the TX and RX may diverge into different LTSSM states.

❑ Next state is Recovery if an EIOS is received on any Lane, the Receiver does not implement L0s, and the Port is not directed to L1 or L2 states by any higher layers. See Section 4.2.6.6.1.

❑ Next state is L1:

(i) If directed

and

(ii) an EIOS is received on any Lane

and

(iii) an EIOSQ is transmitted on all Lanes.

- Note: "if directed" is defined as both ends of the Link having agreed to enter L1 immediately after the condition of both the receipt and transmission of the EIOS(s) is met. A transition to L1 can be initiated by PCI-PM (see Section 5.3.2.1) or by ASPM (see Section 5.4.1.2.1).
- Note: When directed by a higher Layer one side of the Link always initiates and exits to L1 by transmitting the EIOS(s) on all Lanes, followed by a transition to Electrical Idle.[60] The same Port then waits for the receipt of an EIOS on any Lane, and then immediately transitions to L1. Conversely, the side of the Link that first receives the EIOS(s) on any Lane must send an EIOS on all Lanes and immediately transition to L1.

---

[60] The common mode being driven must meet the Absolute Delta Between DC Common Mode During L0 and Electrical Idle ($V_{TX\text{-}CM\text{-}DC\text{-}ACTIVE\text{-}IDLE\text{-}DELTA}$) specification (see Table 8-7).

- Next state is L2:
  - (i) If directed

  and

  - (ii) an EIOS is received on any Lane
  - (iii) an EIOSQ is transmitted on all Lanes.
  - Note: "if directed" is defined as both ends of the Link having agreed to enter L2 immediately after the condition of both the receipt and transmission of the EIOS(s) is met (see Section 5.3.2.3 for more details).
  - Note: When directed by a higher Layer, one side of the Link always initiates and exits to L2 by transmitting EIOS on all Lanes followed by a transition to Electrical Idle.61 The same Port then waits for the receipt of EIOS on any Lane, and then immediately transitions to L2. Conversely, the side of the Link that first receives an EIOS on any Lane must send an EIOS on all Lanes and immediately transition to L2.

4.2.6.6 *L0s*

The L0s substate machine is shown in Figure 4-28.

4.2.6.6.1 Receiver L0s

A Receiver must implement L0s if its Port advertises support for L0s, as indicated by the ASPM Support field in the Link Capabilities register. It is permitted for a Receiver to implement L0s even if its Port does not advertise support for L0s.

4.2.6.6.1.1 Rx_L0s.Entry

- Next state is Rx_L0s.Idle after a $T_{TX\text{-}IDLE\text{-}MIN}$ (Table 8-3) timeout.
  - Note: This is the minimum time the Transmitter must be in an Electrical Idle condition.

4.2.6.6.1.2 Rx_L0s.Idle

- Next state is Rx_L0s.FTS if the Receiver detects an exit from Electrical Idle on any Lane of the configured Link.
- Next state is Rx_L0s.FTS after a 100 ms timeout if the current data rate is 8.0 GT/s or higher and the Port's Receivers do not meet the $Z_{RX\text{-}DC}$ specification for 2.5 GT/s (see Table 8-11). All Ports are permitted to implement the timeout and transition to Rx_L0s.FTS when the data rate is 8.0 GT/s or higher.

---

61 The common mode being driven does not need to meet the Absolute Delta Between DC Common Mode During L0 and Electrical Idle ($V_{TX\text{-}CM\text{-}DC\text{-}ACTIVE\text{-}IDLE\text{-}DELTA}$) specification (see Table 8-7).

4.2.6.6.1.3 Rx_L0s.FTS

- The next state is L0 if a SKP Ordered Set is received in 8b/10b encoding or the SDS Ordered Set is received for 128b/130b encoding on all configured Lanes of the Link.
  - The Receiver must be able to accept valid data immediately after the SKP Ordered Set for 8b/10b encoding.
  - The Receiver must be able to accept valid data immediately after the SDS Ordered Set for 128b/130b encoding.
  - Lane-to-Lane de-skew must be completed before leaving Rx_L0s.FTS.
- Otherwise, next state is Recovery after the N_FTS timeout.
  - When using 8b/10b encoding: The N_FTS timeout shall be no shorter than 40*[N_FTS+3] * UI (The 3 * 40 UI is derived from six Symbols to cover a maximum SKP Ordered Set + four Symbols for a possible extra FTS + 2 Symbols of design margin), and no longer than twice this amount. When the Extended Synch bit is Set the Receiver N_FTS timeout must be adjusted to no shorter than 40* [2048] * UI (2048 FTSs) and no longer than 40* [4096] * UI (4096 FTSs). Implementations must take into account the worst case Lane to Lane skew, their design margins, as well as the four to eight consecutive EIE Symbols in speeds other than 2.5 GT/s when choosing the appropriate timeout value within the specification's defined range.
  - When using 128b/130b encoding: The N_FTS timeout shall be no shorter than 130*[N_FTS+5+ 12 + Floor(N_FTS/32)] * UI and no longer than twice this amount. The 5 + Floor(N_FTS/32) accounts for the first EIEOS, the last EIEOS, the SDS, the periodic EIEOS and an additional EIEOS in case an implementation chooses to send two EIEOS followed by an SDS when N_FTS is divisible by 32. The 12 is there to account for the number of SKP Ordered Sets that will be transmitted if the Extended Synch bit is Set. When the Extended Synch bit is Set, the timeout should be the same as the normal case with N_FTS equal to 4096.
  - The Transmitter must also transition to Recovery, but is permitted to complete any TLP or DLLP in progress.
  - It is recommended that the N_FTS field be increased when transitioning to Recovery to prevent future transitions to Recovery from Rx_L0s.FTS.

4.2.6.6.2 Transmitter L0s

A Transmitter must implement L0s if its Port advertises support for L0s, as indicated by the ASPM Support field in the Link Capabilities register. It is permitted for a Transmitter to implement L0s even if its Port does not advertise support for L0s.

4.2.6.6.2.1 Tx_L0s.Entry

- Transmitter sends an EIOSQ and enters Electrical Idle.

- The DC common mode voltage must be within specification by $T_{TX\text{-}IDLE\text{-}SET\text{-}TO\text{-}IDLE}$.[62]

- Next state is Tx_L0s.Idle after a $T_{TX\text{-}IDLE\text{-}MIN}$ (Table 8-3) timeout.

4.2.6.6.2.2 Tx_L0s.Idle

- Next state is Tx_L0s.FTS if directed.

IMPLEMENTATION NOTE

Increase of N_FTS Due to Timeout in Rx_L0s.FTS

The Transmitter sends the N_FTS fast training sequences by going through Tx_L0s.FTS substates to enable the Receiver to reacquire its bit and Symbol lock or Block alignment. In the absence of the N_FTS fast training sequence, the Receiver will timeout in Rx_L0s.FTS substate and may increase the N_FTS number it advertises in the Recovery state.

4.2.6.6.2.3 Tx_L0s.FTS

- Transmitter must send N_FTS Fast Training Sequences on all configured Lanes.
  - Four to eight EIE Symbols must be sent prior to transmitting the N_FTS (or 4096 if the Extended Synch bit is Set) number of FTS in 5.0 GT/s data rates. An EIEOS must be sent prior to transmitting the N_FTS (or 4096 if the Extended Synch bit is Set) number of FTS with 128b/130b encoding. In 2.5 GT/s speed, up to one full FTS may be sent before the N_FTS (or 4096 if the Extended Synch bit is Set) number of FTSs are sent.
  - SKP Ordered Sets must not be inserted before all FTSs as defined by the agreed upon N_FTS parameter are transmitted.
  - If the Extended Synch bit is Set, the Transmitter must send 4096 Fast Training Sequences, inserting SKP Ordered Sets according to the requirements in Section 4.2.4.5.
- When using 8b/10b encoding, the Transmitter must send a single SKP Ordered Set on all configured Lanes.
- When using 128b/130b encoding, the Transmitter must send one EIEOS followed by one SDS Ordered Set on all configured Lanes. Note: The first Symbol transmitted on Lane 0 after the SDS Ordered Set is the first Symbol of the Data Stream.
- Next state must be L0, after completing the above required transmissions.

[62] The common mode being driven must meet the Absolute Delta Between DC Common Mode During L0 and Electrical Idle ($V_{TX\text{-}CM\text{-}DC\text{-}ACTIVE\text{-}IDLE\text{-}DELTA}$) specification (see Table 8-7).

IMPLEMENTATION NOTE

No SKP Ordered Set requirement when exiting L0s at 16.0 GT/s

Unlike in other LTSSM states, when exiting Tx_L0s.FTS no Control SKP Ordered Set is transmitted before transmitting the SDS. This results in the Data Parity information associated with the last portion of the previous datastream being discarded. Not sending the Control SKP Ordered Set reduces complexity and improves exit latency.

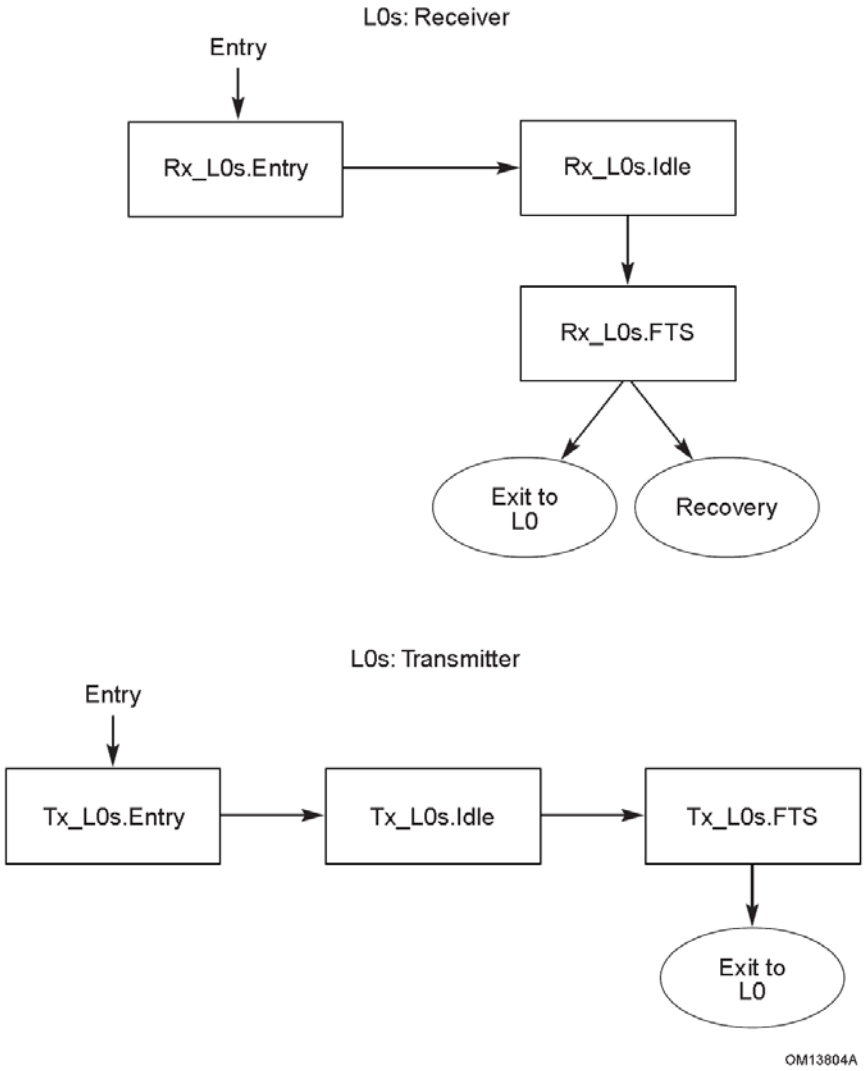


Figure 4-28: L0s Substate Machine

4.2.6.7 *L1*

The L1 substate machine is shown in Figure 4-29.

4.2.6.7.1 L1.Entry

- All configured Transmitters are in Electrical Idle.
  - The DC common mode voltage must be within specification by $T_{TX\text{-}IDLE\text{-}SET\text{-}TO\text{-}IDLE}$.
- The next state is L1.Idle after a $T_{TX\text{-}IDLE\text{-}MIN}$ (Table 8-3) timeout.
  - Note: This guarantees that the Transmitter has established the Electrical Idle condition.

4.2.6.7.2 L1.Idle

- Transmitter remains in Electrical Idle.
- The DC common mode voltage must be within specification, except as allowed by L1 PM Substates, when applicable.[63]
- A substate of L1 is entered when the conditions for L1 PM Substates are satisfied (see Section 5.5).
  - The L1 PM Substate must be L1.0 when L1.Idle is entered or exited.
- Next state is Recovery if exit from Electrical Idle is detected on any Lane of a configured Link, or directed after remaining in this substate for a minimum of 40 ns in speeds other than 2.5 GT/s.
  - Ports are not required to arm the Electrical Idle exit detectors on all Lanes of the Link.
  - Note: A minimum stay of 40 ns is required in this substate in speeds other than 2.5 GT/s to account for the delay in the logic levels to arm the Electrical Idle detection circuitry in case the Link enters L1 and immediately exits the L1 state.
  - A Port is allowed to set the directed_speed_change variable to 1b following identical rules described in L0 for setting this variable. When making such a transition, the changed_speed_recovery variable must be reset to 0b. A Port may also go through Recovery back to L0 and then set the directed_speed_change variable to 1b on the transition from L0 to Recovery.
  - A Port is also allowed to enter Recovery from L1 if directed to change the Link width. The Port must follow identical rules for changing the Link width as described in the L0 state.
- Next state is Recovery after a 100 ms timeout if the current data rate is 8.0 GT/s or higher and the Port's Receivers do not meet the $Z_{RX\text{-}DC}$ specification for 2.5 GT/s). All Ports are permitted,

[63] The common mode being driven must meet the Absolute Delta Between DC Common Mode During L0 and Electrical Idle ($V_{TX\text{-}CM\text{-}DC\text{-}ACTIVE\text{-}IDLE\text{-}DELTA}$) specification (see Table 8-7).

but not encouraged, to implement the timeout and transition to Recovery when the data rate is 8.0 GT/s or higher.

- This timeout is not affected by the L1 PM Substates mechanism.

IMPLEMENTATION NOTE

100 ms Timeout in L1

Ports that meet the $Z_{RX\text{-}DC}$ specification for 2.5 GT/s while in the L1.Idle state and are therefore not required to implement the 100 ms timeout and transition to Recovery should avoid implementing it, since it will reduce the power savings expected from the L1 state.

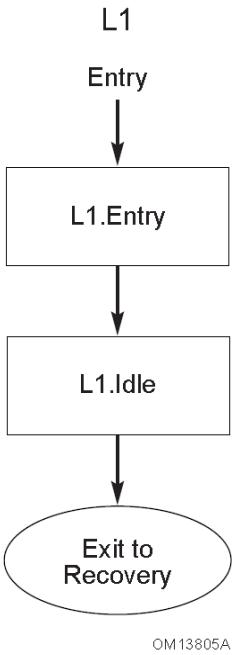


Figure 4-29: L1 Substate Machine

4.2.6.8 *L2*

The L2 substate machine is shown in Figure 4-30.

4.2.6.8.1 L2.Idle

- All Receivers must meet the $Z_{RX\text{-}DC}$ specification for 2.5 GT/s within 1 ms (see Table 8-11).
- All configured Transmitters must remain in Electrical Idle for a minimum time of $T_{TX\text{-}IDLE\text{-}MIN}$.
  - The DC common mode voltage does not have to be within specification.
  - The Receiver needs to wait a minimum of $T_{TX\text{-}IDLE\text{-}MIN}$ to start looking for Electrical Idle Exit.

- For Downstream Lanes:
  - For a Root Port, the next state is Detect if a Beacon is received on at least Lane 0 or if directed.
    - Main power must be restored before entering Detect.
    - Note: "if directed" is defined as a higher layer decides to exit to Detect.
  - For a Switch, if a Beacon is received on at least Lane 0, the Upstream Port must transition to L2.TransmitWake.
- For Upstream Lanes:
  - The next state is Detect if Electrical Idle Exit is detected on any predetermined set of Lanes.
    - The predetermined set of Lanes must include but is not limited to any Lane which has the potential of negotiating to Lane 0 of a Link. For multi-Lane Links the number of Lanes in the predetermined set must be greater than or equal to two.
    - A Switch must transition any Downstream Lanes to Detect.
  - Next state is L2.TransmitWake for an Upstream Port if directed to transmit a Beacon.
    - Note: Beacons may only be transmitted on Upstream Ports in the direction of the Root Complex.

4.2.6.8.2 L2.TransmitWake

This state only applies to Upstream Ports.

- Transmit the Beacon on at least Lane 0.
- Next state is Detect if Electrical Idle exit is detected on any Upstream Port's Receiver that is in the direction of the Root Complex.
  - Note: Power is guaranteed to be restored when Upstream Receivers see Electrical Idle exited, but it may also be restored prior to Electrical Idle being exited.

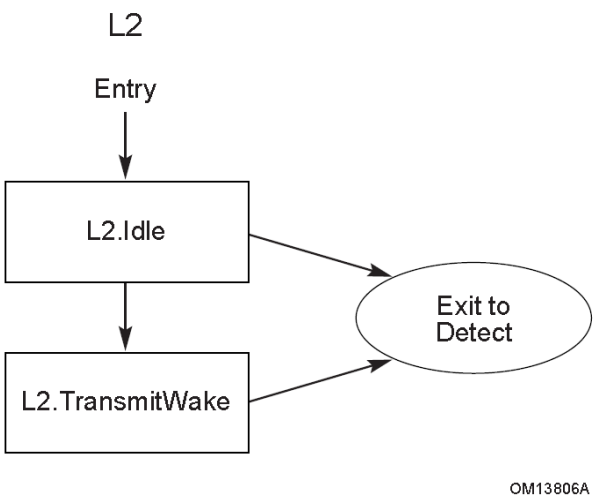


Figure 4-30: L2 Substate Machine

4.2.6.9 *Disabled*

- All Lanes transmit 16 to 32 TS1 Ordered Sets with the Disable Link bit asserted and then transition to Electrical Idle.
  - An EIOSQ must be sent prior to entering Electrical Idle.
  - The DC common mode voltage does not have to be within specification.[64]
- If an EIOSQ was transmitted and an EIOS was received on any Lane (even while transmitting TS1 with the Disable Link bit asserted), then:
  - LinkUp = 0b (False)
    - At this point, the Lanes are considered Disabled.
  - For Upstream Ports: All Receivers must meet the $Z_{RX\text{-}DC}$ specification for 2.5 GT/s within 1 ms (see Table 8-11).
  - For Upstream Ports: The next state is Detect when Electrical Idle exit is detected on at least one Lane.
- For Downstream Ports: The next state is Detect when directed (e.g., when the Link Disable bit is reset to 0b by software).
- For Upstream Ports: If no EIOS is received after a 2 ms timeout, the next state is Detect.

4.2.6.10 *Loopback*

The Loopback substate machine is shown in Figure 4-31.

---

[64] The common mode being driven does need to meet the Absolute Delta Between DC Common Mode During L0 and Electrical Idle ($V_{TX\text{-}CM\text{-}DC\text{-}ACTIVE\text{-}IDLE\text{-}DELTA}$) specification (see Table 8-7).

4.2.6.10.1 Loopback.Entry

- LinkUp = 0b (False)
- The Link and Lane numbers received in the TS1 or TS2 Ordered Sets are ignored by the Receiver while in this substate.
- Loopback master requirements:
  - If Loopback.Entry was entered from Configuration.Linkwidth.Start, determine the highest common data rate of the data rates supported by the master and the data rates received in two consecutive TS1 or TS2 Ordered Sets on any active Lane at the time the transition to Loopback.Entry occurred. If the current data rate is not the highest common data rate:
    - Transmit 16 consecutive TS1 Ordered Sets with the Loopback bit asserted, followed by an EIOSQ, and then transition to Electrical Idle for 1 ms. During the period of Electrical Idle, change the data rate to the highest common data rate.
    - If the highest common data rate is 5.0 GT/s, the slave's transmitter de-emphasis is controlled by setting the Selectable De-emphasis bit of the transmitted TS1 Ordered Sets to the desired value (1b = -3.5 dB, 0b = -6 dB).
    - For data rates of 5.0 GT/s and above, the master is permitted to choose its own transmitter settings in an implementation-specific manner, regardless of the settings it transmitted to the slave.
    - Note: If Loopback is entered after LinkUp has been set to 1b, it is possible for one Port to enter Loopback from Recovery and the other to enter Loopback from Configuration. The Port that entered from Configuration might attempt to change data rate while the other Port does not. If this occurs, the results are undefined. The test set-up must avoid such conflicting directed clauses.
  - Transmit TS1 Ordered Sets with the Loopback bit asserted.
    - The master is also permitted to assert the Compliance Receive bit of TS1 Ordered Sets transmitted in Loopback.Entry, including those transmitted before a data rate change. If it asserts the Compliance Receive bit, it must not deassert it again while in the Loopback.Entry state. This usage model might be helpful for test and validation purposes when one or both Ports have difficulty obtaining bit lock, Symbol lock, or Block alignment after a data rate change. The ability to set the Compliance Receive bit is implementation-specific.
  - Next state is Loopback.Active after 2 ms if the Compliance Receive bit of the transmitted TS1 Ordered Sets is asserted.
  - Next state is Loopback.Active if the Compliance Receive bit of the transmitted TS1 Ordered Sets is deasserted and an implementation-specific set of Lanes receive two consecutive TS1 Ordered Sets with the Loopback bit asserted.

  If the data rate was changed, the master must take into account the amount of time the slave can be in Electrical Idle and transmit a sufficient number of TS1 Ordered Sets for

the slave to acquire Symbol lock or Block alignment before proceeding to Loopback.Active.

IMPLEMENTATION NOTE

Lane Numbering with 128b/130b Encoding in Loopback

If the current data rate uses 128b/130b encoding and Lane numbers have not been negotiated, it is possible that the master and slave will not be able to decode received information because their Lanes are using different scrambling LFSR seed values (since the LFSR seed values are determined by the Lane numbers). This situation can be avoided by allowing the master and slave to negotiate Lane numbers before directing the master to Loopback, directing the master to assert the Compliance Receive bit during Loopback.Entry, or by using some other method of ensuring that the LFSR seed values match.

- Next state is Loopback.Exit after an implementation-specific timeout of less than 100 ms.

- Loopback slave requirements:
  - If Loopback.Entry was entered from Configuration.Linkwidth.Start, determine the highest common data rate of the data rates supported by the slave and the data rates received in the two consecutive TS1 Ordered Sets that directed the slave to this state. If the current data rate is not the highest common data rate:
    - Transmit an EIOSQ, and then transition to Electrical Idle for 2 ms. During the period of Electrical Idle, change the data rate to the highest common data rate.
    - If operating in full swing mode and the highest common data rate is 5.0 GT/s, set the transmitter's de-emphasis to the setting specified by the Selectable De-emphasis bit received in the TS1 Ordered Sets that directed the slave to this state. The de-emphasis is -3.5 dB if the Selectable De-emphasis bit was 1b, and it is -6 dB if the Selectable De-emphasis bit was 0b.
    - If the highest common data rate is 8.0 GT/s or higher and EQ TS1 Ordered Sets directed the slave to this state, set the transmitter to the settings specified by the Preset field of the EQ TS1 Ordered Sets. See Section 4.2.3.2. If the highest common data rate is 8.0 GT/s or higher but standard TS1 Ordered Sets directed the slave to this state, the slave is permitted to use its default transmitter settings.
  - Next state is Loopback.Active if the Compliance Receive bit of the TS1 Ordered Sets that directed the slave to this state was asserted.
    - The slave's transmitter does not need to transition to transmitting looped-back data on any boundary, and it is permitted to truncate any Ordered Set in progress.
  - Else, the slave transmits TS1 Ordered Sets with Link and Lane numbers set to PAD.
    - Next state is Loopback.Active if the data rate is 2.5 GT/s or 5.0 GT/s and Symbol lock is obtained on all active Lanes.

- Next state is Loopback.Active if the data rate is 8.0 GT/s or higher and two consecutive TS1 Ordered Sets are received on all active Lanes. The equalization settings specified by the received TS1 Ordered Sets must be evaluated and applied to the transmitter if the value of the EC field is appropriate for the slave's Port direction (10b or 11b) and the requested setting is a preset or a set of valid coefficients. (Note: This is the equivalent behavior for the Recovery.Equalization state.) Optionally, the slave can accept both EC field values. If the settings are applied, they must take effect within 500 ns of being received, and they must not cause the transmitter to violate any electrical specification for more than 1 ns. Unlike Recovery.Equalization, the new settings are not reflected in the TS1 Ordered Sets that the slave transmits.
- When using 8b/10b encoding, the slave's transmitter must transition to transmitting looped-back data on a Symbol boundary, but it is permitted to truncate any Ordered Set in progress. When using 128b/ 130b encoding, the slave's transmitter does not need to transition to transmitting looped-back data on any boundary, and is permitted to truncate any Ordered Set in progress.

4.2.6.10.2 Loopback.Active

- The loopback master must send valid encoded data. The loopback master must not transmit EIOS as data until it wants to exit Loopback. When operating with 128b/130b encoding, loopback masters must follow the requirements of Section 4.2.2.5.
- A loopback slave is required to retransmit the received encoded information as received, with the polarity inversion determined in Polling applied, while continuing to perform clock tolerance compensation:
  - SKPs must be added or deleted on a per-Lane basis as outlined in Section 4.2.7 with the exception that SKPs do not have to be simultaneously added or removed across Lanes of a configured Link.
    - For 8b/10b encoding, if a SKP Ordered Set retransmission requires adding a SKP Symbol to accommodate timing tolerance correction, the SKP Symbol is inserted in the retransmitted Symbol stream anywhere adjacent to a SKP Symbol in the SKP Ordered Set following the COM Symbol. The inserted SKP Symbol must be of the same disparity as the received SKPs Symbol(s) in the SKP Ordered Set.
    - For 8b/10b encoding, if a SKP Ordered Set retransmission requires dropping a SKP Symbol to accommodate timing tolerance correction, the SKP Symbol is simply not retransmitted.
    - For 128b/130b encoding, if a SKP Ordered Set retransmission requires adding SKP Symbols to accommodate timing tolerance correction, four SKP Symbols are inserted in the retransmitted Symbol stream prior to the SKP_END Symbol in the SKP Ordered Set.
    - For 128b/130b encoding, if a SKP Ordered Set retransmission requires dropping SKP Symbols to accommodate timing tolerance correction, four SKP Symbols

prior to the SKP_END Symbol in the SKP Ordered Set are simply not retransmitted.

- No modifications of the received encoded data (except for polarity inversion determined in Polling) are allowed by the loopback slave even if it is determined to be an invalid encoding (i.e., no legal translation to a control or data value possible for 8b/10b encoding or invalid Sync Header or invalid Ordered Set for 128b/130b encoding).

- Next state of the loopback slave is Loopback.Exit if one of the following conditions apply:
  - If directed or if four consecutive EIOSs are received on any Lane. It must be noted that in 8b/10b encoding, the receiving four consecutive EIOS indicates that the Lane received four consecutive sets of COM, IDL, IDL, IDL or alternatively, two out of three K28.3 (IDL) Symbols in each of the four consecutive sets of transmitted EIOS. In 128b/130b encoding, receiving four consecutive EIOS indicates receiving the full 130-bit EIOS in the first three and the first four Symbols following a 01b Sync Header in the last EIOS.
  - Optionally, if current Link speed is 2.5 GT/s and an EIOS is received or Electrical Idle is detected/inferred on any Lane.
    - Note: As described in Section 4.2.4.3, an Electrical Idle condition may be inferred if any of the configured Lanes remained electrically idle continuously for 128 µs by not detecting an exit from Electrical Idle in the entire 128 µs window.
  - A loopback slave must be able to detect an Electrical Idle condition on any Lane within 1 ms of the EIOS being received by the loopback slave.
  - Note: During the time after an EIOS is received and before Electrical Idle is actually detected by the Loopback Slave, the Loopback Slave may receive a bit stream undefined by the encoding scheme, which may be looped back by the transmitter.
  - The $T_{TX\text{-}IDLE\text{-}SET\text{-}TO\text{-}IDLE}$ parameter does not apply in this case since the loopback slave may not even detect Electrical Idle until as much as 1 ms after the EIOS.

- The next state of the loopback master is Loopback.Exit if directed.

4.2.6.10.3 Loopback.Exit

- The loopback master sends an EIOS for Ports that support only the 2.5 GT/s data rate and eight consecutive EIOSs for Ports that support greater than 2.5 GT/s data rate, and optionally for Ports that only support the 2.5 GT/s data rate, irrespective of the current Link speed, and enters Electrical Idle on all Lanes for 2 ms.
  - The loopback master must transition to a valid Electrical Idle condition[65] on all Lanes within $T_{TX\text{-}IDLE\text{-}SET\text{-}TO\text{-}IDLE}$ after sending the last EIOS.
  - Note: The EIOS can be useful in signifying the end of transmit and compare operations that occurred by the loopback master. Any data received by the loopback master after any EIOS is received should be ignored since it is undefined.

---

[65] The common mode being driven does not need to meet the Absolute Delta Between DC Common Mode During L0 and Electrical Idle ($V_{TX\text{-}CM\text{-}DC\text{-}ACTIVE\text{-}IDLE\text{-}DELTA}$) specification (see Table 8-7).

- The loopback slave must enter Electrical Idle on all Lanes for 2 ms.
  - Before entering Electrical Idle the loopback slave must Loopback all Symbols that were received prior to detecting Electrical Idle. This ensures that the loopback master may see the EIOS to signify the logical end of any Loopback send and compare operations.
- The next state of the loopback master and loopback slave is Detect.

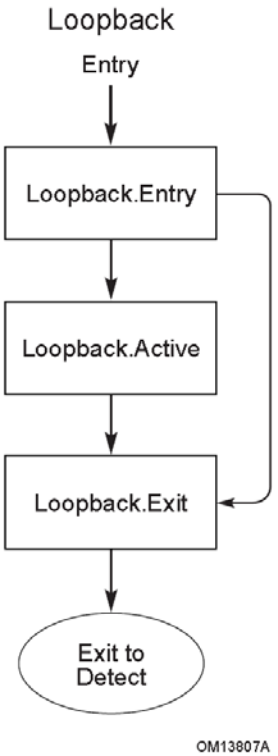


Figure 4-31: Loopback Substate Machine

4.2.6.11 *Hot Reset*

- Lanes that were directed by a higher Layer to initiate Hot Reset:
  - All Lanes in the configured Link transmit TS1 Ordered Sets with the Hot Reset bit asserted and the configured Link and Lane numbers.
  - If two consecutive TS1 Ordered Sets are received on any Lane with the Hot Reset bit asserted and configured Link and Lane numbers, then:
    - LinkUp = 0b (False)
    - If no higher Layer is directing the Physical Layer to remain in Hot Reset, the next state is Detect
    - Otherwise, all Lanes in the configured Link continue to transmit TS1 Ordered Sets with the Hot Reset bit asserted and the configured Link and Lane numbers.
  - Otherwise, after a 2 ms timeout next state is Detect.

- Lanes that were not directed by a higher Layer to initiate Hot Reset (i.e., received two consecutive TS1 Ordered Sets with the Hot Reset bit asserted on any configured Lanes):
  - LinkUp = 0b (False)
  - If any Lane of an Upstream Port of a Switch receives two consecutive TS1 Ordered Sets with the Hot Reset bit asserted, all configured Downstream Ports must transition to Hot Reset as soon as possible.
    - Any optional crosslinks on the Switch are an exception to this rule and the behavior is system specific.
  - All Lanes in the configured Link transmit TS1 Ordered Sets with the Hot Reset bit asserted and the configured Link and Lane numbers.
  - If two consecutive TS1 Ordered Sets were received with the Hot Reset bit asserted and the configured Link and Lane numbers, the state continues to be Hot Reset and the 2 ms timer is reset.
  - Otherwise, the next state is Detect after a 2 ms timeout.

Note: Generally, Lanes of a Downstream or optional crosslink Port will be directed to Hot Reset, and Lanes of an Upstream or optional crosslink Port will enter Hot Reset by receiving two consecutive TS1 Ordered Sets with the Hot Reset bit asserted on any configured Lanes, from Recovery.Idle state.

4.2.7 Clock Tolerance Compensation

SKP Ordered Sets (defined below) are used to compensate for differences in frequencies between bit rates at two ends of a Link. The Receiver Physical Layer logical sub-block must include elastic buffering which performs this compensation. The interval between SKP Ordered Set transmissions is derived from the absolute value of the Transmit and Receive clock frequency difference specified in Table 8-3.

The specification supports two kinds of clocking where the Tx and Rx Refclk rates differ. One allows for a worst case 600 ppm difference with no SSC (Separate Reference Clocks With No SSC - SRNS), and the other for a 5600 ppm difference for separate Refclks utilizing independent SSC (Separate Reference Clocks with Independent SSC - SRIS) (SSC introduces a 5000 ppm difference, and Tx/Rx crystal tolerance introduces another 600 ppm). Note that the common Refclk architecture utilizes the same Refclk for Tx and Rx and so does not introduce any difference between the Tx and Rx Refclk rates.

Specific form factor specifications are permitted to require the use of only SRIS, only SRNS, or to provide a mechanism for clocking architecture selection. Upstream Ports are permitted to implement support for any combination of SRIS and SRNS (including no support for either), but must conform to the requirements of any associated form factor specification. Downstream Ports supporting SRIS must also support SRNS unless the port is only associated with a specific form factor(s) which modifies these requirements. Port configuration to satisfy the requirements of a specific associated form factor is implementation specific.

If the clock tolerance is 600 ppm, on average, the Tx and Rx clocks can shift one clock every 1666 clocks. If the clock tolerance is 5600 ppm, a shift of one clock can occur every 178 clocks.

If the receiver is capable of operating with SKP Ordered Sets being generated at the rate used in SRNS even though the Port is running in SRIS, the Port is permitted to Set its bit for the appropriate data rate in the Lower SKP OS Reception Supported Speeds Vector field of the Link Capabilities 2 register. If the transmitter is capable of operating with SKP Ordered Sets being generated at the rate used in SRNS even though the Port is running in SRIS, the Port is permitted to Set its bit for the appropriate data rate in the Lower SKP OS Generation Supported Speeds Vector field of the Link Capabilities 2 register. System software must check that the bit is Set in the Lower SKP OS Reception Supported Speeds Vector field before setting the appropriate data rate's bit in the link partner's Enable Lower SKP OS Generation Vector field of the Link Control 3 register. Any software transparent extension devices (such as a repeater) present on a Link must also support lower SKP OS Generation for system software to set the bit in the Enable Lower SKP OS Generation Vector field. Software determination of such support in such extension devices is implementation specific. When the bit for the data rate that the link is running in is Set in the Enable Lower SKP OS Generation Vector field, the transmitter schedules SKP Ordered Set generation in L0 at the rate used in SRNS, regardless of which clocking architecture the link is running in. In other LTSSM states, SKP Ordered Set scheduling is at the appropriate rate for the clocking architecture.

Components supporting SRIS may need more entries in their elastic buffers than designs supporting SRNS only. This requirement takes into account the extra time it may take to schedule a SKP Ordered Set if the latter falls immediately after a maximum payload sized packet.

4.2.7.1 *SKP Ordered Set for 8b/10b Encoding*

When using 8b/10b encoding, a transmitted SKP Ordered Set is a COM Symbol followed by three SKP Symbols, except as is allowed for a Loopback Slave in the Loopback.Active LTSSM state. A received SKP Ordered Set is a COM Symbol followed by one to five SKP Symbols. See Section 4.3.6.7 for Retimer rules on SKP Ordered Set modification.

4.2.7.2 *SKP Ordered Set for 128b/130b Encoding*

When using 128b/130b encoding, a transmitted SKP Ordered Set is 16 Symbols, and a received SKP Ordered set can be 8, 12, 16, 20, or 24 Symbols. See Section 4.3.6.7 for Retimer rules on SKP Ordered Set modification.

There are two SKP Ordered Set formats defined for 128b/130b encoding as shown in Table 4-16: and Table 4-17. Both formats contain one to five groups of four SKP Symbols followed by a final group of four Symbols indicated by a SKP_END or SKP_END_CTL Symbol. When operating at 8.0 GT/s, only the Standard SKP Ordered Set is used. When operating at 16.0 GT/s, both the Standard and Control SKP Ordered Sets are used. All statements in this specification that do not refer to a specific SKP Ordered Set format apply to both formats. When a SKP Ordered Set is transmitted, all Lanes must transmit the same format of SKP Ordered Set – all Lanes must transmit the Standard SKP Ordered Set, or all Lanes must transmit the Control SKP Ordered Set.

The Standard SKP Ordered Set contains information following the SKP_END Symbol that is based on the LTSSM state and the sequence of Blocks. When in the Polling.Compliance state, the Symbols contain the Lane's error status information (see Section 4.2.6.2.2. for more information). Otherwise, the Symbols contain the Lane's scrambling LFSR value, and a Data Parity bit when the

SKP Ordered Set follows a Data Block. The Control SKP Ordered Set contains three Data Parity bits and additional information following the SKP_END_CTL Symbol.

When operating at 8.0 GT/s, the Data Parity bit of the Standard SKP Ordered Set is the even parity of the payload of all Data Blocks communicated by a Lane and is computed for each Lane independently[66]. Upstream and Downstream Port Transmitters compute the parity as follows:

- Parity is initialized when a SDS Ordered Set is transmitted.
- Parity is updated with each bit of a Data Block's payload after scrambling has been performed.
- The Data Parity bit of a Standard SKP Ordered Set transmitted immediately following a Data Block is set to the current parity.
- Parity is initialized after a Standard SKP Ordered Set is transmitted.

Upstream and Downstream Port Receivers compute and act on the parity as follows:

- Parity is initialized when a SDS Ordered Set is received.
- Parity is updated with each bit of a Data Block's payload before de-scrambling has been performed.
- When a Standard SKP Ordered Set is received immediately following a Data Block, each Lane compares the received Data Parity bit to its calculated parity. If a mismatch is detected, the receiver must set the bit of the Port's Lane Error Status register that corresponds to the Lane's default Lane number. The mismatch is not a Receiver Error and must not cause a Link retrain.
- Parity is initialized when a Standard SKP Ordered Set is received.

When operating at 16.0 GT/s, the Data Parity bits of both the Standard SKP Ordered Set and the Control SKP Ordered Set is the even parity of the payload of all Data Blocks communicated by a Lane and is computed for each Lane independently. Upstream and Downstream Port Transmitters compute the parity as follows:

- Parity is initialized when the LTSSM is in Recovery.Speed.
- Parity is initialized when a SDS Ordered Set is transmitted.
- Parity is updated with each bit of a Data Block's payload after scrambling has been performed.
- The Data Parity bit of a Standard SKP Ordered Set transmitted immediately following a Data Block is set to the current parity.
- The Data Parity, First Retimer Data Parity, and Second Retimer Data Parity bits of a Control SKP Ordered Set are all set to the current parity.
- Parity is initialized after a Control SKP Ordered Set is transmitted. However, parity is NOT initialized after a Standard SKP Ordered Set is transmitted.

Upstream and Downstream Port Receivers compute and act on the parity as follows:

- Parity is initialized when the LTSSM is in Recovery.Speed.
- Parity is initialized when a SDS Ordered Set is received.

---

[66] The requirements for 8.0 GT/s operation documented here are identical to those in Revision 3.1 of the PCI Express Base Specification.

- Parity is updated with each bit of a Data Block's payload before de-scrambling has been performed.
- When a Control SKP Ordered Set is received, each Lane compares the received Data Parity bit to its calculated parity. If a mismatch is detected, the receiver must set the bit of the Port's Local Data Parity Mismatch Status register that corresponds to the Lane's default Lane number. The mismatch is not a Receiver Error and must not cause a Link retrain.
- When a Control SKP Ordered Set is received, each Lane compares the received First Retimer Data Parity bit to its calculated parity. If a mismatch is detected, the receiver must set the bit of the Port's First Retimer Data Parity Mismatch Status register that corresponds to the Lane's default Lane number. The mismatch is not a Receiver Error and must not cause a Link retrain.
- When a Control SKP Ordered Set is received, each Lane compares the received Second Retimer Data Parity bit to its calculated parity. If a mismatch is detected, the receiver must set the bit of the Port's Second Retimer Data Parity Mismatch Status register that corresponds to the Lane's default Lane number. The mismatch is not a Receiver Error and must not cause a Link retrain.
- When a Standard SKP Ordered Set is received immediately following a Data Block, the receiver is permitted to compare the received Data Parity bit to its calculated parity bit. However, the results of such a comparison must not affect the state of the Lane Error Status register.
- Parity is initialized when a Control SKP Ordered Set is received. However, parity is NOT initialized when a Standard SKP Ordered Set is received.

See Section 4.3.6.7 for the definition of First Retimer and Second Retimer, and for Retimer Pseudo Port requirements for parity computation and modification of the First Retimer Data Parity and Second Retimer Data Parity bits of Control SKP Ordered Sets.

IMPLEMENTATION NOTE

LFSR in Standard SKP Ordered Set

The LFSR value is transmitted to enable trace tools to be able to function even if they need to reacquire block alignment in the midst of a bit stream. Since trace tools cannot force the link to enter Recovery, they can reacquire bit lock, if needed, and monitor for the SKP Ordered Set to obtain Block alignment and perform Lane-to-Lane de-skew. The LFSR value from the SKP Ordered Set can be loaded into its LFSR to start interpreting the bit stream. It must be noted that with a bit stream one may alias to a SKP Ordered Set on a non-Block boundary. The trace tools can validate their Block alignment by using implementation specific means such as receiving a fixed number of valid packets or Sync Headers or subsequent SKP Ordered Sets.

Table 4-16: Standard SKP Ordered Set with 128b/130b Encoding

Symbol Number	Value	Description
0 through (4*N - 1) [N can be 1 through 5]	AAh	SKP Symbol. Symbol 0 is the SKP Ordered Set identifier.
4*N	E1h	SKP_END Symbol. Signifies the end of the SKP Ordered Set after three more Symbols.
4*N + 1	00-FFh	(i) If LTSSM state is Polling.Compliance: AAh (ii) Else if prior block was a Data Block: Bit[7] = Data Parity Bit[6:0] = LFSR[22:16] (iii) Else: Bit[7] = ~LFSR[22] Bit[6:0] = LFSR[22:16]
4*N + 2	00-FFh	(i) If the LTSSM state is Polling.Compliance: Error_Status[7:0] (ii) Else LFSR[15:8]
4*N + 3	00-FFh	(i) If the LTSSM state is Polling.Compliance: ~Error_Status[7:0] (ii) Else: LFSR[7:0]

The Control SKP Ordered Set is different from the Standard SKP Ordered Set in the last 4 Symbols. It is used to communicate the parity bits, as computed by each Retimer, in addition to the Data Parity bit computed by the Upstream/ Downstream Port. It may also be used for Lane Margining at a Retimer's Receiver, as described below.

Table 4-17: Control SKP Ordered Set with 128b/130b Encoding

Symbol Number	Value	Description
0 through (4*N - 1) [N can be 1 through 5]	AAh	SKP Symbol. Symbol 0 is the SKP Ordered Set identifier.
4*N	78h	SKP_END_CTL Symbol. Signifies the end of the Control SKP Ordered Set after three more Symbols.
4*N + 1	00-FFh	Bit 7: Data Parity Bit 6: First Retimer Data Parity Bit 5: Second Retimer Parity Bits [4:0]: Margin CRC [4:0]
4*N + 2	00-FFh	Bit 7: Margin Parity Bits [6:0]: Refer to Section 4.2.13.1
4*N + 3	00-FFh	Bits [7:0]: Refer to Section 4.2.13.1

The 'Margin CRC[4:0]' is computed from Bits [6:0] in Symbols 4N+2 (referred to as d[6:0] in the equations below, where d[0] is Bit 0 of Symbol 4N+2, d[1] is Bit 1 of Symbol 4N+2, … d[6] is Bit 6 of Symbol 4N+2) and Bits [7:0] of Symbol 4N+3 (referred to as d[14:7], where d[7] is Bit 0 of Symbol 4N+3, d[8] is Bit 1 of Symbol 4N+3, .. d[14] is Bit 7 of Symbol 4N+3) as follows:

Margin CRC[0] = d[0] ^ d[3] ^ d[5] ^ d[6] ^ d[9] ^ d[10] ^ d[11] ^ d[12] ^ d[13]
Margin CRC[1] = d[1] ^ d[4] ^ d[6] ^ d[7] ^ d[10] ^ d[11] ^ d[12] ^ d[13] ^ d[14]
Margin CRC[2] = d[0] ^ d[2] ^ d[3] ^ d[6] ^ d[7] ^ d[8] ^ d[9] ^ d[10] ^ d[14]
Margin CRC[3] = d[1] ^ d[3] ^ d[4] ^ d[7] ^ d[8] ^ d[9] ^ d[10] ^ d[11]
Margin CRC[4] = d[2] ^ d[4] ^ d[5] ^ d[8] ^ d[9] ^ d[10] ^ d[11] ^ d[12]

'Margin Parity' is the even parity of Bits [4:0] of Symbol 4N+1, Bits [6:0] of Symbol 4N+2, and Bits [7:0] of Symbol 4N+3 (i.e., Margin Parity = Margin CRC[0] ^ Margin CRC[1] ^ Margin CRC[2] ^ Margin CRC[3] ^ Margin CRC[4] ^ d[0] ^ d[1] ^ d[2] ^ d[3] ^ d[4] ^ d[5] ^ d[6] ^ d[7] ^ d[8] ^ d[9] ^ d[10] ^ d[11] ^ d[12] ^ d[13] ^ d[14]).

The rules for generating and checking the Margin CRC and Margin Parity are described in Section 4.2.1.3.

IMPLEMENTATION NOTE

Error Protection in Control SKP Ordered Set

The 21 bits in Symbol 4N+1 (Bits [4:0]), Symbol 4N+2 (Bits[7:0]) and Symbol 4N+3 (Bits[7:0]) is protected by 5 bits of CRC and one bit of parity, leaving 15 bits for information passing. The parity bit provides detection against odd number of bit-flips (e.g., 1-bit, 3-bit), whereas the CRC provides guaranteed detection of 1-bit and 2-bit flips; thus resulting in a triple bit flip detection guarantee over the 21 bits as well as guaranteed detection of burst errors of length 5. The 5-bit CRC is derived from the primitive polynomial: $x^5 + x^2 + 1$.

Since these 21 bits are not part of a TLP, repeated delivery of the same content provides delivery guarantee. This is achieved through architected registers. Downstream commands are sent from the Downstream Port, reflecting the contents of an architected register whereas the upstream status that passes the error checking is updated into a status register in the Downstream Port. Software thus has a mechanism to issue a command and wait for the status to be reflected back before issuing a new command. Thus, these 15 bits of information act as a micro-packet.

4.2.7.3 *Rules for Transmitters*

- All Lanes shall transmit Symbols at the same frequency (the difference between bit rates is 0 ppm within all multi-Lane Links).
- When transmitted, SKP Ordered Sets of the same length shall be transmitted simultaneously on all Lanes of a multi-Lane Link, except as allowed for a Loopback Slave in the Loopback.Active LTSSM State (see Section 4.2.4.10 and Table 8-3 for the definition of simultaneous in this context).
- The transmitted SKP Ordered Set when using 8b/10b encoding must follow the definition in Section 4.2.7.1.
- The transmitted SKP Ordered Set when using 128b/130b encoding must follow the definition in Section 4.2.7.2.
- When using 8b/10b encoding:
  - If the Link is not operating in SRIS, or the bit corresponding to the current Link speed is Set in the Enable Lower SKP OS Generation Vector field and the LTSSM is in L0, a SKP Ordered Set must be scheduled for transmission at an interval between 1180 and 1538 Symbol Times.
  - If the Link is operating in SRIS and either the bit corresponding to the current Link speed is Clear in the Enable Lower SKP OS Generation Vector field or the LTSSM is not in L0, a SKP Ordered Set must be scheduled for transmission at an interval of less than 154 Symbol Times.
- When using 128b/130b encoding:
  - If the Link is not operating in SRIS, or the bit corresponding to the current Link speed is Set in the Enable Lower SKP OS Generation Vector field and the LTSSM is in L0, a

SKP Ordered Set must be scheduled for transmission at an interval between 370 and 375 Blocks. Loopback Slaves must meet this requirement until they start looping back the incoming bit stream.

- If the Link is operating in SRIS and either the bit corresponding to the current Link speed is Clear in the Enable Lower SKP OS Generation Vector field or the LTSSM is not in L0, a SKP Ordered Set must be scheduled for transmission at an interval less than 38 Blocks. Loopback Slaves must meet this requirement until they start looping back the incoming bit stream.
- When the LTSSM is in the Loopback state and the Link is not operating in SRIS, the Loopback Master must schedule two SKP Ordered Sets to be transmitted, at most two Blocks apart from each other, at an interval between 370 to 375 blocks.
- When the LTSSM is in the Loopback state and the Link is operating in SRIS, the Loopback Master must schedule two SKP Ordered Sets to be transmitted, at most two Blocks apart from each other, at an interval of less than 38 Blocks.
- The Control SKP Ordered Set is transmitted only at the following times:
  - When the data rate is 16.0 GT/s and transmitting a Data Stream. SKP Ordered Sets transmitted within a Data Stream must alternate between the Standard SKP Ordered Set and the Control SKP Ordered Set.
  - When the current data rate is 16.0 GT/s and the LTSSM enters the Configuration.Idle state or Recovery.Idle state. See sections 4.2.6.3.6 and 4.2.6.4.5 for more information. Transmission of this instance of the Control SKP Ordered Set is not subject to any minimum scheduling interval requirements described above. Transmitters are permitted, but not required, to reset their SKP Ordered Set scheduling interval timer after transmitting this instance of the Control SKP Ordered Set.

- Scheduled SKP Ordered Sets shall be transmitted if a packet or Ordered Set is not already in progress, otherwise they are accumulated and then inserted consecutively at the next packet or Ordered Set boundary. Note: When using 128b/130b encoding, SKP Ordered Sets cannot be transmitted in consecutive Blocks within a Data Stream. See Section 4.2.2.3.2 for more information.
- SKP Ordered Sets do not count as an interruption when monitoring for consecutive Symbols or Ordered Sets (e.g., eight consecutive TS1 Ordered Sets in Polling.Active).
- When using 8b/10b encoding: SKP Ordered Sets must not be transmitted while the Compliance Pattern or the Modified Compliance Pattern (see Section 4.2.8) is in progress during Polling.Compliance if the Compliance SOS bit of the Link Control 2 register is 0b. If the Compliance SOS bit of the Link Control 2 register is 1b, two consecutive SKP Ordered Sets must be sent (instead of one) for every scheduled SKP Ordered Set time interval while the Compliance Pattern or the Modified Compliance Pattern is in progress when using 8b/10b encoding.
- When using 128b/130b encoding: The Compliance SOS register bit has no effect. While in Polling.Compliance, Transmitters must not transmit any SKP Ordered Sets other than those specified as part of the Modified Compliance Pattern in Section 4.2.11.

- Any and all time spent in a state when the Transmitter is electrically idle does not count in the scheduling interval used to schedule the transmission of SKP Ordered Sets.
- It is recommended that any counter(s) or other mechanisms used to schedule SKP Ordered Sets be reset any time when the Transmitter is electrically idle.

4.2.7.4 *Rules for Receivers*

- Receivers shall recognize received SKP Ordered Sets as defined in Section 4.2.7.1 when using 8b/10b encoding and as defined in Section 4.2.7.2 when using 128b/130b encoding.
  - The length of the received SKP Ordered Sets shall not vary from Lane-to-Lane in a multi-Lane Link, except as may occur during Loopback.Active.
- Receivers shall be tolerant to receive and process SKP Ordered Sets at an average interval between 1180 to 1538 Symbol Times when using 8b/10b encoding and 370 to 375 blocks when using 128b/130b encoding when the Link is not operating in SRIS or its bit for the current Link speed is Set in the Lower SKP OS Reception Supported Speeds Vector field. Receivers shall be tolerant to receive and process SKP Ordered Sets at an average interval of less than 154 Symbol Times when using 8b/10b encoding and less than 38 blocks when using 128b/130b encoding when the Link is operating in SRIS.
  - Note: Since Transmitters in electrical idle are not required to reset their mechanism for time-based scheduling of SKP Ordered Sets, Receivers shall be tolerant to receive the first time-scheduled SKP Ordered Set following electrical idle in less than the average time interval between SKP Ordered Sets.
- For 8.0 GT/s and above data rates, in L0 state, Receivers must check that each SKP Ordered Set is preceded by a Data Block with an EDS token.
- Receivers shall be tolerant to receive and process consecutive SKP Ordered Sets in 2.5 GT/s and 5.0 GT/s data rates.
  - Receivers shall be tolerant to receive and process SKP Ordered Sets that have a maximum separation dependent on the Max_Payload_Size a component supports. For 2.5 GT/s and 5.0 GT/s data rates, the formula for the maximum number of Symbols (N) between SKP Ordered Sets is: N = 1538 + (Max_payload_size_byte+28). For example, if Max_Payload_Size is 4096 bytes, N = 1538 + 4096 + 28 = 5662.

4.2.8 Compliance Pattern in 8b/10b Encoding

During Polling, the Polling.Compliance substate must be entered from Polling.Active based on the conditions described in Section 4.2.6.2.1. The compliance pattern consists of the sequence of 8b/10b Symbols K28.5, D21.5, K28.5, and D10.2 repeating. The Compliance sequence is as follows:

Symbol	K28.5	D21.5	K28.5	D10.2
Current Disparity	Negative	Positive	Positive	Negative
Pattern	0011111010	1010101010	1100000101	0101010101

The first Compliance sequence Symbol must have negative disparity. It is permitted to create a disparity error to align the running disparity to the negative disparity of the first Compliance sequence Symbol.

For any given device that has multiple Lanes, every eighth Lane is delayed by a total of four Symbols. A two Symbol delay occurs at both the beginning and end of the four Symbol Compliance Pattern sequence. A x1 device, or a xN device operating a Link in x1 mode, is permitted to include the Delay Symbols with the Compliance Pattern.

This delay sequence on every eighth Lane is then:

Symbol:	D	D	K28.5	D21.5	K28.5	D10.2	D	D

Where D is a K28.5 Symbol. The first D Symbol has negative disparity to align the delay disparity with the disparity of the Compliance sequence.

After the eight Symbols are sent, the delay Symbols are advanced to the next Lane, until the delay Symbols have been sent on all eight lanes. Then the delay Symbols cycle back to Lane 0, and the process is repeated. It is permitted to advance the delay sequence across all eight lanes, regardless of the number of lanes detected or supported. An illustration of this process is shown below:

Lane 0	D	D	K28.5-	D21.5	K28.5+	D10.2	D	D	K28.5-	D21.5	K28.5+	D10.2
Lane 1	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2	D	D	K28.5-	D21.5
Lane 2	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2
Lane 3	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2
Lane 4	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2
Lane 5	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2
Lane 6	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2
Lane 7	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2
Lane 8	D	D	K28.5-	D21.5	K28.5+	D10.2	D	D	K28.5-	D21.5	K28.5+	D10.2
Lane 9	K28.5-	D21.5	K28.5+	D10.2	K28.5-	D21.5	K28.5+	D10.2	D	D	K28.5-	D21.5

Key:

K28.5-	COM when disparity is negative, specifically: “0011111010”
K28.5+	COM when disparity is positive, specifically: “1100000101”
D21.5	Out of phase data Symbol, specifically: “1010101010”
D10.2	Out of phase data Symbol, specifically: “0101010101”
D	Delay Symbol K28.5 (with appropriate disparity)

This sequence of delays ensures interference between adjacent Lanes, enabling measurement of the compliance pattern under close to worst-case Inter-Symbol Interference and cross-talk conditions.

4.2.9 Modified Compliance Pattern in 8b/10b Encoding

The Modified Compliance Pattern consists of the same basic Compliance Pattern sequence (see Section 4.2.8) with one change. Two identical error status Symbols followed by two K28.5 are appended to the basic Compliance sequence of 8b/10b Symbols (K28.5, D21.5, K28.5, and D10.2) to form the Modified Compliance Sequence of (K28.5, D21.5, K28.5, D10.2, error status Symbol, error status Symbol, K28.5, K28.5). The first Modified Compliance Sequence Symbol must have negative disparity. It is permitted to create a disparity error to align the running disparity to the negative disparity of the first Modified Compliance Sequence Symbol. For any given device that has multiple Lanes, every eighth Lane is moved by a total of eight Symbols. Four Symbols of K28.5 occurs at the beginning and another four Symbols of K28.7 occurs at the end of the eight Symbol Modified Compliance Pattern sequence. The first D Symbol has negative disparity to align the delay disparity with the disparity of the Modified Compliance Sequence. After the 16 Symbols are sent, the delay Symbols are advanced to the next Lane, until the delay Symbols have been sent on all eight lanes. Then the delay Symbols cycle back to Lane 0, and the process is repeated. It is permitted to advance the delay sequence across all eight lanes, regardless of the number of lanes detected or supported. A x1 device, or a xN device operating a Link in x1 mode, is permitted to include the Delay symbols with the Modified Compliance Pattern.

An illustration of the Modified Compliance Pattern is shown below:

Lane0	D	D	D	D	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	K28.7-	K28.7-	K28.7-	K28.7-	K28.5-	D21.5
Lane1	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	D	D
Lane2	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	K28.5-	D21.5
Lane3	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	K28.5-	D21.5
Lane4	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	K28.5-	D21.5
Lane5	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	K28.5-	D21.5
Lane6	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	K28.5-	D21.5
Lane7	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	K28.5-	D21.5
Lane8	D	D	D	D	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	K28.7-	K28.7-	K28.7-	K28.7-	K28.5-	D21.5
Lane9	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	K28.5-	D21.5	K28.5+	D10.2	ERR	ERR	K28.5-	K28.5+	D	D

Key:

K28.5-	COM when disparity is negative, specifically: “0011111010”
K28.5+	COM when disparity is positive, specifically: “1100000101”
D21.5	Out of phase data Symbol specifically: “1010101010”
D10.2	Out of phase data Symbol, specifically: “0101010101”
D	Delay Symbol K28.5 (with appropriate disparity)
ERR	error status Symbol (with appropriate disparity)
K28.7-	EIE when disparity is negative, specifically “0011111000”

The reason two identical error Symbols are inserted instead of one is to ensure disparity of the 8b/10b sequence is not impacted by the addition of the error status Symbol.

All other Compliance pattern rules are identical (i.e., the rules for adding delay Symbols) so as to preserve all the crosstalk characteristics of the Compliance Pattern.

The error status Symbol is an 8b/10b data Symbol, maintained on a per-Lane basis, and defined in 8-bit domain in the following way:

- Receiver Error Count (Bits 6:0) - Incremented on every Receiver error after the Pattern Lock bit becomes asserted.
- Pattern Lock (Bit 7) - Asserted when the Lane locks to the incoming Modified Compliance Pattern.

4.2.10 Compliance Pattern in 128b/130b Encoding

The compliance pattern consists of the following repeating sequence of 36 Blocks

1. One block with a Sync Header of 01b followed by a 128-bit unscrambled payload of 64 1's followed by 64 0's
2. One block with a Sync Header of 01b followed by a 128-bit unscrambled payload of the following:

	Lane No modulo 8 = 0	**Lane No modulo 8 = 1**	**Lane No modulo 8 = 2**	**Lane No modulo 8 = 3**	**Lane No modulo 8 = 4**	**Lane No modulo 8 = 5**	**Lane No modulo 8 = 6**	**Lane No modulo 8 = 7**
Symbol 0	55h	FFh	FFh	FFh	55h	FFh	FFh	FFh
Symbol 1	55h	FFh	FFh	FFh	55h	FFh	FFh	FFh
Symbol 2	55h	00h	FFh	FFh	55h	FFh	FFh	FFh
Symbol 3	55h	00h	FFh	FFh	55h	FFh	F0h	F0h
Symbol 4	55h	00h	FFh	C0h	55h	FFh	00h	00h
Symbol 5	55h	00h	C0h	00h	55h	E0h	00h	00h
Symbol 6	55h	00h	00h	00h	55h	00h	00h	00h
Symbol 7	{P,~P}	{P,~P}	{P,~P}	{P,~P}	{P,~P}	{P,~P}	{P,~P}	{P,~P}
Symbol 8	00h	1Eh	2Dh	3Ch	4Bh	5Ah	69h	78h
Symbol 9	00h	55h	00h	00h	00h	55h	00h	F0h
Symbol 10	00h	55h	00h	00h	00h	55h	00h	00h
Symbol 11	00h	55h	00h	00h	00h	55h	00h	00h
Symbol 12	00h	55h	0Fh	0Fh	00h	55h	07h	00h
Symbol 13	00h	55h	FFh	FFh	00h	55h	FFh	00h
Symbol 14	00h	55h	FFh	FFh	7Fh	55h	FFh	00h
Symbol 15	00h	55h	FFh	FFh	FFh	55h	FFh	00h

Key: P: Indicates the 4-bit encoding of the Transmitter preset value being used.
~P: Indicates the bit-wise inverse of P.

3. One block with a Sync Header of 01b followed by a 128-bit unscrambled payload of the following:

	Lane No modulo 8 = 0	**Lane No modulo 8 = 1**	**Lane No modulo 8 = 2**	**Lane No modulo 8 = 3**	**Lane No modulo 8 = 4**	**Lane No modulo 8 = 5**	**Lane No modulo 8 = 6**	**Lane No modulo 8 = 7**
Symbol 0	FFh	FFh	55h	FFh	FFh	FFh	55h	FFh
Symbol 1	FFh	FFh	55h	FFh	FFh	FFh	55h	FFh
Symbol 2	FFh	FFh	55h	FFh	FFh	FFh	55h	FFh
Symbol 3	F0h	F0h	55h	F0h	F0h	F0h	55h	F0h
Symbol 4	00h	00h	55h	00h	00h	00h	55h	00h
Symbol 5	00h	00h	55h	00h	00h	00h	55h	00h
Symbol 6	00h	00h	55h	00h	00h	00h	55h	00h
Symbol 7	{P,~P}	{P,~P}	{P,~P}	{P,~P}	{P,~P}	{P,~P}	{P,~P}	{P,~P}
Symbol 8	00h	1Eh	2Dh	3Ch	4Bh	5Ah	69h	78h
Symbol 9	00h	00h	00h	55h	00h	00h	00h	55h
Symbol 10	00h	00h	00h	55h	00h	00h	00h	55h
Symbol 11	00h	00h	00h	55h	00h	00h	00h	55h
Symbol 12	FFh	0Fh	0Fh	55h	0Fh	0Fh	0Fh	55h
Symbol 13	FFh	FFh	FFh	55h	FFh	FFh	FFh	55h
Symbol 14	FFh	FFh	FFh	55h	FFh	FFh	FFh	55h
Symbol 15	FFh	FFh	FFh	55h	FFh	FFh	FFh	55h

Key: P: Indicates the 4-bit encoding of the Transmitter preset being used.
~P: Indicates the bit-wise inverse of P.

4. One EIEOS Block
5. 32 Data Blocks, each with a payload of 16 IDL data Symbols (00h) scrambled

IMPLEMENTATION NOTE

First Two Blocks of the Compliance Pattern

The first block is a very low frequency pattern to help with measurement of the preset settings. The second block is to notify the Lane number and preset encoding the compliance pattern is using along with ensuring the entire compliance pattern is DC Balanced.

The payload in each Data Block is the output of the scrambler in that Lane (i.e., input data is 0b). The scrambler does not advance during the Sync Header bits. The scrambler is initialized when an EIEOS is transmitted. The Lane numbers used to determine the scrambling LFSR seed value depend on how Polling.Compliance is entered. If it is entered due to the Enter Compliance bit in the Link Control 2 register being set, then the Lane numbers are the numbers that were assigned to the Lanes and the Receiver Lane polarity to be used on each Lane is the Lane polarity inversion that was used in the most recent time that LinkUp was 1b. If a Lane was not part of the configured Link

at that time, and for all other methods of entering Polling.Compliance, the Lane numbers are the default numbers assigned by the Port. These default numbers must be unique. For example, each Lane of a x16 Link must be assigned a unique Lane number between 0 to 15. The Data Blocks of the compliance pattern do not form a Data Stream and hence are exempt from the requirement of transmitting an SDS Ordered Set or EDS Token during Ordered Set Block to Data Block transition and vice-versa.

IMPLEMENTATION NOTE

Ordered Sets in Compliance and Modified Compliance Patterns in 128b/130b Encoding

The various Ordered Sets (e.g., EIEOS and SKP OS) follow the Ordered Set definition corresponding to the current Data Rate of operation. For example, at 16.0 GT/s Data Rate, the EIEOS is the 16.0 GT/s EIEOS whereas at 8.0 GT/s Data Rate, the EIEOS is the 8.0 GT/s EIEOS defined earlier. As defined in Section 4.2.7, the SKP Ordered Set is the Standard SKP Ordered Set.

4.2.11 Modified Compliance Pattern in 128b/130b Encoding

The modified compliance pattern, when not operating in SRIS, consists of repeating the following sequence of 65792 Blocks:

1. One EIEOS Block
2. 256 Data Blocks, each with a payload of 16 Idle data Symbols (00h), scrambled
3. 255 sets of the following sequence:
   i. One SKP Ordered Set
   ii. 256 Data Blocks, each with a payload of 16 Idle data Symbols (00h), scrambled

The modified compliance pattern, when operating in SRIS, consists of repeating the following sequence of 67585 Blocks:

1. One EIEOS Block
2. 2048 sets of the following sequence:
   i. One SKP Ordered Set
   ii. 32 Data Blocks, each with a payload of 16 Idle data Symbols (00h), scrambled

The payload in each Data Block is the output of the scrambler in that Lane (i.e., input data is 0b). The scrambler does not advance during the Sync Header bits. The scrambler is initialized when an EIEOS is transmitted. The Lane numbers used to determine the scrambling LFSR seed value depend on how Polling.Compliance is entered. If it is entered due to the Enter Compliance bit in the Link Control 2 register being set, then the Lane numbers are the numbers that were assigned to the Lanes and the Receiver Lane polarity to be used on each Lane is the Lane polarity inversion used in the most recent time that LinkUp was 1b. If a Lane was not part of the configured Link at that time, and for all other methods of entering Polling.Compliance, the Lane numbers are the default numbers assigned by the Port. These default numbers must be unique. For example, each Lane of a

x16 Link must be assigned a unique Lane number from 0 to 15. The Data Blocks of the modified compliance pattern do not form a Data Stream and hence are exempt from the requirement of transmitting an SDS Ordered Set or EDS Token during Ordered Set Block to Data Block transition and vice-versa.

4.2.12 Jitter Measurement Pattern in 128b/130b

The jitter measurement pattern consists of repeating the following Block:

- Sync Header of 01b followed by a 128-bit unscrambled payload of 16 Symbols of 55h

This generates a pattern of alternating 1s and 0s for measuring the transmitter's jitter.

4.2.13 Lane Margining at Receiver

Lane Margining at Receiver, as defined in this Section, is mandatory for all Ports supporting 16.0 GT/s Data Rate, including Pseudo Ports (Retimers). Lane Margining at Receiver enables system software to obtain the margin information of a given Receiver while the Link is in the L0 state. The margin information includes both voltage and time, in either direction from the current Receiver position. For all Ports that implement Lane Margining at Receiver, Lane Margining at Receiver for timing is required, while support of Lane Margining at Receiver for voltage is optional.

Lane Margining at Receiver begins when a margin command is received, the Link is operating at 16.0 GT/s Data Rate or higher, and the Link is in L0 state. Lane Margining at Receiver ends when either a 'Go to Normal Settings' command is received, the Link changes speed, or the Link exits either the L0 or Recovery states. Lane Margining at Receiver optionally ends when certain error thresholds are exceeded. Lane Margining at Receiver is is permitted to be suspended while the Link is in Recovery for independent samplers.

Lane Margining at Receiver is not supported by PCIe Links operating at 2.5 GT/s, 5.0 GT/s, or 8.0 GT/s.

Software uses the per-Lane Margining Lane Control and Margining Lane Status registers in each Port (Downstream or Upstream) for sending margin commands and obtaining margin status information for the corresponding Receiver associated with the Port. For the Retimers, the commands to get information about the Receiver's capabilities and status and the commands to margin the Receiver are conveyed in the Control SKP Ordered Sets in the Downstream direction. The status and error reporting of the target Retimer Receiver is conveyed in the Control SKP Ordered Sets in the Upstream direction. Software controls margining in the Receiver of a Retimer by writing to the appropriate bits in the Margining Lane Control register in the Downstream Port. The Downstream Port also updates the status information conveyed by the Retimer(s) in the Link through the Control SKP Ordered Set into its Margining Lane Status register.

4.2.13.1 *Receiver Number, Margin Type, Usage Model, and Margin Payload Fields*

The contents of the four command fields of the Margining Lane Control register in the Downstream Port are always reflected in the identical fields in the Downstream Control SKP Ordered Sets. The contents of the Upstream Control SKP Ordered Set received in the Downstream

Port is always reflected in the corresponding status fields of the Margining Lane Status register in the Downstream Port. The following table provides the bit placement of these fields in the Control SKP Ordered Set.

Table 4-18: Margin Command Related Fields in the Control SKP Ordered Set

Symbol	Description where the 'Usage Model' is 'Lane Margining at Receiver'
4*N + 2	Bit 6: Usage Model (0b: Lane Margining at Receiver, 1b: Reserved) Bits [5:3]: Margin Type Bits [2:0]: Receiver Number
4*N + 3	Bits [7:0]: Margin Payload

Usage Model: An encoding of 0b indicates that the usage model is Lane Margining at Receiver. An encoding of 1b in this field is reserved for future usages.

If the 'Usage Model' field is 1b, Bits [5:0] of Symbol 4N+2 and Bits [7:0] of Symbol 4N+3 are Reserved.

When evaluating received Control SKP Ordered Set for Margin Commands, all Receivers that do not comprehend the usage associated with 'Usage Model' = 1b are required to ignore Bits[5:0] of Symbol 4N+2 and Bits[7:0] of Symbol 4N+3 of the Control SKP Ordered Set, if the 'Usage Model' field is 1b.

IMPLEMENTATION NOTE

Potential future usage of Control SKP Ordered Set

The intended usage for the 15 bits of information in the Control SKP Ordered Set, as defined in Table 4-18 is Lane Margining at Receiver. However a single bit (Bit 7 of Symbol 4N+2) is Reserved for any future usage beyond Lane Margining at Receiver. If such a usage is defined in the future, this bit will be set to 1b and the remaining 14 bits can be defined as needed by the new usage model. Alternatively, Symbol 4N could use a different encoding than 78h for any future usage, permitting all bits in Symbols 4N+1, 4N+2, and 4N+3 to be defined for that usage model.

Receiver Number:Receivers are identified in Figure 4-32. The following 'Receiver Number' encodings are used in the Downstream Port for Margin Commands targeting that Downstream Port or a Retimer below that Downstream Port:

000b: Broadcast (Downstream Port Receiver and all Retimer Pseudo Port Receivers)

001b: Rx(A) (Downstream Port Receiver)

010b: Rx(B) (Retimer X or Z Upstream Pseudo Port Receiver)

011b: Rx(C) (Retimer X or Z Downstream Pseudo Port Receiver)

100b: Rx(D) (Retimer Y Upstream Pseudo Port Receiver)

101b: Rx(E) (Retimer Y Downstream Pseudo Port Receiver)

110b: Reserved

111b: Reserved

The following 'Receiver Number' encodings are used in the Upstream Port for Margin Commands targeting that Upstream Port:

000b: Broadcast (Upstream Port Receiver)

001b: Reserved

010b: Reserved

011b: Reserved

100b: Reserved

101b: Reserved

110b: Rx (F) (Upstream Port Receiver)

111b: Reserved

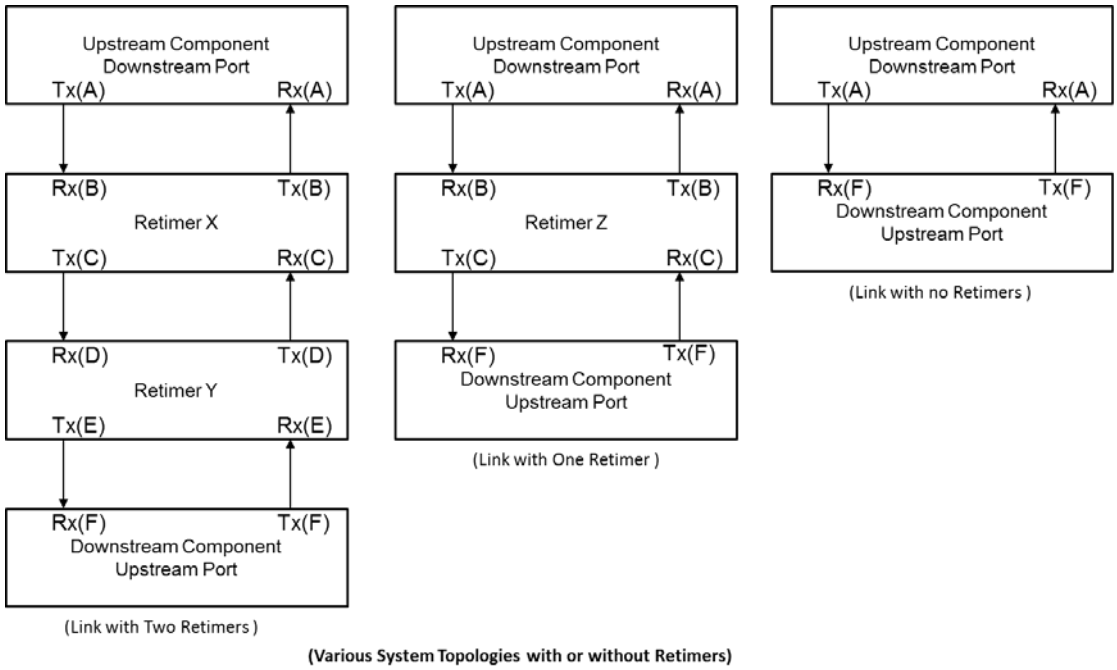


Figure 4-32: Receiver Number Assignment

Margin Type and Margin Payload: The 'Margin Type' field together with a valid 'Receiver Number'(s), associated with the 'Margin Type' encoding, and specific 'Margin Payload' field define various commands used for margining (referred to as 'margin command'). Table 4-19 defines the encodings of valid margin commands along with the corresponding responses, used in both the Control SKP Ordered Sets as well as the Margining Lane Control and Margining Lane Status registers. Margin commands that are always broadcast will use the broadcast encoding for the Receiver Number, even when only one Receiver is the target (e.g., USP or a DSP in a Link with no Retimers). The Receiver Number field in the response to a margin command other than 'No Command' reflects the number of the Receiver that is responding, even for a margin command that is broadcast. The margin commands go Downstream whereas the responses go Upstream in the

Control SKP Ordered Sets. The responses reflect the 'Margin Type' to which the target Receiver is responding. The Receiver Number field of the response corresponds to the target Receiver that is responding. The various parameters such as $M_{SampleCount}$ used here are defined in Section 8.4.4. All the unused encodings described below are 'Reserved' and must not considered to be a valid margin command.

Table 4-19: Margin Commands and Corresponding Responses

Command				**Response**	
Margin Command	**Margin Type [2:0]**	**Valid Receiver Number(s) [2:0]**	**Margin Payload [7:0]**	**Margin Type [2:0]**	**Margin Payload [7:0]**
No Command	111b	000b	9Ch ('No Command' is also an independent command in Upstream direction. The expected Response is 'No Command' with the Receiver Number = 000b.)		
Access Retimer register (Optional)	001b	010b, 100b	Register offset in bytes: 00h – 87h, A0h – FFh	001b	Register value, if supported. Target Receiver on Retimer returns 00h if it does not support accessing its registers.
Report Margin Control Capabilities	001b	001b through 110b	88h	001b	Margin Payload[7:5] = Reserved; Margin Payload[4:0] = { $M_{IndErrorSampler}$, $M_{SampleReportingMethod}$, $M_{IndLeftRightTiming}$, $M_{IndUpDownVoltage}$, $M_{VoltageSupported}$ }
Report $M_{NumVoltageSteps}$	001b	001b through 110b	89h	001b	Margin Payload [7] = Reserved Margin Payload[6:0] = $M_{NumVoltageSteps}$
Report $M_{NumTimingSteps}$	001b	001b through 110b	8Ah	001b	Margin Payload [7:6] = Reserved Margin Payload [5:0] = $M_{NumTimingSteps}$
Report $M_{MaxTimingOffset}$	001b	001b through 110b	8Bh	001b	Margin Payload [7] = Reserved Margin Payload[6:0] = $M_{MaxTimingOffset}$
Report $M_{MaxVoltageOffset}$	001b	001b through 110b	8Ch	001b	Margin Payload [7] = Reserved Margin Payload[6:0] = $M_{MaxVoltageOffset}$
Report $M_{SamplingRateVoltage}$	001b	001b through 110b	8Dh	001b	Margin Payload [7:6] = Reserved Margin Payload[5:0] = { $M_{SamplingRateVoltage}$ [5:0]}
Report $M_{SamplingRateTiming}$	001b	001b through 110b	8Eh	001b	Margin Payload [7:6] = Reserved Margin Payload[5:0] = { $M_{SamplingRateTiming}$ [5:0]}

Command				Response	
Margin Command	**Margin Type [2:0]**	**Valid Receiver Number(s) [2:0]**	**Margin Payload [7:0]**	**Margin Type [2:0]**	**Margin Payload [7:0]**
Report $M_{SampleCount}$	001b	001b through 110b	8Fh	001b	Margin Payload [7] = Reserved Margin Payload[6:0] = $M_{SampleCount}$
Report $M_{MaxLanes}$	001b	001b through 110b	90h	001b	Margin Payload [7:5] = Reserved Margin Payload[4:0] = $M_{MaxLanes}$
Report Reserved	001b	001b through 110b	91-9Fh	001b	Margin Payload[7:0] = Reserved
Set Error Count Limit	010b	001b through 110b	Margin Payload [7:6] = 11b Margin Payload[5:0] = Error Count Limit	010b	Margin Payload [7:6] = 11b Margin Payload[5:0] = Error Count Limit registered by the target Receiver
Go to Normal Settings	010b	000b through 110b	0Fh	010b	0Fh
Clear Error Log	010b	000b through 110b	55h	010b	55h
Step Margin to timing offset to right/left of default	011b	001b through 110b	See Section 4.2.13.1.2	011b	Margin Payload[7:6] = Step Margin Execution Status (see Section 4.2.13.1.1) Margin Payload[5:0] = $M_{ErrorCount}$
Step Margin to voltage offset to up/down of default	100b	001b through 110b	See Section 4.2.13.1.2	100b	Margin Payload[7:6] = Step Margin Execution Status (see Section 4.2.13.1.1) Margin Payload[5:0] = $M_{ErrorCount}$
Vendor Defined	101b	001b through 110b	Vendor Defined	101b	Vendor Defined

4.2.13.1.1 Step Margin Execution Status

The Step Margin Execution Status used in Table 4-19 is a 2-bit field defined as follows:

- 11b: NAK. Indicates that an unsupported Lane Margining command was issued. For example,timing margin beyond +/- 0.2 UI. $M_{ErrorCount}$ is 0.

- 10b: Margining in progress. The Receiver is executing the step margin command. $M_{ErrorCount}$ reflects the number of errors detected as defined in Section 8.4.4.
- 01b: Set up for margin in progress. This indicates the Receiver is getting ready but has not yet started executing the step margin command. $M_{ErrorCount}$ is 0.
- 00b: Too many errors – Receiver autonomously went back to its default settings. $M_{ErrorCount}$ reflects the number of errors detected as defined in Section 8.4.4. Note that $M_{ErrorCount}$ might be greater than Error Count Limit.

4.2.13.1.2 Margin Payload for Step Margin Commands

For the 'Step Margin to timing offset to right/left of default' command, the Margin Payload field is defined as follows:

- Margin Payload [7]: Reserved.
- If $M_{IndLeftRightTiming}$ for the targeted Receiver is Set:
  - Margin Payload [6] indicates whether the margin command is right vs left. A 0b indicates to move the Receiver to the right of the normal setting whereas a 1b indicates to move the Receiver to the left of the normal setting.
  - Margin Payload [5:0] indicates the number of steps to the left or right of the normal setting.
- If $M_{IndLeftRightTiming}$ for the targeted Receiver is Clear:
  - Margin Payload [6]: Reserved
  - Margin Payload [5:0] indicates the number of steps beyond the normal setting.

For the 'Step Margin to voltage offset to up/down of default' command, the Margin Payload field is defined as follows:

- If $M_{IndUpDownVoltage}$ for the targeted Receiver is Set:
  - Margin Payload [7] indicates whether the margin command is up vs down. A 0b indicates to move the Receiver up from the normal setting whereas a 1b indicates to move the Receiver down from the normal setting.
  - Margin Payload [6:0] indicates the number of steps up or down from the normal setting.
- If $M_{IndUpDownVoltage}$ for the targeted Receiver is Clear:
  - Margin Payload [7]: Reserved
  - Margin Payload [6:0] indicates the number of steps beyond the normal setting.

4.2.13.2 *Margin Command and Response Flow*

Each Receiver advertises its capabilities as defined in Section 8.4.4. The Receiver being margined must report the number of errors that are consistent with data samples occurring at the indicated location for margining. For simplicity, the margin commands and requirements are described in terms of moving the data sampler location though the actual margining method may be implementation specific. For example, the timing margin could be implemented on the actual data

sampler or an independent/error sampler. Further, the timing margin can be implemented by injecting an appropriate amount of stress/jitter to the data sample location, or by actually moving the data/error sample location. When an independent data/error sampler is used, the errors encountered with the independent data/error sampler must be reported in $M_{ErrorCount}$ even though the Link may not experience any errors. To margin a Receiver, Software moves the target Receiver to a voltage/timing offset from its default sampling position.

The following rules must be followed:

- Every Retimer Upstream Pseudo Port Receiver and the Downstream Port Receiver must compute the 'Margin CRC' and 'Margin Parity' bits and compare against the received 'Margin CRC' and 'Margin Parity' bits. Any mismatch must result in ignoring the contents of Symbols 4N+2 and 4N+3. A Downstream Port Receiver must report 'Margin CRC' and 'Margin Parity' errors in the Lane Error Status Register (see Section 7.7.3.3).
- The Upstream Port Receiver is permitted to ignore the 'Margin CRC' bits, 'Margin Parity' bits, and all bits in the Symbols 4N+2 and 4N+3 of the Control SKP Ordered Set. If it checks 'Margin CRC' and 'Margin Parity', any mismatch must be reported in the Lane Error Status Register.
- The Downstream Port must transmit Control SKP Ordered Sets in each Lane, with the 'Margin Type', 'Receiver Number', 'Usage Model', and 'Margin Payload' fields reflecting the corresponding control fields in the Margining Lane Control register. Any Control SKP Ordered Set transmitted more than 10 µsec after the Configuration Write Completion must reflect the Margining Lane Control values written by that Configuration Write.
    - This requirement applies regardless of the values in the Margining Lane Control register.
    - This requirement applies regardless of the number of Retimer(s) in the Link.
- For Control SKP Ordered Sets received by the Upstream Pseudo Port, a Retimer Receiver is the target of a valid margin command, if all of the following conditions are true:
    - the 'Margin Type' is not 'No Command'
    - the 'Receiver Number' is the number assigned to the Receiver, or 'Margin Type' is either 'Clear Error Log' or 'Go to Normal Settings' and the 'Receiver Number' is 'Broadcast'.
    - the 'Usage Model' field is 0b
    - the 'Margin Type', 'Receiver Number', and 'Margin Payload' fields are consistent with the definitions in Table 4-18
    - the CRC check and Margin Parity check pass.
- For Upstream and Downstream Ports, a Receiver is the target of a valid margin command, if all of the following conditions are true for its Margining Lane Control register:
    - the 'Margin Type' is not 'No Command'
    - the 'Receiver Number' is the number assigned to the Receiver or 'Margin Type' is either 'Clear Error Log' or 'Go to Normal Settings' and the 'Receiver Number' is 'Broadcast'
    - the 'Usage Model' field is 0b
    - the 'Margin Type', the 'Receiver Number', and 'Margin Payload' fields are consistent with the definitions in Table 4-19

- The Upstream Port must transmit the Control SKP Ordered Set with 'No Command'.
- A target Receiver must apply and respond to the margin command within 1ms of receiving the valid margin command if the Link is still in L0 state and operating at 16.0 GT/s Data Rate.
  - A target Receiver in a Retimer must send a response in the Control SKP Ordered Set in the Upstream Direction within 1 ms of receiving the margin command.
  - A target Receiver in the Upstream Port must update the Status field of the Lane Margin Command and Status register within 1 ms of receiving the margin command.
  - A target Receiver in the Downstream Port must update the Status field of the Lane Margin Command and Status register within 1 ms of receiving the margin command if the command is not broadcast or no Retimer(s) are present
- For a valid 'Margin Type', other than 'No Command', that is broadcast and received by a Retimer:
  - A Retimer, in position X (see Figure 4-32), forwards the response unmodified in the Upstream Control SKP Ordered Set, if the command has been applied, else it sends the 'No Command'.
  - The Receiver Number field of the response must be set to an encoding of one of the Retimer's Pseudo Ports.
  - The Retimer must respond only after both Pseudo Ports have completed the Margin Command.
- The Retimer must overwrite Bits [4:0] of Symbol 4N+1, Bits[7, 5:0] of Symbol 4N+2 and Bits [7:0] in Symbol 4N+3 as it forwards the Control SKP Ordered Set in the Upstream direction if it is the target Receiver of a margin command and is executing the command.
- On receipt of a Control SKP Ordered Set, the Downstream Port must reflect the Margining Lane Status register from the corresponding fields in the received Control SKP Ordered Set within 1 µsec, if it passes the 'Margin CRC' and 'Margin Parity' checks and one of the following conditions apply:
  - In the Margining Lane Control register: 'Receiver Number' is 010b through 101b
  - In the Margining Lane Control register: 'Receiver Number' is 000b, 'Margin Command' is 'Clear Error Log', 'No Command', or 'Go to Normal Settings', and there are Retimer(s) in the Link
  - Optionally, if the Margining Lane Control register 'Usage Model' field is 1b
  - Optionally, if the Margining Lane Control register 'Receiver Number' field is 110b or 111b

  The Margining Lane Status register fields are updated regardless of the Usage Model bit in the received Control SKP Ordered Set.
- A component must advertise the same value for each parameter defined in Table 8-12 in Section 8.4.4 across all its Receivers. A component must not change any parameter value except for $M_{SampleCount}$ and $M_{ErrorCount}$ defined in Table 8-12 in Section 8.4.4 while LinkUp = 1b.
- A target Receiver that receives a valid 'Step Margin' command must continue to apply that offset until any of the following occur:

- it receives a valid 'Go to Normal Settings' command
- it receives a subsequent valid 'Step Margin' command with different 'Margin Type' or 'Margin Payload' field
- $M_{IndErrorSampler}$ is 0b and $M_{ErrorCount}$ exceeds 'Error Count Limit'
- Optionally, $M_{IndErrorSampler}$ is 1b and $M_{ErrorCount}$ exceeds 'Error Count Limit'.

- If a 'Step Margin' command terminates because $M_{ErrorCount}$ exceeds 'Error Count Limit', the target Receiver must automatically return to its default sample position and indicate this in the 'Margin Payload' field ('Step Margin Execution Status' = 00b). Note: termination for this reason is optional if $M_{IndErrorSampler}$ is 1b.
- If $M_{IndErrorSampler}$ is 0b, an error is detected when:
  - The target Receiver is a Port that enters Recovery or detects a Data Parity mismatch while in L0
  - The target Receiver is a Pseudo Port that enters Forwarding training sets or detects a Data Parity mismatch while forwarding non-training sets.
- If $M_{IndErrorSampler}$ is 1b, an error is detected when:
  - The target Receiver is a Port and a bit error is detected while in L0
  - The target Receiver is a Pseudo Port and a bit error is detected while the Retimer is forwarding non-training sets
- If $M_{IndErrorSampler}$ is 0b and either (1) the target Receiver is a Port that enters Recovery or (2) the target Receiver is a Pseudo Port that enters Forwarding training sets:
  - The target Receiver must go back to the default sample position
  - If the target Receiver is a Port that is still performing margining, it must resume the margin position within 128 µsec of entering L0
  - If the target Receiver is a Pseudo Port that is still performing margining, it must resume the margin position within 128 µsec of Forwarding non-training sets
- A target Receiver is required to clear its accumulated error count on receiving 'Clear Error Log' command, while it continues to margin (if it is the target Receiver of a 'Step Margin' command still in progress), if it was doing so.
- For a target Receiver of a 'Set Error Count Limit' command, the new value is used for all future 'Step Margin' commands until a new 'Set Error Count Limit' command is received.
- If no 'Set Error Count Limit' is received by a Receiver since entering L0, the default value is 4.
- Behavior is undefined if a 'Set Error Count Limit' command is received while a 'Step Margin' command is in effect.
- Once a target Receiver reports a 'Step Margin Execution Status' of 11b (NAK) or 00b ('Too many errors'), it must continue to report the same status as long as the 'Step Margin' command is in effect.
- A target Receiver must not report a 'Step Margin Execution Status' of 01b ('Set up for margin in progress') for more than 100 ms after it receives a new valid 'Step Margin' command

- A target Receiver that reports a 'Step Margin Execution Status' other than 01b, cannot report 01b subsequently unless it receives a new valid 'Step Margin' command.
- Reserved bits in the Margin Payload must follow these rules:
  - The Downstream or Upstream Port must transmit 0s for Reserved bits
  - The retimer must forward Reserved bits unmodified
  - All Receivers must ignore Reserved bits
- Reserved encodings of the 'Margin Command', 'Receiver Number', or 'Margin Payload' fields must follow these rules:
  - The retimer must forward Reserved encodings unmodified
  - All Receivers must treat Reserved encodings as if they are not the target of the margin command
- A Vendor Defined margin command or response, that is not defined by a retimer is ignored and forwarded normally.
- A target Receiver on a Retimer must return 00h on the response payload on 'Access Retimer register' command, if it does not support register access. If a Retimer supports 'Access Retimer register' command, the following must be observed:
  - It must return a non-zero value for the DWORD at locations 80h and 84h respectively.
  - It must not place any registers corresponding to 'Margin Payload' locations 88h through 9Fh.

4.2.13.3 *Receiver Margin Testing Requirements*

Software must ensure that the following conditions are met before performing Lane Margining at Receiver:

- The current Link data rate must be 16.0 GT/s.
- The current Link width must include the Lanes that are to be tested.
- The Upstream Port's Function(s) must be programmed to a D-state that prevents the Port from entering the L1 Link state. See Section 5.2 for more information.
- The ASPM Control field of the Link Control register must be set to 00b (Disabled) in both the Downstream Port and Upstream Port.
- The Hardware Autonomous Speed Disable bit of the Link Control 2 register must be set to 1b in both the Downstream Port and Upstream Port.
- The Hardware Autonomous Width Disable bit of the Link Control register must be set to 1b in both the Downstream Port and Upstream Port.

While margining, software must ensure the following:

- All margin commands must have the 'Usage Model' field in the Margining Lane Control register set to 0b. While checking for the status of an outstanding margin command, software must check that the 'Usage Model' field of the status part of the Margining Lane Status register is set to 0b.

- Software must read the capabilities offered by a Receiver and margin it within the constraints of the capabilities it offers. The commands issued and the process followed to determine the margin must be consistent with the definitions provided in Sections 4.2.13 and 8.4.4. For example, if the Port does not support voltage testing, then software must not initiate a voltage test. In addition, if a Port supports testing of 2 Lanes simultaneously, then software must test only 1 or 2 Lanes at the same time and not more than 2 Lanes.
- For Receivers where $M_{IndErrorSampler}$ is 1b, any combination of such Receivers are permitted to be margined in parallel.
- For Receivers where $M_{IndErrorSampler}$ is 0b, at most one such Receiver is permitted to be margined at a time. However, margining may be performed on multiple Lanes simultaneously, as long as it is within the maximum number of Lanes the device supports.
- Software must ensure that the margin command it provides in the Margining Lane Control register is a valid one, as defined in Section 4.2.13.1. For example, the 'Margin Type' must have a defined encoding and the 'Receiver Number' and 'Margin Payload' consistent with it.
- After issuing a command by writing to the Margining Lane Control register atomically, software must check for the completion of this command. This is done by atomically reading the Margining Lane Status register and checking that the status fields match the expected response for the issued command (see Table 4-18). If 10 ms has elapsed after a new margin command was issued and the values read do not match the expected response, software is permitted to assume that the Receiver will not respond, and declare that the target Receiver failed margining. For a broadcast command other than 'No Command' the Receiver Number in the response must correspond to one of the Pseudo Ports in Retimer Y or Retimer Z, as described in Figure 4-32.
- Any two reads of the Margining Lane Status register should be spaced at least 10 µsec apart to make sure they are reading results from different Control SKP Ordered Sets.
- Software must broadcast 'No Command' and wait for it to complete prior to issuing a new 'Margin Type' or 'Receiver Number' or 'Margin Payload' in the Margining Lane Control register.
- At the end of margining in a given direction (voltage/ timing and up/down/left/right), software must broadcast 'Go to Normal Settings', 'No Command', 'Clear Error Log', and 'No Command' in series in the Downstream and Upstream Ports, after ensuring each command has been acknowledged by the target Receiver.
- If the Data Rate has changed during margining, margining results (if any) are not accurate and software must exit the margining procedure. Software must set the Margining Lane Control register to 'No Command' to avoid starting margining if the Data Rate later changes to 16.0 GT/s or higher.
- Software is permitted to issue a 'Clear Error Log' command periodically while margining is in progress, to gather error information over a long period of time.
- Software must not attempt to margin both timing and voltage of a target Receiver simultaneously. Results are undefined if a Receiver receives commands that would place both voltage and timing margin locations away from the default sample position at the same time.
- Software should allow margining to run for at least 10^8 bits margined by the Receiver under test before switching to the next margin step location (unless the error limit is exceeded).

- Software must account for the 'set up for margin in progress' status while measuring the margin time or the number of bits sampled by the Receiver.
- If a target Receiver is reporting 'set up for margin in progress' for 200 ms after issuing one of the 'Step Margin to Timing (or Voltage)' commands, Software is permitted to assume that the Receiver will not respond and declare that the target Receiver failed margining.
- If a Receiver reports a 'NAK' in the 'Margin Payload' status field and the corresponding 'Step Margin' command was valid and within the allowable range (as defined in Sections 4.2.13 and 8.4.4), Software is permitted to declare that the target Receiver failed margining.

IMPLEMENTATION NOTE

Example Software Flow for Lane Margining at Receiver

For getting the invariant parameters the following steps may be followed. Once obtained, the same parameters can be used across multiple sets of margining tests as long as LinkUp=1b continues to be true. For each component in the Link, do the following Steps. Software can do these steps in parallel for different components on different Lanes of the Link.

Step 1: Issue 'Report Margin Control Capabilities' ('Margin Type' = 001b, 'Margin Payload' = 88h, 'Receiver Number' = target device in the Margining Lane Control register)

Step 2: Read the Margining Lane Status register.

(a) If 'Margin Type' = 001b and 'Receiver Number' = target Receiver: Go to Step 3
(b) Else: If 10 ms has expired since command issued, declare Receiver failed margining and exit; else wait for >10 µsec and Go to Step 2
Step 3: Store the information provided 'Margin Payload' status field for use during margining.

Step 4: Broadcast 'No Command' ('Margin Type' = 111b, 'Receiver Number' = 000b, and 'Margin Payload' = 9Ch in the Margining Lane Control register) and wait for those to be reflected back in the Margining Lane Status register. If 10 ms expires without getting the command completion handshake, declare the Receiver failed margining and exit.

Step 5: Repeat Steps 1-4 for 'Report $M_{NumVoltageSteps}$', 'Report $M_{NumTimingSteps}$', 'Report $M_{MaxTimingOffset}$', 'Report $M_{MaxVoltageOffset}$', 'Report $M_{SamplingRateVoltage}$', and 'Report $M_{SamplingRateTiming}$'. It may be noted that this step can be executed in parallel across different Lanes for different Margin Type.

Margining on each Lane across the Link can be a sequence of separate commands. Prior to launching the sequence, software should read the maximum number of Lanes it is allowed to run margining simultaneously. The steps would be similar to Steps 1-4 above with the 'Report $M_{MaxLanes}$' command. After that software can simultaneously margin up to that many Lanes of the Link. On each Link, each Receiver is margined based on its capability, subject to the constraints described here, after ensuring the Link is operating at full width in 16.0 GT/s Data Rate and the hardware autonomous width and speed change as well as ASPM power states have been disabled.

If software desires to set an Error Count Limit value different than default of 4 or whatever was programmed last, it executes the following Steps prior to going to Step 1 below.

Step 1: Issue 'Set Error Count Limit' ('Margin Type' = 010b, the target 'Receiver Number', and 'Margin Payload' = {11b, Error Count Limit} in the Margining Lane Control register)

Step 2: Read the Margining Lane Status register.

(a) If 'Margin Type' = 010b, 'Receiver Number' = target Receiver, and 'Margin Payload' = 'Margin Payload' control field (Bits [14:7]), go to Step 4
(b) Else: If 10 ms has expired since command issued, go to Step 3; else wait for >10 µsec and Go to Step 2
Step 3: Margining has failed. Invoke the system checks to find out if the Link degraded in width/speed due to reliability reasons.

Step 4: Broadcast 'No Command' and wait for those to be reflected back in the status fields. If 10 ms expires without getting the command completion handshake, declare the Receiver failed margining and exit.

The following steps is an example flow of one margin point for a given Receiver executing 'Step Margin to timing offset to right of default' starting with 15 steps:

Step 1: Write 'Margin Type' = 011b, 'Receiver Number' = target Receiver, and 'Margin Payload' = {0000b, 1111b} in the Margining Lane Control register

Step 2: Read the Margining Lane Status register.

(a) If 'Margin Type' = 011b and 'Receiver Number' = target Receiver, Go to Step 3

(b) Else If 10 ms has expired since command issued, declare Receiver has failed margining and go to Step 7

(c) Wait for >10 µsec and Go to Step 2

Step 3: (In the Margining Lane Status register) If 'Margin Payload' [7:6] = 11b: If we exceeded the 0.2 UI, that is the margin; Else report margin failure at this point and go to Step 7;

Else if 'Margin Payload' [7:6] = 00b: report margin failure at this point and go to Step 7

Else if 'Margin Payload' [7:6] = 01b:

(a) If 200 ms has elapsed since entering Step 3, report that the Receiver failed margining test and exit; else wait 1 ms, read the Margining Lane Status register and go to Step 3

Else go to Step 4

Step 4: Wait for the desired amount of time for margining to happen while sampling the Margining Lane Status register periodically for the number of errors reported in the 'Margin Payload' field (Bits [5:0] – $M_{ErrorCount}$). For longer runs, issue the 'Clear Error Log' command, (using similar to Steps 1-4 for 'Set Error Count Limit', with the corresponding expected status fields, preceded by issuing the 'No Command' in Step 4 in the 'Set Error Count Limit' example) if the length of time will cause the error count to exceed the 'Set Error Count Limit' even when staying within the expected BER target. If the aggregate error count remains within the expected error count and the 'Margin Payload [7:6]' in the status field remains 10b till the end, the Receiver has the required Margin at the timing margin step; else it fails that timing margin step go to Step 7.

Step 5: Broadcast 'No Command' and wait for those to be reflected back in the status fields. If 10 ms expires without getting the command completion handshake, declare the Receiver failed margining and exit.

Step 6: Go to Step 1, incrementing the number of timing steps through the 'Margin Payload' control field (Bits[5:0]) if we want to test against a higher margin amount; else go to Step 8 noting the margin value that the Receiver passed

Step 7: Margin failed; The previous margin step the Receiver passed in Step 6 is the margin of the Receiver

Step 8: Broadcast 'No Command', 'Clear Error Log', 'No Command', 'Go to Normal Settings' series of commands (using similar to Steps 1-4 for 'Set Error Count Limit' with the corresponding expected status fields)

4.3 Retimers

This Section defines the requirements for Retimers that are Physical Layer protocol aware and that interoperate with any pair of Components with any compliant channel on each side of the Retimer. An important capability of a Physical Layer protocol aware Retimer is to execute the Phase 2/3 of the equalization procedure in each direction. A maximum of two Retimers are permitted between an Upstream and a Downstream Port.

The two Retimer limit is based on multiple considerations, most notably limits on modifying SKP Ordered Sets and limits on the time spent in Phase 2/3 of the equalization procedure. To ensure interoperability, platform designers must ensure that the two Retimer limit is honored for all PCI Express Links, including those involving form factors as well as those involving active cables. Form factor specifications may define additional Retimer rules that must be honored for their form factors. Assessing interoperability with any Extension Device not based on the Retimer definition in this section is outside the scope of this specification.

Many architectures of Extension Devices are possible, i.e., analog only Repeater, protocol unaware Retimer, etc. This specification describes a Physical Layer protocol aware Retimer. It may be possible to use other types of Extension Devices in closed systems if proper analysis is done for the specific channel, Extension Device, and end-device pair – but a specific method for carrying out this analysis is outside the scope of this specification.

Retimers have two Pseudo Ports, one facing Upstream, and the other facing Downstream. The Transmitter of each Pseudo Port must derive its clock from a 100 MHz reference clock. The reference clock(s) must meet the requirements of Section 8.6. A Retimer supports one or more reference clocking architectures as defined in Section 8.6 Electrical Sub-block.

In most operations Retimers simply forward received Ordered Sets, DLLPs, TLPs, Logical Idle, and Electrical Idle. Retimers are completely transparent to the Data Link Layer and Transaction Layer. System software shall not enable L0s on any Link where a Retimer is present. Support of beacon by Retimers is optional and beyond the scope of this specification.

When using 128b/130b encoding the Retimer executes the protocol so that each Link Segment undergoes independent Link equalization as described in Section 4.3.6.

The Pseudo Port orientation (Upstream or Downstream), is determined dynamically, while the Link partners are in Configuration. Both crosslink and regular Links are supported.

4.3.1 Retimer Requirements

The following is a high level summary of Retimer requirements:

- Retimers are required to comply with all the electrical specification described in Chapter 8 Electrical Sub-block. Retimers must operate in one of two modes:
  - Retimers' Receivers operate at 8.0 GT/s and above with an impedance that meets the range defined by the $Z_{RX\text{-}DC}$ parameter for 2.5 GT/s.
  - Retimers' Receivers operate at 8.0 GT/s and above with an impedance that does not meet the range defined by the $Z_{RX\text{-}DC}$ parameter for 2.5 GT/s. In this mode the $Z_{RX\text{-}DC}$

parameter for 2.5 GT/s must be met with in 1 ms of receiving an EIOS or inferring Electrical Idle and while the Receivers remain in Electrical Idle.

- ❑ Forwarded Symbols must always be de-skewed when more than one Lane is forwarding Symbols (including upconfigure cases).
- ❑ Determine Port orientation dynamically.
- ❑ Perform Lane polarity inversion (if needed).
- ❑ Execute the Link equalization procedure for Phase 2 and Phase 3, when using 128b/130b encoding, on each Link Segment.
- ❑ Interoperate with de-emphasis negotiation at 5.0 GT/s, on each Link Segment.
- ❑ Interoperate with Link Upconfigure
- ❑ Pass loopback data between the loopback master and loopback slave.
    - Optionally execute Slave Loopback on one Pseudo Port.
- ❑ Generate the Compliance Pattern on each Pseudo Port.
    - Load board method (i.e., time out in Polling.Active).
    - ❑ Forward Modified Compliance Pattern when the Link enters Polling.Compliance via Compliance Receive bit in TS1 Ordered Sets.
    - ❑ Forward Compliance or Modified Compliance Patterns when Ports enter Polling.Compliance via the Enter Compliance bit in the Link Control 2 register is set to 1b in both the Upstream Port and the Downstream Port and Retimer Enter Compliance is set to 1b (accessed in an implementation specific manner) in the Retimer.
- ❑ Adjust the data rate of operation in concert with the Upstream and Downstream Ports of the Link.
- ❑ Adjust the Link width in concert with the Upstream and Downstream Ports of the Link.
- ❑ Capture Lane numbers during Configuration.
    - Lane numbers are required when using 128b/130b encoding for the scrambling seed.
- ❑ Dynamically adjust Retimer Receiver impedance to match end Component Receiver impedance.
- ❑ Infer entering Electrical Idle at all data rates.
- ❑ Modify certain fields of Ordered Sets while forwarding.
- ❑ Perform clock compensation via addition or removal of SKP Symbols.
- ❑ Support L1.
    - Optionally Support L1 PM Substates.

4.3.2 Supported Topologies

Figure 4-33 shows the topologies supported by Retimers defined in this specification. There may be one or two Retimers between the Upstream and Downstream Ports on a Link. Each Retimer has

two Pseudo Ports, which determine their Downstream/Upstream orientation dynamically. Each Retimer has an Upstream Path and a Downstream Path. Both Pseudo Ports must always operate at the same data rate, when in Forwarding mode. Thus each Path will also be at the same data rate. A Retimer is permitted to support any width option defined by this specification as its maximum width. The behavior of the Retimer in each high level operating mode is:

- Forwarding mode:
  - Symbols, Electrical Idle, and exit from Electrical Idle; are forwarded on each Upstream and Downstream Path.
- Execution mode:
  - The Upstream Pseudo Port acts as an Upstream Port of a Component. The Downstream Pseudo Port acts as a Downstream Port of a Component. This mode is used in the following cases:
    - Polling.Compliance.
    - Phase 2 and Phase 3 of the Link equalization procedure.
    - Optionally Slave Loopback.

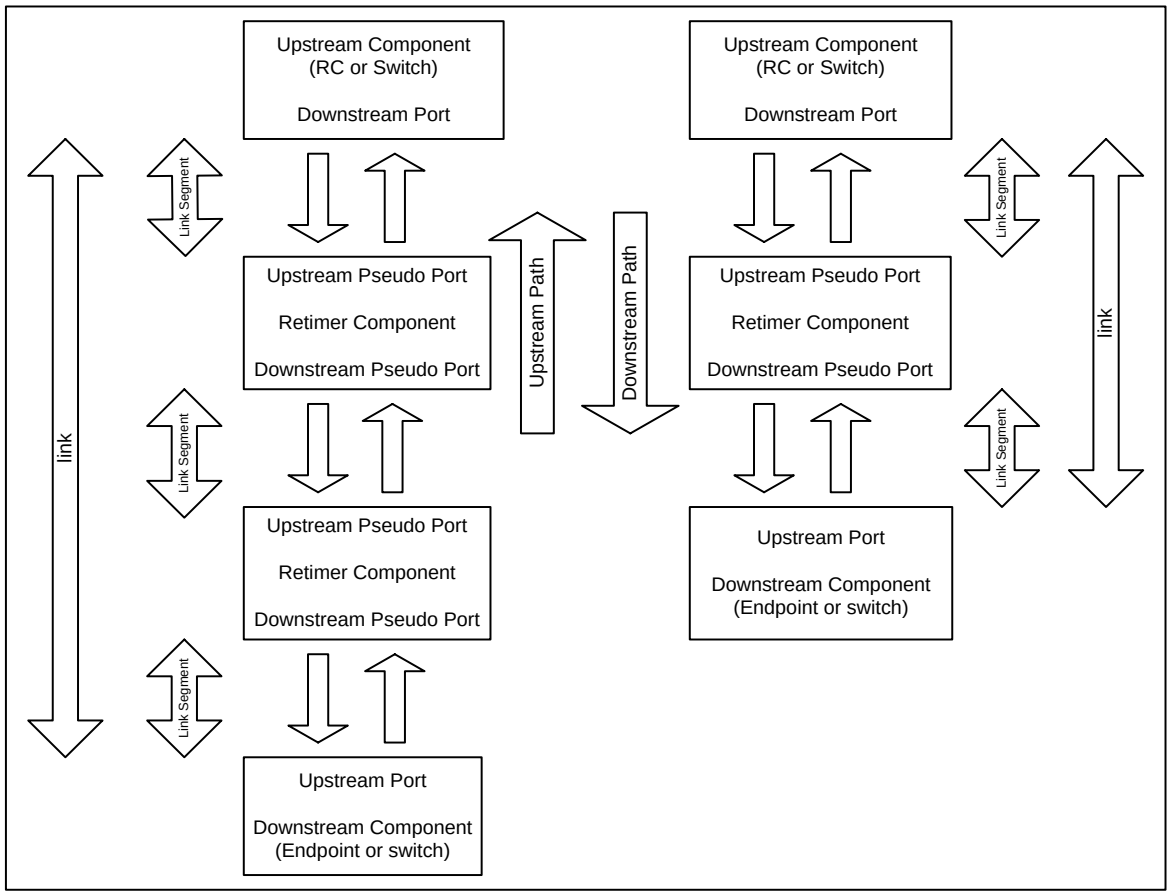


Figure 4-33: Supported Topologies

4.3.3 Variables

The following variables are set to the following specified values following a Fundamental Reset or whenever the Retimer receives Link and Lane number equal to PAD on two consecutive TS2 Ordered Sets on all Lanes that are receiving TS2 Ordered Sets on both Upstream and Downstream Pseudo Ports within a 1 μs time window from the last Symbol of the second TS2 Ordered Set on the first Lane to the last Symbol of the second TS2 Ordered Set on the last Lane.

- RT_port_orientation = undefined
- RT_captured_lane_number = PAD
- RT_captured_link_number = PAD
- RT_G3_EQ_complete = 0b
- RT_G4_EQ_complete = 0b
- RT_LinkUp = 0b
- RT_next_data_rate = 2.5 GT/s
- RT_error_data_rate = 2.5 GT/s

4.3.4 Receiver Impedance Propagation Rules

The Retimer Transmitters and Receivers shall meet the requirements in Section 4.2.4.8.1 while Fundamental Reset is asserted. When Fundamental Reset is deasserted the Retimer is permitted to take up to 20 ms to begin active determination of its Receiver impedance. During this interval the Receiver impedance remains as required during Fundamental Reset. Once this interval has expired Receiver impedance on Retimer Lanes is determined as follows:

- Within 1.0 ms of the Upstream or Downstream Port's Receiver meeting the $Z_{RX\text{-}DC}$ parameter, the low impedance is back propagated, (i.e., the Retimer's Receiver shall meet the $Z_{RX\text{-}DC}$ parameter on the corresponding Lane on the other Pseudo Port). Each Lane operates independently and this requirement applies at all times.
- The Retimer must keep its Transmitter in Electrical Idle until the $Z_{RX\text{-}DC}$ state has been detected. This applies on an individual Lane basis.

4.3.5 Switching Between Modes

The Retimer operates in two basic modes, Forwarding mode or Execution mode. When switching between these modes the switch must occur on an Ordered Set boundary for all Lanes of the Transmitter at the same time. No other Symbols shall be between the last Ordered Set transmitted in the current mode and the first Symbol transmitted in the new mode.

When using 128b/130b the Transmitter must maintain the correct scrambling seed and LFSR value when switching between modes.

When switching between Forwarding and Execution modes, the Retimer must ensure that at least 16 TS1 Ordered Sets and at most 64 TS1 Ordered Sets are transmitted between the last EIEOS transmitted in the previous mode and the first EIEOS transmitted in the new mode.

When switching to and from the Execution Link Equalization mode the Retimer must ensure a Transmitter does not send two SKP Ordered Sets in a row, and that the maximum allowed interval is not exceeded between SKP Ordered Sets, see Section 4.2.7.3.

4.3.6 Forwarding Rules

These rules apply when the Retimer is in Forwarding mode. The Retimer is in Forwarding mode after the deassertion of Fundamental Reset.

- If the Retimer's Receiver detects an exit from Electrical Idle on a Lane the Retimer must enter Forwarding mode and forward the Symbols on that Lane to the opposite Pseudo Port as described in Section 4.3.6.3.
- The Retimer must continue to forward the received Symbols on a given Lane until it enters Execution mode or until an EIOS is received, or until Electrical Idle is inferred on that Lane. This requirement applies even if the Receiver loses Symbol lock or Block Alignment. See Section 4.3.6.5 for rules regarding Electrical Idle entry.
- A Retimer shall forward all Symbols unchanged, except as described in Section 4.3.6.9 and 4.3.6.7.
- When operating at 2.5 GT/s data rate, if any Lane of a Pseudo Port receives TS1 Ordered Sets with Link and Lane numbers set to PAD for 5 ms or longer, and the other Pseudo Port does not detect an exit from Electrical Idle on any Lane in that same window, and either of the following occurs:
    - The following sequence occurs:
        - An EIOS is received on any Lane that was receiving TS1 Ordered Sets
        - followed by a period of Electrical Idle, for less than 5 ms
        - followed by Electrical Idle Exit that cannot be forwarded according to section 4.3.6.3
        - Note: this is interpreted as the Port attached to the Receiver going into Electrical Idle followed by a data rate change for a Compliance Pattern above 2.5 GT/s.
    - Compliance Pattern at 2.5 GT/s is received on any Lane that was receiving TS1 Ordered Sets.

    Then the Retimer enters the Execution mode CompLoadBoard state, and follows Section 4.3.7.1.
- If any Lane on the Upstream Pseudo Port receives two consecutive TS1 Ordered Sets with the EC field equal to 10b, when using 128b/130b encoding, then the Retimer enters Execution mode Equalization, and follows Section 4.3.7.2.
- If the Retimer is configured to support Execution mode Slave Loopback and if any Lane on either Pseudo Port receives two consecutive TS1 Ordered Sets or two consecutive TS2 Ordered Sets with the Loopback bit set to 1b then the Retimer enters Execution mode Slave Loopback, and follows Section 4.3.7.3.

4.3.6.1 *Forwarding Type Rules*

A Retimer must determine what type of Symbols it is forwarding. The rules for inferring Electrical Idle are a function of the type of Symbols the Retimer is forwarding. If a Path forwards two consecutive TS1 Ordered Sets or two consecutive TS2 Ordered Sets, on any Lane, then the Path is forwarding training sets. If a Path forwards eight consecutive Symbol Times of Idle data on all Lanes that are forwarding Symbols then the Path is forwarding non-training sets. When a Retimer transitions from forwarding training sets to forwarding non-training sets, the variable RT_error_data_rate is set to 2.5 GT/s.

4.3.6.2 *Orientation and Lane Numbers Rules*

The Retimer must determine the Port orientation, Lane assignment, and Lane polarity dynamically as the Link trains.

- When RT_LinkUp=0, the first Pseudo Port to receive two consecutive TS1 Ordered Sets with a non-PAD Lane number on any Lane, has its RT_port_orientation variable set to Upstream Port, and the other Pseudo Port has its RT_port_orientation variable set to Downstream Port.
- The Retimer plays no active part of Lane number determination. The Retimer must capture the Lane numbers with the RT_captured_lane_number variable at the end of the Configuration state, between the Link Components. This applies on the first time through Configuration, i.e., when RT_LinkUp is set to 0b. Subsequent trips through Configuration during Link width configure must not change the Lane numbers. Lane numbers are required for the scrambling seed when using 128b/130b. Link numbers are required in some cases when the Retimer is in Execution mode. Link numbers and Lane numbers are captured with the RT_captured_lane_number, and RT_captured_link_number variables whenever the first two consecutive TS2 Ordered Sets that contain non-PAD Lane and non-PAD Link numbers are received after RT_LinkUp variable is set to 0b. A Retimer must function normally if Lane reversal occurs. When the Retimer has captured the Lane numbers and Link numbers the variable RT_LinkUp is set to 1b. In addition if the Disable Scrambling bit in the TS2 Ordered Sets is set to 1b, in either case above, then the Retimer determines that scrambling is disabled when using 8b/10b encoding.
- Lane polarity is determined any time the Lane exits Electrical Idle, and achieves Symbol lock at 2.5 GT/s as described in Section 4.2.4.4:
    - If polarity inversion is determined the Receiver must invert the received data. The Transmitter must never invert the transmitted data.

4.3.6.3 *Electrical Idle Exit Rules*

At data rates other than 2.5 GT/s, EIEOS are sent within the training sets to ensure that the analog circuit detects an exit from Electrical Idle. Receiving an EIEOS is required when using 128b/130b encoding to achieve Block Alignment. When the Retimer starts forwarding data after detecting an Electrical Idle exit, the Retimer starts transmitting on a training set boundary. The first training sets it forwards must be an EIEOS, when operating at data rates higher than 2.5 GT/s. The first EIEOS sent will be in place of the TS1 or TS2 Ordered Set that it would otherwise forward.

If no Lanes meet $Z_{RX\text{-}DC}$ on a Pseudo Port, and the following sequence occurs:

- An exit from Electrical Idle is detected on any Lane of that Pseudo Port.
- And then if not all Lanes infer Electrical Idle, via absence of exit from Electrical Idle in a 12 ms window on that Pseu do Port and the other Pseudo Port is not receiving Ordered Sets on any Lane in that same 12 ms window.

Then the same Pseudo Port, where no Lanes meet $Z_{RX\text{-}DC}$, sends the Electrical Idle Exit pattern described below for 5 µs on all Lanes.

If operating at 2.5 GT/s and the following occurs:

- any Lane detects an exit from Electrical Idle
- and then receives two consecutive TS1 Ordered Sets with Lane and Link numbers equal to PAD
- and the other Pseudo Port is not receiving Ordered Sets on any Lane

Then Receiver Detection is performed on all Lanes of the Pseudo Port that is not receiving Ordered Sets. If no Receivers were detected then:

- The result is back propagated as described in Section 4.3.4, within 1.0 ms.
- The same Pseudo Port that received the TS1 Ordered Sets with Lane and Link numbers equal to PAD, sends the Electrical Idle Exit pattern described below for 5 µ s on all Lanes.

If a Lane detects an exit from Electrical Idle then the Lane must start forwarding when all of the following are true:

- Data rate is determined, see Section 4.3.6.4, current data rate is changed to RT_next_data_rate if required.
- Lane polarity is determined, see Section 4.3.6.2.
- Two consecutive TS1 Ordered Sets or two consecutive TS2 Ordered Sets are received.
- Two consecutive TS1 Ordered Sets or two consecutive TS2 Ordered Sets are received on all Lanes that detected an exit from Electrical Idle or the max Retimer Exit Latency has occurred, see Table 4-20.
- Lane De-skew is achieved on all Lanes that received two consecutive TS1 or two consecutive TS2 Ordered Sets.
- <u>If a data rate change has occurred then 6 µs has elasped since Electrical Idle Exit was detected.</u>

All Ordered Sets used to establish forwarding must be discarded. Only Lanes that have detected a Receiver on the other Pseudo Port, as described in Section 4.3.4, are considered for forwarding.

Otherwise after a 3.0 ms timeout, if the other Pseudo Port is not receiving Ordered Sets then Receiver Detection is performed on all Lanes of the Pseudo Port that is not receiving Ordered Sets, the result is back propagated as described in Section 4.3.4, and if no Receivers were detected:

- Then the same Pseudo Port that was unable to receive two consecutive TS1 or TS2 Ordered Sets on any Lane sends the Electrical Idle Exit pattern described below for 5 µs on all Lanes.
- Else the Electrical Idle Exit pattern described below is forwarded on all Lanes that detected an exit from Electrical Idle.
- When using 128b/130b encoding:

- One EIEOS
- 32 Data Blocks, each with a payload of 16 Idle data Symbols (00h), scrambled, for Symbols 0 to 13.
- Symbol 14 and 15 of each Data Block either contain Idle data Symbols (00h), scrambled, or DC Balance, determined by applying the same rules in Section 4.2.4.1 to these Data Blocks.

- When using 8b/10b encoding:
  - The Modified Compliance Pattern with the error status Symbol set to 00h.
- This Path now is forwarding the Electrical Idle Exit pattern. In this state Electrical Idle is inferred by the absence of Electrical Idle Exit, See Table 4-21. The Path continues forwarding the Electrical Idle Exit pattern until Electrical Idle is inferred on any lane, or a 48 ms time out occurs. If a 48 ms time out occurs then:
  - RT_LINK_UP is set to 0b
  - The Pseudo Port places its Transmitter in Electrical Idle
  - The RT_next_data_rate and the RT_error_data_rate must be set to 2.5 GT/s for both Pseudo Ports
  - Receiver Detect is performed on the Pseudo Port that was sending the Electrical Idle Exit pattern and timed out, the result is back propagated as described in section 4.3.4.

The Transmitter, on the opposite Pseudo Port that was sending the Electrical Idle Exit Pattern and timed out, sends the Electrical Idle Exit Pattern described above for 5 µs.

IMPLEMENTATION NOTE

Electrical Idle Exit

Forwarding Electrical Idle Exit occurs in error cases where a Retimer is unable to decode training sets. Upstream and Downstream Ports use Electrical Idle Exit (without decoding any Symbols) during Polling.Compliance, and Recovery.Speed. If the Retimer does not forward Electrical Idle Exit then the Upstream and Downstream Ports will misbehave in certain conditions. For example, this may occur after a speed change to a higher data rate. In this event forwarding Electrical Idle Exit is required to keep the Upstream and Downstream Ports in lock step at Recovery.Speed, so that the data rate will return to the previous data rate, rather than a Link Down condition from a time out to Detect.

When a Retimer detects an exit from Electrical Idle and starts forwarding data, the time this takes is called the Retimer Exit Latency, and allows for such things as data rate change (if required), clock and data recovery, Symbol lock, Block Alignment, Lane-to-Lane de-skew, Receiver tuning, etc. The maximum Retimer Exit Latency is specified below for several conditions:

- The data rate before and after Electrical Idle and Electrical Idle exit detect does not change.
- Data rate change to a data rate that uses 8b/10b encoding.
- Data rate change to a data rate that uses 128b/130b encoding for the first time.

- Data rate change to a data rate that uses 128b/130b encoding not for the first time.
- How long both transmitters have been in Electrical Idle when a data rate change occurs.

Retimers are permitted to change their data rate while in Electrical Idle, and it is recommended that Retimers start the data rate change while in Electrical Idle to minimize Retimer Exit latency.

Table 4-20: Maximum Retimer Exit Latency

Condition	Link in EI For X µs, where, X < 500 µs	Link in EI for For X ≥ 500 µs
No data rate change	4 µs	4 µs
When forwarding TS1 Ordered Sets at 2.5 GT/s with Lane and Link number equal to PAD.	1 ms	1 ms
Any data rate change to 8b/10b encoding data rate	500 – X µ s	4 µs
First data rate change to 128b/130b encoding date rate	1.5 -X ms	1 ms
Subsequent data rate change to 128b/130b encoding date rate	500 –X µs	4 µs

4.3.6.4 *Data Rate Change and Determination Rules*

The data rate of the Retimer is set to 2.5 GT/s after deassertion of Fundamental Reset.

Both Pseudo Ports of the Retimer must operate at the same data rate. If a Pseudo Port places its Transmitter in Electrical Idle, then the Symbols that it has just completed transmitting determine the variables RT_next_data_rate and RT_error_data_rate. Only when both Pseudo Ports have all Lanes in Electrical Idle shall the Retimer change the data rate. If both Pseudo Ports do not make the same determination of these variables then both variables must be set to 2.5 GT/s.

- If both Pseudo Ports were forwarding non-training sequences, then the RT_next_data_rate must be set to the current data rate. The RT_error_data_rate must be set to 2.5 GT/s. Note: this covers the case where the Link has entered L1 from L0.
- If both Pseudo Ports were forwarding TS2 Ordered Sets with the speed_change bit set to 1b and either:
  - the data rate, when forwarding those TS2s, is greater than 2.5 GT/s or,
  - the highest common data rate received in the data rate identifiers in both directions is greater than 2.5 GT/s,

  then RT_next_data_rate must be set to the highest common data rate and the RT_error_data_rate is set to current data rate. Note: this covers the case where the Link has entered Recovery.Speed from Recovery.RcvrCfg and is changing the data rate according to the highest common data rate.

- Else the RT_next_data_rate must be set to the RT_error_data_rate. The RT_error_data_rate is set to 2.5 GT/s. Note this covers the two error cases:
  - This indicates that the Link was unable to operate at the current data rate (greater than 2.5 GT/s) and the Link will operate at the 2.5 GT/s data rate or,
  - This indicates that the Link was unable to operate at the new negotiated data rate and will revert back to the old data rate with which it entered Recovery from L0 or L1.

4.3.6.5 *Electrical Idle Entry Rules*

The Rules for Electrical Idle entry in Forwarding mode are a function of whether the Retimer is forwarding training sets or non-training sets. The determination of this is described in Section 4.3.6.1.

Before a Transmitter enters Electrical Idle, it must always send the Electrical Idle Ordered Set Sequence (EIOSQ), unless otherwise specified.

If the Retimer is forwarding training sets then:

- If an EIOS is received on a Lane, then the EIOSQ is forwarded on that Lane and only that Lane places its Transmitter in Electrical Idle.
- If Electrical Idle is inferred on a Lane, then that Lane places its Transmitter in Electrical Idle, after EIOSQ is transmitted on that Lane.

Else if the Retimer is forwarding non-training sets then:

- If an EIOS is received on any Lane, then the EIOSQ is forwarded on all Lanes that are currently forwarding Symbols and all Lanes place their Transmitters in Electrical Idle.
- If Electrical Idle is inferred on a Lane, then that Lane places its Transmitter in Electrical Idle, and EIOSQ is not transmitted on that Lane.

The Retimer is required to infer Electrical Idle. The criteria for a Retimer inferring Electrical Idle are described in Table 4-21.

Table 4-21: Inferring Electrical Idle

State	2.5 GT/s	5.0 GT/s	8.0 GT/s	16.0 GT/s
Forwarding: Non Training Sequence	Absence of a SKP Ordered Set in a 128 µs window	Absence of a SKP Ordered Set in a 128 µs window	Absence of a SKP Ordered Set in a 128 µs window	Absence of a SKP Ordered Set in a 128 µs window
Forwarding: Training Sequence	Absence of a TS1 or TS2 Ordered Set in a 1280 UI interval	Absence of a TS1 or TS2 Ordered Set in a 1280 UI interval	Absence of a TS1 or TS2 Ordered Set in a 4680 UI interval	Absence of a TS1 or TS2 Ordered Set in a 4680 UI interval
Forwarding: Electrical Idle Exit Executing: Force Timeout	Absence of an exit from Electrical Idle in a 2000 UI interval	Absence of an exit from Electrical Idle in a 16000 UI interval	Absence of an exit from Electrical Idle in a 16000 UI interval	Absence of an exit from Electrical Idle in a 16000 UI interval

State	2.5 GT/s	5.0 GT/s	8.0 GT/s	16.0 GT/s
Forwarding: Loopback Executing: Loopback Slave	Absence of an exit from Electrical Idle in a 128 µs window	N/A	N/A	N/A

4.3.6.6 *Transmitter Settings Determination Rules*

When a data rate change to 16.0 GT/s occurs the Retimer transmitter settings are determined as follows:

- If the RT_G4_EQ_complete variable is set to 1b:
  - The Transmitter must use the coefficient settings agreed upon at the conclusion of the last equalization procedure applicable to 16.0 GT/s operation.
- Else:
  - An Upstream Pseudo Port must use the 16.0 GT/s Transmitter preset values it registered from the received eight consecutive 8GT EQ TS2 Ordered Sets in its Transmitter preset setting as soon as it starts transmitting at the 16.0 GT/s data rate and must ensure that it meets the preset definition in Section 8.3.3.3. Lanes that received a Reserved or unsupported Transmitter preset value must use an implementation specific method to choose a supported Transmitter preset setting for use as soon it starts transmitting at 16.0 GT/s.
  - A Downstream Pseudo Port determines its Transmitter Settings in an implementation specific manner when it starts transmitting at 16.0 GT/s.

The RT_G4_EQ_complete variable is set to 1b when:

- Two consecutive TS1 Ordered Sets are received with EC = 01b at 16.0 GT/s.

The RT_G4_EQ_complete variable is set to 0b when any of the following occur:

- RT_LinkUp variable is set to 0b.
- Eight consecutive 8GT EQ TS2 Ordered Sets are received on any Lane of the Upstream Pseudo Port. The value in the 16.0 GT/s Transmitter Preset field is registered for later use at 16.0 GT/s for that Lane.

When a data rate change to 8.0 GT/s occurs the Retimer transmitter settings are determined as follows:

- If the RT_G3_EQ_complete variable is set to 1b:
  - The Transmitter must use the coefficient settings agreed upon at the conclusion of the last equalization procedure applicable to 8.0 GT/s operation.
- Else:
  - An Upstream Pseudo Port must use the 8.0 GT/s Transmitter preset values it registered from the received eight consecutive EQ TS2 Ordered Sets in its Transmitter preset setting as soon as it starts transmitting at the 8.0 GT/s data rate and must ensure that it

meets the preset definition in Section 8.3.3. Lanes that received a Reserved or unsupported Transmitter preset value must use an implementation specific method to choose a supported Transmitter preset setting for use as soon it starts transmitting at 8.0 GT/s. The Upstream Pseudo Port may optionally use the 8.0 GT/s Receiver preset hint values it registered in those EQ TS2 Ordered Sets.

- A Downstream Pseudo Port determines its Transmitter preset settings in an implementation specific manner when it starts transmitting at 8.0 GT/s.

The RT_G3_EQ_complete variable is set to 1b when:

- Two consecutive TS1 Ordered Sets are received with EC = 01b at 8.0 GT/s.

The RT_G3_EQ_complete variable is set to 0b when any of the following occur:

- RT_LinkUp variable is set to 0b
- Eight consecutive EQ TS1 or eight consecutive EQ TS2 Ordered Sets are received on any Lane of the Upstream Pseudo Port. The value in the 8.0 GT/s Transmitter Preset and optionally the 8.0 GT/s Receiver Preset Hint fields are registered for later use at 8.0 GT/s for that Lane.

When a data rate change to 5.0 GT/s occurs the Retimer transmitter settings are determined as follows:

- The Upstream Pseudo Port must sets its Transmitters to either -3.5 dB or -6.0 dB, according to the Selectable De-emphasis bit (bit 6 of Symbol 4) received in eight consecutive TS2 Ordered Sets, in the most recent series of TS2 Ordered sets, received prior to entering Electrical Idle.
- The Downstream Pseudo Port sets its Transmitters to either -3.5 dB or -6.0 dB in an implementation specific manner.

4.3.6.7 *Ordered Set Modification Rules*

Ordered Sets are forwarded, and certain fields are modified according to the following rules:

- The Retimer shall not modify any fields except those specifically allowed/required for modification in this specification.
- LF: the Retimer shall overwrite the LF field in TS1 Ordered Sets transmitted in both directions. The new value is determined in an implementation specific manner by the Retimer.
- FS: the Retimer shall overwrite the FS field in TS1 Ordered Sets transmitted in both directions. The new value is determined in an implementation specific manner by the Retimer.
- Pre-Cursor Coefficient: the Retimer shall overwrite the Pre-Cursor Coefficient field in TS1 Ordered Sets transmitted in both directions. The new value is determined by the current Transmitter settings.
- Cursor Coefficient: the Retimer shall overwrite the Cursor Coefficient field in TS1 Ordered Sets transmitted in both directions. The new value is determined by the current Transmitter settings.
- Post-Cursor Coefficient: the Retimer shall overwrite the Post-Cursor Coefficient field in the TS1 Ordered Sets transmitted in both directions. The new value is determined by the current Transmitter settings.

- Parity: the Retimer shall overwrite the Parity bit of forwarded TS1 Ordered Sets. This bit is the even parity of all bits of Symbols 6, 7, and 8 and bits 6:0 of Symbol 9.
- Transmitter Preset: the Retimer shall overwrite the Transmitter Preset field in TS1 Ordered Sets transmitted in both directions. If the Transmitter is using a Transmitter preset setting then the value is equal to the current setting, else it is recommended that the Transmitter Preset field be set to the most recent Transmitter preset setting that was used for the current data rate.

  The Retimer is permitted to do the following:

  - overwrite the Transmitter Preset in EQ TS1 Ordered Sets in either direction
  - overwrite the 8.0 GT/s Transmitter Preset field in EQ TS2 Ordered Sets in the Downstream direction.
  - overwrite the 16.0 GT/s Transmitter Preset field in 8GT EQ TS2 Ordered Sets, in the Downstream direction.

  The new values for the 8.0 GT/s Transmitter Preset and 16.0 GT/s Transmitter Preset fields are determined in an implementation specific manner by the Retimer.

  During phase 0 of Equalization to 16.0 GT/s (i.e., the current Data Rate is 8.0 GT/s) the Retimer is permitted to do the following in the Upstream direction:

  - Forward received TS2 Ordered Sets.
  - Convert TS2 Ordered Sets to 8GT EQ TS2 Ordered Sets, the value for the 16.0 GT/s Transmitter Preset field is determined in an implementation specific manner by the Retimer.
  - Forward received 8GT EQ TS2 Ordered Sets with modification, the value for the 16.0 GT/s Transmitter Preset field is determined in an implementation specific manner by the Retimer.
  - Convert 8GT EQ TS2 Ordered Sets to TS2 Ordered Sets.
- Receiver Preset Hint: the Retimer is permitted to do the following:
  - overwrite the Receiver Preset Hint in EQ TS1 Ordered Sets in either direction
  - overwrite the 8.0 GT/s Receiver Preset Hint field in EQ TS2 Ordered Sets in the Downstream direction.

  The new values, for the Receiver Preset Hint and 8.0 GT/s Receiver Preset Hint fields are determined in an implementation specific manner by the Retimer.
- SKP Ordered Set: The Retimer is permitted to adjust the length of SKP Ordered Sets transmitted in both directions. The Retimer must perform the same adjustment on all Lanes. When operating with 8b/10b encoding, the Retimer is permitted to add or remove one SKP Symbol of a SKP Ordered Set. When operating with 128b/130b encoding, a Retimer is permitted to add or remove 4 SKP Symbols of a SKP Ordered Set.
- Control SKP Ordered Set: The Retimer must modify the First Retimer Data Parity, or the Second Retimer Data Parity, of the Control SKP Ordered Set when the Retimer is in forwarding mode at 16.0 GT/s, according to its received parity. The received even parity is computed independently on each Lane as follows:

- Parity is initialized when a data rate change occurs.
- Parity is initialized when a SDS Ordered Set is received.
- Parity is updated with each bit of a Data Block's payload before de-scrambling has been performed.
- Parity is initialized when a Control SKP Ordered Set is received. However, parity is NOT initialized when a Standard SKP Ordered Set is received.

If a Pseudo Port detects the Retimer Present bit was 0b in the most recently received two consecutive TS2 or EQ TS2 Ordered Sets received by that Pseudo Port when operating at 2.5 GT/s then that Pseudo Port receiver modifies the First Retimer Data Parity as it forwards the Control SKP Ordered Set, else that Pseudo Port receiver modifies the Second Retimer Data Parity as it forwards the Control SKP Ordered Set.

The Retimer must modify symbols 4*N+1, 4*N+2, and 4*N+3 of the Control SKP Ordered Set in the Upstream direction as described in Section 4.2.13.

See Section 4.2.7.2 for Control SKP Ordered Set definition.

- Selectable De-emphasis: the Retimer is permitted to overwrite the Selectable De-emphasis field in the TS1 or TS2 Ordered Set in both directions. The new value is determined in an implementation specific manner by the Retimer.
- The Data Rate Identifier: The Retimer must set the Data Rate Supported bits of the Data Rate Identifier Symbol consistent with the data rates advertised in the received Ordered Sets and its own max supported Data Rate, i.e., it clears to 0b all Symbol 4 bits[5:0] Data Rates that it does not support. A Retimer must support all data rates below and including its maximum supported data rate. A Retimer makes its determination of maximum supported Data Rate once, after fundamental reset.
- DC Balance: When operating with 128b/130b encoding, the Retimer tracks the DC Balance of its Pseudo Port transmitters and transmits DC Balance Symbols as described in Section 4.2.4.1.
- Retimer Present: When operating at 2.5 GT/s, the Retimer must set the Retimer Present bit of all forwarded TS2 and EQ TS2 Ordered Sets to 1b.
- Two Retimers Present: If the Retimer supports 16.0 GT/s, then when operating at 2.5 GT/s, the Retimer must set the Two Retimers Present bit of all forwarded TS2 and EQ TS2 Ordered Sets if it receives a TS2 or EQ TS2 Ordered Set with the Retimer Present bit set to 1b. If the Retimer does not support 16.0 GT/s, then when operating at 2.5 GT/s, the Retimer is permitted to set the Two Retimers Present bit of all forwarded TS2 and EQ TS2s Ordered Sets if it receives a TS2 or EQ TS2 Ordered Sets with the Retimer Present bit set to 1b.
- Loopback: When optionally supporting Slave Loopback in Execution mode, the Loopback bit must be cleared to 0b when forwarding training sets.

4.3.6.8 *DLLP, TLP, and Logical Idle Modification Rules*

DLLPs, TLPs, and Logical Idle are forwarded with no modifications to any of the Symbols unless otherwise specified.

4.3.6.9 *8b/10b Encoding Rules*

The Retimer shall meet the requirements in Section 4.2.1.1.3 except as follows:

- When the Retimer is forwarding and an 8b/10b decode error or a disparity error is detected in the received data, the Symbol with an error is replaced with the D21.3 Symbol with incorrect disparity in the forwarded data.
- This clause in Section 4.2.1.1.3 does not apply: If a received Symbol is found in the column corresponding to the incorrect running disparity or if the Symbol does not correspond to either column, the Physical Layer must notify the Data Link Layer that the received Symbol is invalid. This is a Receiver Error, and is a reported error associated with the Port (see Section 6.2).

IMPLEMENTATION NOTE

Retimer Transmitter Disparity

The Retimer must modify certain fields of the TS1 and TS2 Ordered Sets (e.g., Receiver Preset Hint, Transmitter Preset), therefore the Retimer must recalculate the running disparity. Simply using the disparity of the received Symbol may lead to an error in the running disparity. For example some 8b/10b codes have 6 ones and 4 zeros for positive disparity, while other codes have 5 ones and 5 zeros.

4.3.6.10 *8b/10b Scrambling Rules*

A Retimer is required to determine if scrambling is disabled when using 8b/10b encoding as described in Section 4.3.6.2.

4.3.6.11 *Hot Reset Rules*

If any Lane of the Upstream Pseudo Port receives two consecutive TS1 Ordered Sets with the Hot Reset bit set to 1b and both the Disable Link and Loopback bits set to 0b, and then both Pseudo Ports either receive an EIOS or infer Electrical Idle on any Lane, that is receiving TS1 Ordered Sets, the Retimer does the following:

- Clears variable RT_LinkUp = 0b.
- Places its Transmitters in Electrical Idle on both Pseudo Ports.
- Set the RT_next_data_rate variable to 2.5 GT/s.
- Set the RT_error_data_rate variable to 2.5 GT/s.
- Waits for an exit from Electrical Idle on every Lane on both Pseudo Ports.

The Retimer does not perform Receiver detection on either Pseudo Port.

4.3.6.12 *Disable Link Rules*

If any Lane of the Upstream Pseudo Port receives two consecutive TS1 Ordered Sets with the Disable Link bit set to 1b and both the Hot Reset and Loopback bits set to 0b, and then both

Pseudo Ports either receive an EIOS or infer Electrical Idle on any Lane, that is receiving TS1 Ordered Sets, the Retimer does the following:

- Clears variable RT_LinkUp = 0b.
- Places its Transmitters in Electrical Idle on both Pseudo Ports.
- Set the RT_next_data_rate variable to 2.5 GT/s.
- Set the RT_error_data_rate variable to 2.5 GT/s.
- Waits for an exit from Electrical Idle on any Lane on either Pseudo Port.

The Retimer does not perform Receiver detection on either Pseudo Port.

4.3.6.13 *Loopback*

The Retimer follows these additional rules if any Lane receives two consecutive TS1 Ordered Sets with the Loopback bit equal to 1b and both the Hot Reset and Disable Link bits set to 0b and the ability to execute Slave Loopback is not configured in an implementation specific way. The purpose of these rules is to allow interoperation when a Retimer (or two Retimers) exist between a Loopback master and a Loopback slave.

- The Pseudo Port that received the TS1 Ordered Sets with the Loopback bit set to 1b acts as the Loopback Slave (the other Pseudo Port acts as Loopback Master). The Upstream Path is defined as the Pseudo Port that is the Loopback master to the Pseudo Port that is the Loopback slave. The other Path is the Downstream Path.
- Once established, if a Lane loses the ability to maintain Symbol Lock or Block alignment, then the Lane must continue to transmit Symbols while in this state.
- When using 8b/10b encoding and Symbol lock is lost, the Retimer must attempt to re-achieve Symbol Lock.
- When using 128b/130b encoding and Block Alignment is lost, the Retimer must attempt to re-achieve Block Alignment via SKP Ordered Sets.
- If Loopback was entered while the Link Components were in Configuration.Linkwidth.Start, then determine the highest common data rate of the data rates supported by the Link via the data rates received in two consecutive TS1 Ordered Sets or two consecutive TS2 Ordered Sets on any Lane, that was receiving TS1 or TS2 Ordered Sets, at the time the transition to Forwarding.Loopback occurred. If the current data rate is not the highest common data rate, then:
    - Wait for any Lane to receive EIOS, and then place the Transmitters in Electrical Idle for that Path.
    - When all Transmitters are in Electrical Idle, adjust the data rate as previously determined.
    - If the new data rate is 5.0 GT/s, then the Selectable De-emphasis is determined the same as way as described in Section 4.2.6.10.1.
    - If the new data rate uses 128b/130b encoding, then the Transmitter preset setting is determined the same as way as described in Section 4.2.6.10.1.

- In the Downstream Path; wait for Electrical Idle exit to be detected on each Lane and then start forwarding when two consecutive TS1 Ordered Sets have been received, on a Lane by Lane basis. This is considered the first time to this data rate for the Retimer exit latency.
- In the Upstream Path; if the Compliance Receive bit of the TS1 Ordered Sets that directed the slave to this state was not asserted, then wait for Electrical Idle exit to be detected on each Lane, and start forwarding when two consecutive TS1 Ordered Sets have been received, on a Lane by Lane basis. This is considered the first time to this data rate for the Retimer exit latency.

- In the Upstream Path; if the Compliance Receive bit of the TS1 Ordered Sets that directed the slave to this state was set to 1b, then wait for Electrical Idle exit to be detected on each Lane, and start forwarding immediately, on a Lane by Lane basis. This is considered the first time to this data rate for the Retimer exit latency.
- If four EIOS (one EIOS if the current data rate is 2.5 GT/s) are received on any Lane then:
  - Transmit eight EIOS on every Lane that is transmitting TS1 Ordered Sets on the Pseudo Port that did not receive the EIOS and place the Transmitters in Electrical Idle.
- When both Pseudo Ports have placed their Transmitters in Electrical Idle then:
  - Set the RT_next_data_rate variable to 2.5 GT/s.
  - Set the RT_error_data_rate variable to 2.5 GT/s.
  - The additional rules for Loopback no longer apply unless the rules for entering this Section are met again.

4.3.6.14 *Compliance Receive Rules*

The Retimer follows these additional rules if any Lane receives eight consecutive TS1 Ordered Sets (or their complement) with the Compliance Receive bit set to 1b and the Loopback bit set to 0b. The purpose of the following rules is to support Link operation with a Retimer when the Compliance Receive bit is Set and the Loopback bit is Clear in TS1 Ordered Sets, transmitted by the Upstream or Downstream Port, while the Link is in Polling.Active.

- Pseudo Port A is defined as the first Pseudo Port that receives eight consecutive TS1 Ordered Sets (or their complement) with the Compliance Receive bit is Set and the Loopback bit is Clear. Pseudo Port B is defined as the other Pseudo Port.
- The Retimer determines the highest common data rate of the Link by examining the data rate identifiers in the TS1 Ordered Sets received on each Pseudo Port, and the max data rate supported by the Retimer.
- If the highest common data rate is equal to 5.0 GT/s then:
  - The Retimer must change its data rate to 5.0 GT/s as described in Section 4.3.6.4.
  - The Retimer Pseudo Port A must set its de-emphasis according to the selectable de-emphasis bit received in the eight consecutive TS1 Ordered Sets.
  - The Retimer Pseudo Port B must set its de-emphasis in an implementation specific manner.

- If the highest common data rate is equal to 8.0 GT/s or higher then:
  - The Retimer must change its data rate to as applicable, as described in Section 4.3.6.4.
  - Lane numbers are determined as described in Section 4.2.11.
  - The Retimer Pseudo Port A must set its Transmitter coefficients on each Lane to the Transmitter preset value advertised in Symbol 6 of the eight consecutive TS1 Ordered Sets and this value must be used by the Transmitter (use of the Receiver preset hint value advertised in those TS1 Ordered Sets is optional). If the common data rate is 8.0 GT/s or higher, any Lanes that did not receive eight consecutive TS1 Ordered Sets with Transmitter preset information can use any supported Transmitter preset setting in an implementation specific manner.
  - The Retimer Pseudo Port B must set its Transmitter and Receiver equalization in an implementation specific manner.
- The Retimer must forward the Modified Compliance Pattern when it has locked to the pattern. This occurs independently on each Lane in each direction. If a Lane's Receiver loses Symbol Lock or Block Alignment, the associated Transmitter (i.e., same Lane on opposite Pseudo Port) Continues to forward data.
- Once locked to the pattern, the Retimer keeps an internal count of received Symbol errors, on a per-Lane basis. The pattern lock and Lane error is permitted to be readable in an implementation specific manner, on a per-Lane basis.
- When operating with 128b/130b encoding, Symbols with errors are forwarded unmodified by default, or may optionally be corrected to remove error pollution. The default behavior must be supported and the method of selecting the optional behavior, if supported, is implementation specific.
- When operating with 8b/10b encoding, Symbols with errors are replaced with the D21.3 Symbol with incorrect disparity by default, or may optionally be corrected to remove error pollution. The default behavior must be supported and the method of selecting the optional behavior, if supported, is implementation specific.
- The error status Symbol when using 8b/10b encoding or the Error_Status field when using 128b/130b encoding is forwarded unmodified by default, or may optionally be redefined as it is transmitted by the Retimer. The default behavior must be supported and the method of selecting the optional behavior, if supported, is implementation specific.
- If any Lane receives an EIOS on either Pseudo Port then:
  - Transmit EIOS on every Lane of the Pseudo Port that did not receive EIOS and place the Transmitters in Electrical Idle. Place the Transmitters of the other Pseudo Port in Electrical Idle; EIOS is not transmitted by the other Pseudo Port.
  - Set the RT_next_data_rate variable to 2.5 GT/s.
  - Set the RT_error_data_rate variable to 2.5 GT/s.
  - The Compliance Receive additional rules no longer apply unless the rules for entering this Section are met again.

4.3.6.15 *Enter Compliance Rules*

The Retimer follows these additional rules if the Retimer is exiting Electrical Idle after entering Electrical Idle as a result of Hot Reset, and the Retimer Enter Compliance bit is set in the Retimer. The purpose of the following rules is to support Link operation with a Retimer when the Link partners enter compliance as a result of the Enter Compliance bit in the Link Control 2 Register set to 1b in both Link Components and a Hot Reset occurring on the Link. Retimers do not support Link operation if the Link partners enter compliance when they exit detect if the entry into detect was not caused by a Hot Reset.

Retimers must support the following register fields in an implementation specific manner:

- Retimer Target Link Speed
  - One field per Retimer
  - Type = RWS
  - Size = 3 bits
  - Default = 001b
  - Encoding:
    - 001b = 2.5 GT/s
    - 010b = 5.0 GT/s
    - 011b = 8.0 GT/s
    - 100b = 16.0 GT/s
- Retimer Transmit Margin
  - One field per Pseudo Port
  - Type = RWS
  - Size = 3 bits
  - Default = 000b
  - Encoding:
    - 000b = Normal Operating Range
    - 001b-111b = As defined in Section 8.3.4, not all encodings are required to be implemented
- Retimer Enter Compliance
  - One bit per Retimer
  - Type = RWS
  - Size = 1 bit
  - Default = 0b
  - Encoding:

        - 0b = do not enter compliance
        - 1b = enter compliance
- Retimer Enter Modified Compliance
    - One bit per Retimer
    - Type = RWS
    - Size = 1 bit
    - Default = 0b
    - Encoding:
        - 0b = do not enter modified compliance
        - 1b = enter modified compliance
- Retimer Compliance SOS
    - One bit per Retimer
    - Type = RWS
    - Size = 1 bit
    - Default = 0b
    - Encoding:
        - 0b = Send no SKP Ordered Sets between sequences when sending the Compliance Pattern or Modified Compliance Pattern with 8b/10b encoding.
        - 1b = Send two SKP Ordered Sets between sequences when sending the Compliance Pattern or Modified Compliance Pattern with 8b/10b encoding.
- Retimer Compliance Preset/De-emphasis
    - One field per Pseudo Port
    - Type = RWS
    - Size = 4 bits
    - Default = 0000b
    - Encoding when Retimer Target Link Speed is 5.0 GT/s:
        - 0000b -6.0 dB
        - 0001b -3.5 dB
    - Encoding when Retimer Target Link Speed is 8.0 GT/s or higher: the Transmitter Preset.

A Retimer must examine the values in the above registers when the Retimer exits from Hot Reset. If the Retimer Enter Compliance bit is Set the following rules apply:

- The Retimer adjusts its data rate as defined by Retimer Target Link Speed. No data is forwarded until the data rate change has occurred.

- The Retimer configures its Transmitters according to Retimer Compliance Preset/De-emphasis on a per Pseudo Port basis.
- The Retimer must forward the Compliance or Modified Compliance Pattern when it has locked to the pattern. The Retimer must search for the Compliance Pattern if the Retimer Enter Modified Compliance bit is Clear or search for the Modified Compliance Pattern if the Retimer Enter Modified Compliance bit is Set. This occurs independently on each Lane in each direction.
- When using 8b/10b encoding, a particular Lane's Receiver independently determines a successful lock to the incoming Modified Compliance Pattern or Compliance Pattern by looking for any one occurrence of the Modified Compliance Pattern or Compliance Pattern.
    - An occurrence is defined above as the sequence of 8b/10b Symbols defined in Section 4.2.8.
    - In the case of the Modified Compliance Pattern, the error status Symbols are not to be used for the lock process since they are undefined at any given moment.
    - Lock must be achieved within 1.0 ms of receiving the Modified Compliance Pattern.
- When using 128b/130b encoding each Lane determines Pattern Lock independently when it achieves Block Alignment as described in Section 4.2.2.2.1.
    - Lock must be achieved within 1.5 ms of receiving the Modified Compliance Pattern or Compliance Pattern.
- When 128b/130b encoding is used, Symbols with errors are forwarded unmodified by default, or may optionally be corrected to remove error pollution. The default behavior must be supported and the method of selecting the optional behavior, if supported, is implementation specific.
- When 8b/10b encoding is used, Symbols with errors are replaced with the D21.3 Symbol with incorrect disparity by default, or may optionally be corrected to remove error pollution. The default behavior must be supported.
- Once locked, the Retimer keeps an internal count of received Symbol errors, on a per-Lane basis. If the Retimer is forwarding the Modified Compliance Pattern then the error status Symbol when using 8b/10b encoding or the Error_Status field when using 128b/130b encoding is forwarded unmodified by default, or may optionally be redefined as it is transmitted by the Retimer. The default behavior must be supported and the method of selecting the optional behavior, if supported, is implementation specific. The Retimer is permitted to make the pattern lock and Lane error information available in an implementation specific manner, on a per-Lane basis.
- If an EIOS is received on any Lane then:
    - All Lanes in that direction transmit 8 EIOS and then all Transmitters in that direction are placed in Electrical Idle.
    - When both directions have sent 8 EIOS and placed their Transmitters in Electrical Idle the data rate is changed to 2.5 GT/s.
    - Set the RT_next_data_rate variable to 2.5 GT/s.
    - Set the RT_error_data_rate variable to 2.5 GT/s.

- The Retimer Enter Compliance bit and Retimer Enter Modified Compliance bit are both set to 0b.
- The above additional rules no longer apply unless the rules for entering this Section and clause are met again.

4.3.7 Execution Mode Rules

In Execution mode, Retimers directly control all information transmitted by the Pseudo Ports rather than forwarding information.

4.3.7.1 *CompLoadBoard Rules*

While the Retimer is in the CompLoadBoard (Compliance Load Board) state both Pseudo Ports are executing the protocol as regular Ports, generating Symbols as specified in the following sub-sections on each Port, rather than forwarding from one Pseudo Port to the other.

IMPLEMENTATION NOTE

Passive Load on Transmitter
This state is entered when a passive load is placed on one Pseudo Port, and the other Pseudo Port is receiving traffic.

4.3.7.1.1 CompLoadBoard.Entry

- RT_LinkUp = 0b.
- The Pseudo Port that received Compliance Pattern (Pseudo Port A) does the following:
  - The data rate remains at 2.5 GT/s.
  - The Transmitter is placed in Electrical Idle.
  - The Receiver ignores incoming Symbols.
- The other Pseudo Port (Pseudo Port B) does the following:
  - The data rate remains at 2.5 GT/s.
  - The Transmitter is placed in Electrical Idle. Receiver detection is performed on all Lanes as described in Section 8.4.5.7.
  - The Receiver ignores incoming Symbols.
- If Pseudo Port B's Receiver detection determines there are no Receivers attached on any Lanes, then the next state for both Pseudo Ports is CompLoadBoard.Exit.
- Else the next state for both Pseudo Ports is CompLoadBoard.Pattern.

4.3.7.1.2 CompLoadBoard.Pattern

When The Retimer enters CompLoadBoard.Pattern the following occur:

- Pseudo Port A does the following:
  - The Transmitter remains in Electrical Idle.
  - The Receiver ignores incoming Symbols.
- Pseudo Port B does the following:
  - The Transmitter sends out the Compliance Pattern on all Lanes that detected a Receiver at the data rate and de-emphasis/preset level determined as described in Section 4.2.6.2.2, (i.e., each consecutive entry into CompLoadBoard advances the pattern), except that the Setting is not set to Setting #1 during Polling.Configuration. Setting #26 and later are not used if Pseudo Port B has received a TS1 or TS2 Ordered Set (or their complement) since the exit of Fundamental Reset. If the new data rate is not 2.5 GT/s, the Transmitter is placed in Electrical Idle prior to the data rate change. The period of Electrical Idle must be greater than 1 ms but it is not to exceed 2 ms.
- If Pseudo Port B detects an Electrical Idle exit of any Lane that detected a Receiver, then the next state for both Pseudo Ports is CompLoadBoard.Exit.

4.3.7.1.3 CompLoadBoard.Exit

When The Retimer enters CompLoadBoard.Exit the following occur:

- The Pseudo Port A:
  - Data rate remains at 2.5 GT/s.
  - The Transmitter sends the Electrical Idle Exit pattern described in section 4.3.6.3, on the Lane(s) where Electrical Idle exit was detected on Pseudo Port B for 1 ms. Then the Transmitter is placed in Electrical Idle.
  - The Receiver ignores incoming Symbols.
- Pseudo Port B:
  - If the Transmitter is transmitting at a rate other than 2.5 GT/s the Transmitter sends eight consecutive EIOS.
  - The Transmitter is placed in Electrical Idle. If the Transmitter was transmitting at a rate other than 2.5 GT/s the period of Electrical Idle must be at least 1.0 ms.
  - Data rate is changed to 2.5 GT/s, if not already at 2.5 GT/s.
- Both Pseudo Ports are placed in Forwarding mode.

IMPLEMENTATION NOTE

TS1 Ordered Sets in Forwarding mode

Once in Forwarding mode one of two things will likely occur:

- TS1 Ordered Sets are received and forwarded from Pseudo Port's B Receiver to Pseudo Port's A Transmitter. Link training continues.
- Or: TS1 Ordered Sets are not received because 100 MHz pulses are being received on a lane from the compliance load board, advancing the Compliance Pattern. In this case the Retimer must transition from Forwarding mode to CompLoadBoard when the device attached to Pseudo Port A times out from Polling.Active to Polling.Compliance. The Retimer advances the Compliance Pattern on each entry to CompLoadBoard.

4.3.7.2 *Link Equalization Rules*

When in the Execution mode performing Link Equalization, the Pseudo Ports act as regular Ports, generating Symbols on each Port rather than forwarding from one Pseudo Port to the other. When the Retimer is in Execution mode it must use the Lane and Link numbers stored in RT_captured_lane_number and RT_captured_link_number.

This mode is entered while the Upstream and Downstream Ports on the Link are in negotiation to enter Phase 2 of the Equalization procedure following the procedure for switching to Execution mode described in Section 4.3.5.

4.3.7.2.1 Downstream Lanes

The LF and FS values received in two consecutive TS1 Ordered Sets when the Upstream Port is in Phase 0 must be stored for use during Phase 3, if the Downstream Pseudo Port wants to adjust the Upstream Port's Transmitter.

4.3.7.2.1.1 Phase 2

Transmitter behaves as described in Section 4.2.6.4.2.1.2 except as follows:

- If the data rate of operation is 16.0 GT/s, the Retimer Equalization Extend bit of the transmitted TS1 Ordered Sets is set to 1b when the Upstream Pseudo Port state is Phase 2 Active, and it is set to 0b when the Upstream Pseudo Port state is Phase 2 Passive.
- Next phase is Phase 3 Active if all configured Lanes receive two consecutive TS1 Ordered Sets with EC=11b.
- Else, next state is Force Timeout after a 32 ms timeout with a tolerance of -0 ms and +4 ms.

4.3.7.2.1.2 Phase 3 Active

If the data rate of operation is 8.0 GT/s then the transmitter behaves as described in Section 4.2.6.4.2.1.3 except the 24 ms timeout is 2.5 ms and as follows:

- Next phase is Phase 3 Passive if all configured Lanes are operating at their optimal settings.
- Else, next state is Force Timeout after a timeout of 2.5 ms with a tolerance of -0 ms and +0.1 ms

If the data rate of operation is 16.0 GT/s then the transmitter behaves as described in Section 4.2.6.4.2.1.3 except the 24 ms timeout is 22 ms and as follows:

- The Retimer Equalization Extend bit of transmitted TS1 Ordered Sets is set to 0b.
- Next phase is Phase 3 Passive if all configured Lanes are operating at their optimal settings and all configured Lanes receive two consecutive TS1 Ordered Sets with the Retimer Equalization Extend bit set to 0b.
- Else, next state is Force Timeout after a timeout of 22 ms with a tolerance of -0 ms and +1.0 ms.

4.3.7.2.1.3 Phase 3 Passive

- Transmitter sends TS1 Ordered Sets with EC = 11b, Retimer Equalization Extend = 0b, and the Transmitter Preset field and the Coefficients fields must not be changed from the final value transmitted in Phase 3 Active.
- The transmitter switches to Forwarding mode when the Upstream Pseudo Port exits Phase 3.

4.3.7.2.2 Upstream Lanes

The LF and FS values received in two consecutive TS1 Ordered Sets when the Downstream Port is in Phase 1 must be stored for use during Phase 2, if the Upstream Pseudo Port wants to adjust the Downstream Port's Transmitter.

4.3.7.2.2.1 Phase 2 Active

If the data rate of operation is 8.0 GT/s then the transmitter behaves as described in Section 4.2.6.4.2.2.3 except the 24 ms timeout is 2.5 ms and as follows:

- Next state is Phase 2 Passive if all configured Lanes are operating at their optimal settings.
- Else, next state is Force Timeout after a 2.5 ms timeout with a tolerance of -0 ms and +0.1 ms

If the data rate of operation is 16.0 GT/s then the transmitter behaves as described in Section 4.2.6.4.2.2.3 except the 24 ms timeout is 22 ms and as follows:

- The Retimer Equalization Extend bit of transmitted TS1 Ordered Sets is set to 0b.
- Next phase is Phase 2 Passive if all configured Lanes are operating at their optimal settings and all configured Lanes receive two consecutive TS1 Ordered Sets with the Retimer Equalization Extend bit set to 0b.
- Else, next state is Force Timeout after a 22 ms timeout with a tolerance of -0 ms and +1.0 ms.

4.3.7.2.2.2 Phase 2 Passive

- Transmitter sends TS1 Ordered Sets with EC = 10b, Retimer Equalization Extend = 0b, and the Transmitter Preset field and the Coefficients fields must not be changed from the final value transmitted in Phase 2 Active.
- If the data rate of operation is 8.0 GT/s, the next state is Phase 3 when the Downstream Pseudo Port has completed Phase 3 Active.
- If the data rate of operation is 16.0 GT/s, the next state is Phase 3 when the Downstream Pseudo Port has started Phase 3 Active.

4.3.7.2.2.3 Phase 3

Transmitter follows Phase 3 rules for Upstream Lanes in Section 4.2.6.4.2.2.4 except as follows:

- If the data rate of operation is 16.0 GT/s, the Retimer Equalization Extend bit of the transmitted TS1 Ordered Sets is set to 1b when the Downstream Pseudo Port state is Phase 3 Active, and it is set to 0b when the Downstream Pseudo Port state is Phase 3 Passive.
- If all configured Lanes receive two consecutive TS1 Ordered Sets with EC=00b then the Retimer switches to Forwarding mode.
- Else, next state is Force Timeout after a timeout of 32 ms with a tolerance of -0 ms and +4 ms

4.3.7.2.3 Force Timeout

- The Electrical Idle Exit Pattern described in section 4.3.6.3 is transmitted by both Pseudo Ports at the current data rate for a minimum of 1.0 ms.
- If a Receiver receives an EIOS or infers Electrical Idle (inferred Electrical Idle is via not detecting an exit from Electrical Idle (see Table 4-21) on any Lane then, the Transmitters on all Lanes of the opposite Pseudo Port send an EIOSQ and are then placed in Electrical Idle.
- If both Paths have placed their Transmitters in Electrical Idle then, the RT_next_data_rate is set to the RT_error_data_rate, and the RT_error_data_rate is set to 2.5 GT/s, on both Pseudo Ports, and the Retimer enters Forwarding mode.
    - The Transmitters of both Pseudo Ports must be in Electrical Idle for at least 6 µs, before forwarding data.
- Else after a 48 ms timeout, the RT_next_data_rate is set to 2.5 GT/s and the RT_error_data_rate is set to 2.5 GT/s, on both Pseudo Ports, and the Retimer enters Forwarding mode.

IMPLEMENTATION NOTE

Purpose of Force Timeout State

The purpose of this state is to ensure both Link Components are in Recovery.Speed at the same time so they go back to the previous data rate.

4.3.7.3 *Slave Loopback*

Retimers optionally support Slave Loopback in Execution mode. By default Retimers are configured to forward loopback between loopback master and loopback slave. Retimers are permitted to allow configuration in an implementation specific manner to act as a loopback slave on either Pseudo Port. The other Pseudo Port that is not the loopback slave, places its Transmitter in Electrical Idle, and ignores any data on its Receivers.

4.3.7.3.1 Slave Loopback.Entry

The Pseudo Port that did not receive the TS1 Ordered Set with the Loopback bit set to 1b does the following:

- The Transmitter is placed in Electrical Idle.
- The Receiver ignores incoming Symbols.

The Pseudo Port that did receive the TS1 Ordered Set with the Loopback bit set to 1b behaves as the loopback slave as described in Section 4.2.6.10.1 with the following exceptions:

- The statement "LinkUp = 0b (False)" is replaced by "RT_LinkUp = 0b".
- The statement "If Loopback.Entry was entered from Configuration.Linkwidth.Start" is replaced by "If Slave.Loopback.Entry was entered when RT_LinkUp =0b".
- References to Loopback.Active become Slave Loopback.Active.

4.3.7.3.2 Slave Loopback.Active

The Pseudo Port that did not receive the TS1 Ordered Set with the Loopback bit set to 1b does the following:

- The Transmitter remains in Electrical Idle.
- The Receiver continues to ignore incoming Symbols.

The Pseudo Port that did receive the TS1 Ordered Set with the Loopback bit set to 1b behaves as the loopback slave as described in Section 4.2.6.10.2 with the following exception:

- References to Loopback.Exit become Slave Loopback.Exit.

4.3.7.3.3 Slave Loopback.Exit

The Pseudo Port that did not receive the TS1 Ordered Set with the Loopback bit set to 1b must do the following:

- Maintain the Transmitter in Electrical Idle.
- Set the data rate to 2.5 GT/s.
- The Receiver continues to ignore incoming Symbols.

The Pseudo Port that did receive the TS1 or TS2 Ordered Set with the Loopback bit set to 1b must behave as the loopback slave as described in Section 4.2.6.10.3 with the following exception:

- The clause "The next state of the loopback master and loopback slave is Detect" becomes "The Data rate is set to 2.5 GT/s and then both Pseudo Ports are placed in Forwarding mode".

4.3.8 Retimer Latency

This Section defines the requirements on allowed Retimer Latency.

4.3.8.1 *Measurement*

Latency must be measured when the Retimer is in Forwarding mode and the Link is in L0, and is defined as the time from when the last bit of a Symbol is received at the input pins of one Pseudo Port to when the equivalent bit is transmitted on the output pins of the other Pseudo Port.

Retimer vendors are strongly encouraged to specify the latency of the Retimer in their data sheets.

Retimers are permitted to have different latencies at different data rates, and when this is the case it is strongly recommended the latency be specified per data rate.

4.3.8.2 *Maximum Limit on Retimer Latency*

Retimer latency shall be less than the following limit, when not operating in SRIS.

Table 4-22. Retimer Latency Limit not SRIS (Symbol times)

	2.5 GT/s	5.0 GT/s	8.0 GT/s	16.0 GT/s
Maximum Latency	32	32	64	128

4.3.8.3 *Impacts on Upstream and Downstream Ports*

Retimers will add to the channel latency. The round trip delay is 4 times the specified latency when two Retimers are present. It is recommended that designers of Upstream and Downstream Ports consider Retimer latency when determining the following characteristics:

- Data Link Layer Retry Buffer size
- Transaction Layer Receiver buffer size and Flow Control Credits
- Data Link Layer REPLAY_TIMER Limits

Additional buffering (replay or FC) may be required to compensate for the additional channel latency.

4.3.9 SRIS

Retimers are permitted but not required to support SRIS. Retimers that support SRIS must provide a mechanism for enabling the higher rate of SKP Ordered Set transmission, as Retimers must generate SKP Ordered Sets while in Execution mode. Retimers that are enabled to support SRIS will incur additional latency in the elastic store between receive and transmit clock domains. The additional latency is required to handle the case where a Max_Payload_Size TLP is transmitted and SKP Ordered Sets, which are scheduled, are not sent. The additional latency is a function of Link width and Max_Payload_Size. This additional latency is not included in Table 4-22.

A SRIS capable Retimer must provide an implementation specific mechanism to configure the supported Max_Payload_Size while in SRIS, that must be configured to be greater than or equal to the Max_Payload_Size for the Transmitter in the Port that the Pseudo Port is receiving. Retimer latency must be less than the following limit for the current supported Max_Payload_Size, with SRIS.

Table 4-23: Retimer Latency Limit SRIS (Symbol times)

Max_Payload_Size	2.5 GT/s	5.0 GT/s	8.0 GT/s	16.0 GT/s
128 Bytes	34 (max)	34 (max)	66 (max)	130 (max)
256 Bytes	36 (max)	36 (max)	68 (max)	132 (max)
512 Bytes	39 (max)	39 (max)	71 (max)	135 (max)
1024 Bytes	46 (max)	46 (max)	78 (max)	142 (max)
2048 Bytes	59 (max)	59 (max)	91 (max)	155 (max)
4096 Bytes	86 (max)	86 (max)	118 (max)	182 (max)

IMPLEMENTATION NOTE

Retimer Latency with SRIS Calculation:

Table 4-23 is calculated assuming that the link is operating at x1 Link width. The max Latency is the sum of Table 4-22 and the additional latency required in the elastic store for SRIS clock compensation. The SRIS additional latency in symbol times required for SRIS clock compensation is described in the following equation:

$$2 * \left(\frac{\left({}^{(\text{SRIS Link Payload Size} + \text{TLPOverhead})}/_{\text{LinkWidth}} \right)}{\text{SKP}_{\text{rate}}} \right)$$

Where:

SRIS Link Payload Size: is the value programmed in the Retimer.

TLPOverhead: Represents the additional TLP components which consume Link bandwidth (TLP Prefix, header, LCRC, framing Symbols) and is treated here as a constant value of 28 Symbols.

LinkWidth: The operating width of the Link.

SKP_rate: The rate that a transmitter schedules SKP Ordered Sets when using 8b/10b encoding, 154, see Section 4.2.7.3. When using the 128b/130b encoding the effective rate is the same.

The nominal latency would be ½ of the SRIS additional latency, and is the nominal fill of the elastic store. This makes a worse case assumption that every blocked SKP Ordered Set requires an additional symbol of latency in the elastic store. When a Max Payload Size TLP is transmitted the actual fill of the elastic store could go to zero, or two times the nominal fill depending on the relative clock frequencies. Link width down configure may occur at any time, a lane fails for example, and this down configure may occur faster than the Retimer is able to adjust its nominal elastic store. By default Retimer's will configure its nominal fill based on x1 link width, regardless of the actual current link width.

Retimers that optionally support SRIS, may optionally support a dynamic elastic store. Dynamic elastic store changes the nominal buffer fill as the link width changes. Retimers are permitted delay the Link LTSSM transitions, only while the Link down configures, in Configuration, for up to 40us. Retimers are permitted to delay the TS1 Order Set to TS2 Ordered Set transition between Configuration.Lanenum.Accept and Configuration.Complete to increase their elastic store.

4.3.10 L1 PM Substates Support

The following Section describes the Retimer's requirements to support the optional L1 PM Substates.

The Retimer enters L1.1 when CLKREQ# is sampled as deasserted. The following occur:

- REFCLK to the Retimer is turned off.
- The PHY remains powered.

- The Retimer places all Transmitters in Electrical Idle on both Pseudo Ports (if not already in Electrical Idle, the expected state). Transmitters maintain their common mode voltage.
- The Retimer must ignore any Electrical Idle exit from all Receivers on both Pseudo Ports.

The Retimer exits L1.1 when CLKREQ# is sampled as asserted. The following occur:

- REFCLK to the Retimer is enabled.
- Normal operation of the Electrical Idle exit circuit is resumed on all Lanes of both Pseudo Ports of the Retimer.
- Normal exit from Electrical Idle exit behavior is resumed, See 4.3.6.3.

Retimers do not support L1.2, but if they support L1.1 and the removal of the reference clock then they must not interfere with the attached components ability to enter L1.2.

Retimer vendors must document specific implementation requirements applying to CLKREQ#. For example, a Retimer implementation that does not support the removal of the reference clock might require an implementation to pull CLKREQ# low.

IMPLEMENTATION NOTE

CLKREQ# Connection Topology with a Retimer Supporting L1 PM Substates

In this platform configuration Downstream Port (A) has only a single CLKREQ# signal. The Upstream and Downstream Ports' CLKREQ# (A and C), and the Retimer's CLKREQB# signals are connected to each other. In this case, Downstream Port (A), must assert CLKREQ# signal whenever it requires a reference clock. Component A, Component B, and the Retimer have their REFCLKs removed/restored at the same time.

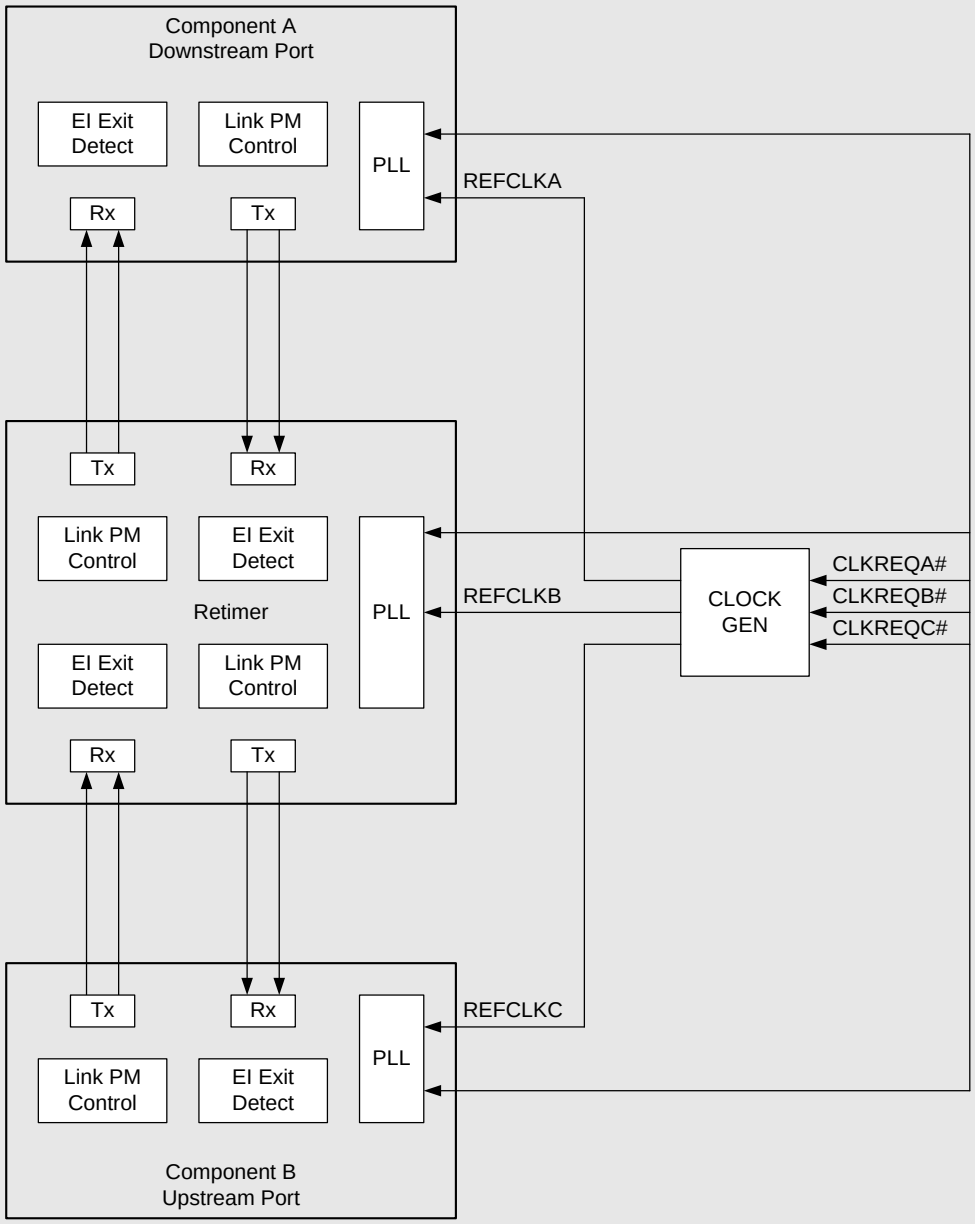


Figure 4-34: Retimer CLKREQ# Connection Topology

4.3.11 Retimer Configuration Parameters

Retimers must provide an implementation specific mechanism to configure each of the parameters in this Section.

The parameters are split into two groups: parameters that are configurable globally for the Retimer and parameters that are configurable for each physical Retimer Pseudo Port.

If a per Pseudo Port parameter only applies to an Upstream or a Downstream Pseudo Port the Retimer is not required to provide an implementation specific mechanism to configure the parameter for the other type of Pseudo Port.

4.3.11.1 *Global Parameters*

- **Port Orientation Method**. This controls whether the Port Orientation is determined dynamically as described in Section 4.4.6.2, or statically based on vendor assignment of Upstream and Downstream Pseudo Ports. If the Port Orientation is set to static the Retimer is not required to dynamically adjust the Port Orientation as described in Section 4.4.6.2. The default behavior is for the Port Orientation to be dynamically determined.
- **Maximum Data Rate.** This controls the maximum data rate that the Retimer sets in the Data Rate Identifier field of training sets that the Retimer transmits. Retimers that support only the 2.5 GT/s speed are permitted not to provide this configuration parameter.
- **SRIS Enable.** This controls whether the Retimer is configured for SRIS and transmits SKP Ordered sets at the SRIS mode rate when in Execution mode. Retimers that do not support SRIS and at least one other clocking architecture are not required to provide this configuration parameter.
- **SRIS Link Payload Size.** This controls the maximum payload size the Retimer supports while in SRIS. The value must be selectable from all the Maximum Payload Sizes shown in Table 4-22. The default value of this parameter is to support a payload size of 4096 bytes. Retimers that do not support SRIS are not required to provide this configuration parameter.

  The following are example of cases where it might be appropriate to configure the SRIS Link Payload Size to a smaller value than the default:

  - A Retimer is part of a motherboard with a Root Port that supports a maximum payload size less than 4096 bytes.
  - A Retimer is part of an add-in card with an Endpoint that supports a Maximum Payload Size less than 4096 bytes.
  - A Retimer is located Downstream of the Downstream Port of a Switch integrated as part of a system, the Root Port silicon supports a Maximum Payload Size less than 4096 bytes and the system does not support peer to peer traffic.

4.3.11.2 *Per Physical Pseudo Port Parameters*

- **Port Orientation**. This is applicable only when the Port Orientation Method is configured for static determination. This is set for either Upstream or Downstream. Each Pseudo Port must be configured for a different orientation, or the behavior is undefined.

- **Selectable De-emphasis**. When the Downstream Pseudo Port is operating at 5.0 GT/s this controls the transmit de-emphasis of the Link to either -3.5 dB or -6 dB in specific situations and the value of the Selectable De-emphasis field in training sets transmitted by the Downstream Pseudo Port. See Section 4.2.6 for detailed usage information. When the Link Segment is not operating at the 5.0 GT/s speed, the setting of this bit has no effect. Retimers that support only the 2.5 GT/s speed are permitted not to provide this configuration parameter.
- **Rx Impedance Control.** This controls whether the Retimer dynamically applies and removes 50 Ω terminations or statically has 50 Ω terminations present. The value must be selectable from Dynamic, Off, and On. The default behavior is Dynamic.
- **Tx Compliance Disable.** This controls whether the Retimer transmits the Compliance Pattern in the CompLoadBoard.Pattern state. The default behavior is for the Retimer to transmit the Compliance Pattern in the CompLoadBoard.Pattern state. If TX Compliance Pattern is set to disabled, the Retimer Transmitters remain in Electrical Idle and do not transmit Compliance Pattern in CompLoadBoard.Pattern – all other behavior in the CompLoadBoard state is the same.
- **Pseudo Port Slave Loopback.** This controls whether the Retimer operates in a Forwarding mode during loopback on the Link or enters Slave Loopback on the Pseudo Port. The default behavior is for the Retimer to operate in Forwarding mode during loopback. Retimers that do not support optional Slave Loopback are permitted not to provide this configuration parameter. This configuration parameter shall only be enabled for one physical Port. Retimer behavior is undefined if the parameter is enabled for more than one physical Port.
- **Downstream Pseudo Port 8GT TX Preset.** This controls the initial TX preset used by the Downstream Pseudo Port transmitter for 8.0 GT/s transmission. The default value is implementation specific. The value must be selectable from all applicable values in Table 4-3.
- **Downstream Pseudo Port 16GT TX Preset.** This controls the initial TX preset used by the Downstream Pseudo Port transmitter for 16.0 GT/s transmission. The default value is implementation specific. The value must be selectable from all applicable values in Table 4-3.
- **Downstream Pseudo Port 8GT Requested TX Preset.** This controls the initial transmitter preset value used in the EQ TS2 Ordered Sets transmitted by the Downstream Pseudo Port for use at 8.0 GT/s. The default value is implementation specific. The value must be selectable from all values in Table 4-3.
- **Downstream Pseudo Port 16GT Requested TX Preset**. This controls the initial transmitter preset value used in the 8GT EQ TS2 Ordered Sets transmitted by the Downstream Pseudo Port for use at 16.0 GT/s. The default value is implementation specific. The value must be selectable from all values in Table 4-3.
- **Downstream Pseudo Port 8GT RX Hint.** This controls the Receiver Preset Hint value used in the EQ TS2 Ordered Sets transmitted by the Downstream Pseudo Port for use at 8.0 GT/s. The default value is implementation specific. The value must be selectable from all values in Table 4-4.

4.3.12 In Band Register Access

- Retimers operating at 16.0 GT/s may optionally support inband read only access. Control SKP Ordered Sets at 16.0 GT/s provide the mechanism via the Margin Command 'Access Retimer Register', see Table 4-19. Retimers that support inband read only access must return a non-zero value for the DWORD at Registers offsets 80h and 84h. Retimers that do not support inband read only access must return a zero value.
- Register offsets between A0h and FFh are designated as Vendor Defined register space.
- Register offsets between 00h and 87h are Reserved for PCI-SIG future use.

5

5 Power Management

This chapter describes power management (PM) capabilities and protocols.

5.1 Overview

Power Management states are as follows:

- D states are associated with a particular Function
  - D0 is the operational state and consumes the most power
  - D1 and D2 are intermediate power saving states
  - $D3_{hot}$ is a very low power state
  - $D3_{cold}$ is the power off state
- L states are associated with a particular Link
  - L0 is the operational state
  - L0s, L1, L1.0, L1.1, and L1.2 are various lower power states

Other specifications define related power states (e.g. S states). This specification does not describe relationships between those states and D/L/B states.

PM provides the following services:

- A mechanism to identify power management capabilities of a given Function
- The ability to transition a Function into a certain power management state
- Notification of the current power management state of a Function
- The option to wakeup the system on a specific event

PM is compatible with the *PCI Bus Power Management Interface Specification*, and the *Advanced Configuration and Power Interface Specification*. This chapter also defines PCI Express native power management extensions.

PM defines Link power management states that a PCI Express physical Link is permitted to enter in response to either software driven D-state transitions or active state Link power management activities. PCI Express Link states are not visible directly to legacy bus driver software, but are derived from the power management state of the components residing on those Links. Defined Link states are L0, L0s, L1, L2, and L3. The power savings increase as the Link state transitions from L0 through L3.

Components may wakeup the system using a wakeup mechanism followed by a power management event (PME) Message. PCI Express systems may provide the optional auxiliary power supply (Vaux) needed for wakeup operation from states where the main power supplies are off.

The specific definition and requirements associated with Vaux are form-factor specific, and throughout this document the terms "auxiliary power" and "Vaux" should be understood in reference to the specific form factor in use.

Another distinction of the PCI Express-PM PME mechanism is its separation of the following two PME tasks:

- Reactivation (wakeup) of the associated resources (i.e., re-establishing reference clocks and main power rails to the PCI Express components)
- Sending a PME Message to the Root Complex

Active State Power Management (ASPM) is an autonomous hardware-based, active state mechanism that enables power savings even when the connected components are in the D0 state. After a period of idle Link time, an ASPM Physical-Layer protocol places the idle Link into a lower power state. Once in the lower-power state, transitions to the fully operative L0 state are triggered by traffic appearing on either side of the Link. ASPM may be disabled by software. Refer to Section 5.4.1 for more information on ASPM.

5.2 Link State Power Management

PCI Express defines Link power management states, replacing the bus power management states that were defined by the *PCI Bus Power Management Interface Specification*. Link states are not visible to PCI-PM legacy compatible software, and are either derived from the power management D-states of the corresponding components connected to that Link or by ASPM protocols (see Section 5.4.1).

Note that the PCI Express Physical Layer may define additional intermediate states. Refer to Chapter 4 for more detail on each state and how the Physical Layer handles transitions between states.

PCI Express-PM defines the following Link power management states:

- L0 – Active state.

L0 support is required for both ASPM and PCI-PM compatible power management.

All PCI Express transactions and other operations are enabled.

- L0s – A low resume latency, energy saving "standby" state.

L0s support is optional for ASPM unless the applicable form factor specification for the Link explicitly requires L0s support.

All main power supplies, component reference clocks, and components' internal PLLs must be active at all times during L0s. TLP and DLLP transmission is disabled for a Port whose Link is in Tx_L0s.

The Physical Layer provides mechanisms for quick transitions from this state to the L0 state. When common (distributed) reference clocks are used on both sides of a Link, the transition time from L0s to L0 is desired to be less than 100 Symbol Times.

It is possible for the Transmit side of one component on a Link to be in L0s while the Transmit side of the other component on the Link is in L0.

- L1 – Higher latency, lower power "standby" state.

L1 support is required for PCI-PM compatible power management. L1 is optional for ASPM unless specifically required by a particular form factor.

When L1 PM Substates is enabled by setting one or more of the enable bits in the L1 PM Substates Control 1 Register this state is referred to as the L1.0 substate.

All main power supplies must remain active during L1. As long as they adhere to the advertised L1 exit latencies, implementations are explicitly permitted to reduce power by applying techniques such as, but not limited to, periodic rather than continuous checking for Electrical Idle exit, checking for Electrical Idle exit on only one Lane, and powering off of unneeded circuits. All platform-provided component reference clocks must remain active during L1, except as permitted by Clock Power Management (using CLKREQ#) and/or L1 PM Substates when enabled. A component's internal PLLs may be shut off during L1, enabling greater power savings at a cost of increased exit latency.[67]

The L1 state is entered whenever all Functions of a Downstream component on a given Link are programmed to a D-state other than D0. The L1 state also is entered if the Downstream component requests L1 entry (ASPM) and receives positive acknowledgement for the request.

Exit from L1 is initiated by an Upstream-initiated transaction targeting a Downstream component, or by the Downstream component's initiation of a transaction heading Upstream. Transition from L1 to L0 is desired to be a few microseconds.

TLP and DLLP transmission is disabled for a Link in L1.

L1 PM Substates – optional L1.1 and L1.2 substates of the L1 low power Link state for PCI-PM and ASPM.

In the L1.1 substate, the Link common mode voltages are maintained. The L1.1 substate is entered when the Link is in the L1.0 substate and conditions for entry into L1.1 substate are met. See Section 5.5.1 for details.

In the L1.2 substate, the Link common mode voltages are not required to be maintained. The L1.2 substate is entered when the Link is in the L1.0 substate and conditions for entry into L1.2 substate are met. See Section 5.5.1. for details.

Exit from all L1 PM Substates is initiated when the CLKREQ# signal is asserted (see Sections 5.5.2.1 and 5.5.3.3).

- L2/L3 Ready – Staging point for L2 or L3.

L2/L3 Ready transition protocol support is required.

L2/L3 Ready is a pseudo-state (corresponding to the LTSSM L2 state) that a given Link enters when preparing for the removal of power and clocks from the Downstream component or from both attached components. This process is initiated after PM software transitions a device into a D3 state, and subsequently calls power management software to initiate the removal of power and clocks. After the Link enters the L2/L3 Ready state the component(s) are ready for power removal.

---

[67] For example, disabling the internal PLL may be something that is desirable when in $D3_{hot}$, but not so when in D1 or D2.

After main power has been removed, the Link will either transition to L2 if Vaux is provided and used, or it will transition to L3 if no Vaux is provided or used. Note that these are PM pseudo-states for the Link; under these conditions, the LTSSM will in, general, operate only on main power, and so will power off with main power removal.

The L2/L3 Ready state entry transition process must begin as soon as possible following the acknowledgment of a PME_Turn_Off Message, (i.e., the injection of a PME_TO_Ack TLP). The Downstream component initiates L2/L3 Ready entry by sending a PM_Enter_L23 DLLP. Refer to Section 5.7 for further detail on power management system Messages.

TLP and DLLP transmission is disabled for a Link in L2/L3 Ready.

Note: Exit from L2/L3 Ready back to L0 will be through intermediate LTSSM states. Refer to Chapter 4 for detailed information.

- L2 – Auxiliary-powered Link, deep-energy-saving state.

L2 support is optional, and dependent upon the presence of Vaux.

A component may only consume Vaux power if enabled to do so as described in Section 5.6.

In L2, the component's main power supply inputs and reference clock inputs are shut off.

When in L2, any Link reactivation wakeup logic (Beacon or WAKE#), PME context, and any other "keep alive" logic is powered by Vaux.

TLP and DLLP transmission is disabled for a Link in L2.

- L3 – Link Off state.

When no power is present, the component is in the L3 state.

- LDn – A transitional Link Down pseudo-state prior to L0.

This pseudo-state is associated with the LTSSM states Detect, Polling, and Configuration, and, when applicable, Disabled, Loopback, and Hot Reset.

Refer to Section 4.2 for further detail relating to entering and exiting each of the L-states between L0 and L2/L3 Ready (L2.Idle from the Chapter 4 perspective). The L2 state is an abstraction for PM purposes distinguished by the presence of auxiliary power, and should not be construed to imply a requirement that the LTSSM remain active.

The electrical section specifies the electrical properties of drivers and Receivers when no power is applied. This is the L3 state but the electrical section does not refer to L3.

Figure 5-1 shows an overview of L-state transitions that may occur.

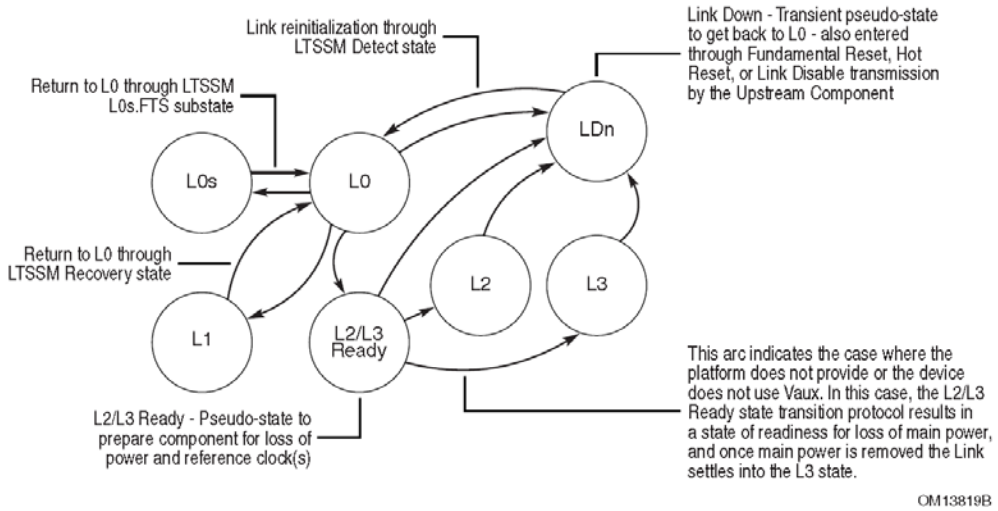


Figure 5-1: Link Power Management State Flow Diagram

The L1 and L2/L3 Ready entry negotiations happen while in the L0 state. L1 and L2/L3 Ready are entered only after the negotiation completes. Link Power Management remains in L0 until the negotiation process is completed, unless LDn occurs. Note that these states and state transitions do not correspond directly to the actions of the Physical Layer LTSSM. For example in Figure 5-1, L0 encompasses the LTSSM L0, Recovery, and, during LinkUp, Configuration states. Also, the LTSSM is typically powered by main power (not Vaux), so LTSSM will not be powered in either the L2 or the L3 state.

The following example sequence illustrates the multi-step Link state transition process leading up to entering a system sleep state:

1. System software directs all Functions of a Downstream component to $D3_{hot}$.
2. The Downstream component then initiates the transition of the Link to L1 as required.
3. System software then causes the Root Complex to broadcast the PME_Turn_Off Message in preparation for removing the main power source.
4. This Message causes the subject Link to transition back to L0 in order to send it and to enable the Downstream component to respond with PME_TO_Ack.
5. After sending the PME_TO_Ack, the Downstream component initiates the L2/L3 Ready transition protocol.

L0 → L1 → L0 → L2/L3 Ready

As the following example illustrates, it is also possible to remove power without first placing all Functions into $D3_{hot}$:

1. System software causes the Root Complex to broadcast the PME_Turn_Off Message in preparation for removing the main power source.
2. The Downstream components respond with PME_TO_Ack.

3. After sending the PME_TO_Ack, the Downstream component initiates the L2/L3 Ready transition protocol.

L0 → L2/L3 Ready

The L1 entry negotiation (whether invoked via PCI-PM or ASPM mechanisms) and the L2/L3 Ready entry negotiation map to a state machine which corresponds to the actions described later in this chapter. This state machine is reset to an idle state. For a Downstream component, the first action taken by the state machine, after leaving the idle state, is to start sending the appropriate entry DLLPs depending on the type of negotiation. If the negotiation is interrupted, for example by a trip through Recovery, the state machine in both components is reset back to the idle state. The Upstream component must always go to the idle state, and wait to receive entry DLLPs. The Downstream componentmust always go to the idle state and must always proceed to sending entry DLLPs to restart the negotiation.

Table 5-1 summarizes each L-state, describing when they are used, and the platform and component behaviors that correspond to each.

A "Yes" entry indicates that support is required (unless otherwise noted). "On" and "Off" entries indicate the required clocking and power delivery. "On/Off" indicates an optional design choice.

Table 5-1: Summary of PCI Express Link Power Management States

	L-State Description	Used by S/W Directed PM	Used by ASPM	Platform Reference Clocks	Platform Main Power	Component Internal PLL	Platform Vaux
L0	Fully active Link	Yes (D0)	Yes (D0)	On	On	On	On/Off
L0s	Standby state	No	Yes [1] (opt., D0)	On	On	On	On/Off
L1	Lower power standby	Yes (D1-$D3_{hot}$)	Yes (opt., D0)	On/Off [6]	On	On/Off [2]	On/Off
L2/L3 Ready (pseudo-state)	Staging point for power removal	Yes [3]	No	On/Off [6]	On	On/Off	On/Off
L2	Low power sleep state (all clocks, main power off)	Yes [4]	No	Off	Off	Off	On [5]
L3	Off (zero power)	n/a	n/a	Off	Off	Off	Off
LDn (pseudo-state)	Transitional state preceding L0	Yes	N/A	On	On	On/Off	On/Off

Notes:

1. L0s exit latency will be greatest in Link configurations with independent reference clock inputs for components connected to opposite ends of a given Link (vs. a common, distributed reference clock).
2. L1 exit latency will be greatest for components that internally shut off their PLLs during this state.
3. L2/L3 Ready entry sequence is initiated at the completion of the PME_Turn_Off/PME_TO_Ack protocol handshake. It is not directly affiliated with either a D-State transition or a transition in accordance with ASPM policies and procedures.
4. Depending upon the platform implementation, the system's sleep state may use the L2 state, transition to fully off (L3), or it may leave Links in the L2/L3 Ready state. L2/L3 Ready state transition protocol is initiated by the Downstream component following reception and TLP acknowledgement of the PME_Turn_Off TLP Message. While platform support for an L2 sleep state configuration is optional (depending on the availability of Vaux), component protocol support for transitioning the Link to the L2/L3 Ready state is required.
5. L2 is distinguished from the L3 state only by the presence and use of Vaux. After the completion of the L2/L3 Ready state transition protocol and before main power has been removed, the Link has indicated its readiness for main power removal.
6. Low-power mobile or handheld devices may reduce power by clock gating the reference clock(s) via the "clock request" (CLKREQ#) mechanism. As a result, components targeting these devices should be tolerant of the additional delays required to re-energize the reference clock during the low-power state exit.

5.3 PCI-PM Software Compatible Mechanisms

5.3.1 Device Power Management States (D-States) of a Function

While the concept of these power states is universal for all Functions in the system, the meaning, or intended functional behavior when transitioned to a given power management state, is dependent upon the type (or class) of the Function.

The D0 power management state is the normal operation state of the Function. Other states are various levels of reduced power, where the Function is either not operating or supports a limited set of operations. D1 and D2 are intermediate states that are intended to afford the system designer more flexibility in balancing power savings, restore time, and low power feature availability tradeoffs for a given device class. The D1 state could, for example, be supported as a slightly more power consuming state than D2, however one that yields a quicker restore time than could be realized from D2.

The D3 power management state constitutes a special category of power management state in that a Function could be transitioned into D3 either by software or by physically removing its power. In that sense, the two D3 variants have been designated as $D3_{hot}$ and $D3_{cold}$ where the subscript refers to the presence or absence of main power respectively. Functions in $D3_{hot}$ are permitted to be transitioned to the D0 state via software by writing to the Function's PMCSR register. Functions in the $D3_{cold}$ state are permitted to be transitioned to the $D0_{uninitialized}$ state by reapplying main power and asserting Fundamental Reset.

All Functions must support the D0 and D3 states (both $D3_{hot}$ and $D3_{cold}$). The D1 and D2 states are optional.

IMPLEMENTATION NOTE

Switch and Root Port Virtual Bridge Behavior in Non-D0 States

When a Type 1 Function associated with a Switch/Root Port (a "virtual bridge") is in a non-D0 power state, it will emulate the behavior of a conventional PCI bridge in its handling of Memory, I/O, and Configuration Requests and Completions. All Memory and I/O requests flowing Downstream are terminated as Unsupported Requests. All Type 1 Configuration Requests are terminated as Unsupported Requests, however Type 0 Configuration Request handling is unaffected by the virtual bridge D state. Completions flowing in either direction across the virtual bridge are unaffected by the virtual bridge D state.

Note that the handling of Messages is not affected by the PM state of the virtual bridge.

5.3.1.1 *D0 State*

All Functions must support the D0 state. D0 is divided into two distinct substates, the "un-initialized" substate and the "active" substate. When a component comes out of Conventional Reset all Functions of the component enter the $D0_{uninitialized}$ state. When a Function completes FLR, it enters the $D0_{uninitialized}$ state. After configuration is complete a Function enters the $D0_{active}$ state, the

fully operational state for a PCI Express Function. A Function enters the $D0_{active}$ state whenever any single or combination of the Function's Memory Space Enable, I/O Space Enable, or Bus Master Enable bits have been Set[68].

5.3.1.2 *D1 State*

D1 support is optional. While in the D1 state, a Function must not initiate any Request TLPs on the Link with the exception of a PME Message as defined in Section 1.1.1. Configuration and Message Requests are the only TLPs accepted by a Function in the D1 state. All other received Requests must be handled as Unsupported Requests, and all received Completions may optionally be handled as Unexpected Completions. If an error caused by a received TLP (e.g., an Unsupported Request) is detected while in D1, and reporting is enabled, the Link must be returned to L0 if it is not already in L0 and an error Message must be sent. If an error caused by an event other than a received TLP (e.g., a Completion Timeout) is detected while in D1, an error Message must be sent when the Function is programmed back to the D0 state.

Note that a Function's software driver participates in the process of transitioning the Function from D0 to D1. It contributes to the process by saving any functional state (if necessary), and otherwise preparing the Function for the transition to D1. As part of this quiescence process the Function's software driver must ensure that any mid-transaction TLPs (i.e., Requests with outstanding Completions), are terminated prior to handing control to the system configuration software that would then complete the transition to D1.

5.3.1.3 *D2 State*

D2 support is optional. When a Function is not currently being used and probably will not be used for some time, it may be put into D2. This state requires the Function to provide significant power savings while still retaining the ability to fully recover to its previous condition. While in the D2 state, a Function must not initiate any Request TLPs on the Link with the exception of a PME Message as defined in Section 1.1.1. Configuration and Message requests are the only TLPs accepted by a Function in the D2 state. All other received Requests must be handled as Unsupported Requests, and all received Completions may optionally be handled as Unexpected Completions. If an error caused by a received TLP (e.g., an Unsupported Request) is detected while in D2, and reporting is enabled, the Link must be returned to L0 if it is not already in L0 and an error Message must be sent. If an error caused by an event other than a received TLP (e.g., a Completion Timeout) is detected while in D2, an error Message must be sent when the Function is programmed back to the D0 state.

Note that a Function's software driver participates in the process of transitioning the Function from D0 to D2. It contributes to the process by saving any functional state (if necessary), and otherwise preparing the Function for the transition to D2. As part of this quiescence process the Function's software driver must ensure that any mid-transaction TLPs (i.e., Requests with outstanding Completions), are terminated prior to handing control to the system configuration software that would then complete the transition to D2.

---

[68] A Function remains in $D0_{active}$ even if these enable bits are subsequently cleared.

System software must restore the Function to the $D0_{active}$ state before memory or I/O space can be accessed. Initiated actions such as bus mastering and interrupt request generation can only commence after the Function has been restored to $D0_{active}$.

There is a minimum recovery time requirement of 200 µs between when a Function is programmed from D2 to D0 and the next Request issued to the Function. Behavior is undefined for Requests received in this recovery time window (see Section 7.9.17).

5.3.1.4 *D3 State*

D3 support is required, (both the $D3_{cold}$ and the $D3_{hot}$ states).

Functional context is required to be maintained by Functions in the $D3_{hot}$ state if the No_Soft_Reset field in the PMCSR is Set. In this case, System Software is not required to re-initialize the Function after a transition from $D3_{hot}$ to D0 (the Function will be in the $D0_{active}$ state). If the No_Soft_Reset bit is Clear, functional context is not required to be maintained by the Function in the $D3_{hot}$ state, however it is not guaranteed that functional context will be cleared and software must not depend on such behavior. As a result, in this case System Software is required to fully re-initialize the Function after a transition to D0 as the Function will be in the $D0_{uninitialized}$ state.

The Function will be reset if the Link state has transitioned to the L2/L3 Ready state regardless of the value of the No_Soft_Reset bit.

IMPLEMENTATION NOTE

Transitioning to L2/L3 Ready

As described in Section 5.2, transition to the L2/L3 Ready state is initiated by platform power management software in order to begin the process of removing main power and clocks from the device. As a result, it is expected that a device will transition to $D3_{cold}$ shortly after its Link transitions to L2/L3 Ready, making the No_Soft_Reset bit, which only applies to $D3_{hot}$, irrelevant. While there is no guarantee of this correlation between L2/L3 Ready and $D3_{cold}$, system software should ensure that the L2/L3 Ready state is entered only when the intent is to remove device main power. Device Functions, including those that are otherwise capable of maintaining functional context while in $D3_{hot}$ (i.e., set the No_Soft_Reset bit), are required to re-initialize internal state as described in Section 2.9.1 when exiting L2/L3 Ready due to the required DL_Down status indication.

Unless the Immediate_Readiness_on_Return_to_D0 bit in the PCI-PM Power Management Capabilities register is Set, System Software must allow a minimum recovery time following a $D3_{hot} \rightarrow D0$ transition of at least 10 ms (see Section7.9.17), prior to accessing the Function. This recovery time may, for example, be used by the $D3_{hot} \rightarrow D0$ transitioning component to bootstrap any of its component interfaces (e.g., from serial ROM) prior to being accessible. Attempts to target the Function during the recovery time (including configuration request packets) will result in undefined behavior.

5.3.1.4.1 $D3_{hot}$ State

Configuration and Message requests are the only TLPs accepted by a Function in the $D3_{hot}$ state. All other received Requests must be handled as Unsupported Requests, and all received Completions may optionally be handled as Unexpected Completions. If an error caused by a received TLP (e.g., an Unsupported Request) is detected while in $D3_{hot}$, and reporting is enabled, the Link must be returned to L0 if it is not already in L0 and an error Message must be sent. If an error caused by an event other than a received TLP (e.g., a Completion Timeout) is detected while in $D3_{hot}$, an error Message may optionally be sent when the Function is programmed back to the D0 state. Once in $D3_{hot}$ the Function can later be transitioned into $D3_{cold}$ (by removing power from its host component).

Note that a Function's software driver participates in the process of transitioning the Function from D0 to $D3_{hot}$. It contributes to the process by saving any functional state that would otherwise be lost with removal of main power, and otherwise preparing the Function for the transition to $D3_{hot}$. As part of this quiescence process the Function's software driver must ensure that any outstanding transactions (i.e., Requests with outstanding Completions), are terminated prior to handing control to the system configuration software that would then complete the transition to $D3_{hot}$.

Note that $D3_{hot}$ is also a useful state for reducing power consumption by idle components in an otherwise running system.

Functions that are in $D3_{hot}$ are permitted to be transitioned by software (writing to their PMCSR PowerState field) to the $D0_{active}$ state or the $D0_{uninitialized}$ state. Functions that are in $D3_{hot}$ must respond to Configuration Space accesses as long as power and clock are supplied so that they can be returned to D0 by software. Note that the Function is not required to generate an internal hardware reset during or immediately following its transition from $D3_{hot}$ to D0 (see usage of the No_Soft_Reset bit in the PMCSR).

If not requiring an internal reset, upon completion of the $D3_{hot}$ to $D0_{initialized}$ state, no additional operating system intervention is required beyond writing the PowerState field. If the internal reset is required, devices return to $D0_{uninitialized}$ and a full reinitialization is required on the device. The full reinitialization sequence returns the device to $D0_{initialized}$.

If the device supports PME events, and PME_En is Set, PME context must be preserved in $D3_{hot}$. PME context must also be preserved in a PowerState command transition back to D0.

IMPLEMENTATION NOTE

Devices Not Performing an Internal Reset

Bus controllers to non-PCIe buses and resume from $D3_{hot}$ bus controllers on PCIe buses that serve as interfaces to non-PCIe buses, (e.g., CardBus, USB, and IEEE 1394) are examples of bus controllers that would benefit from not requiring an internal reset upon resume from $D3_{hot}$. If this internal reset is not required, the bus controller would not need to perform a downstream bus reset upon resume from $D3_{hot}$ on its secondary (non-PCIe) bus.

IMPLEMENTATION NOTE

Multi-Function Device Issues with Soft Reset

With Multi-Function Devices (MFDs), certain control settings affecting overall device behavior are determined either by the collective settings in all Functions or strictly off the settings in Function 0. Here are some key examples:

- With non-ARI MFDs, certain controls in the Device Control register and Link Control registers operate off the collective settings of all Functions (see Section 7.5.3.4 and Section 7.5.3.7).
- With ARI Devices, certain controls in the Device Control register and Link Control registers operate strictly off the settings in Function 0 (see Section 7.5.3.4 and Section 7.5.3.7).
- With all MFDs, certain controls in the Device Control 2 and Link Control 2 registers operate strictly off the settings in Function 0 (see Section 7.5.3.16 and Section 7.5.3.19).

Performing a soft reset on any Function (especially Function 0) may disrupt the proper operation of other active Functions in the MFD. Since some Operating Systems transition a given Function between $D3_{hot}$ and D0 with the expectation that other Functions will not be impacted, it is strongly recommended that every Function in an MFD be implemented with the No_Soft_Reset bit Set in the Power Management Control/Status register. This way, transitioning a given Function from $D3_{hot}$ to D0 will not disrupt the proper operation of other active Functions.

It is also strongly recommended that every Endpoint Function in an MFD implement Function Level Reset (FLR). FLR can be used to reset an individual Endpoint Function without impacting the settings that might affect other Functions, particularly if those Functions are active. As a result of FLR's quiescing, error recovery, and cleansing for reuse properties, FLR is also recommended for single-Function Endpoint devices.

5.3.1.4.2 $D3_{cold}$ State

A Function transitions to the $D3_{cold}$ state when its main power is removed. A power-on sequence with its associated cold reset transitions a Function from the $D3_{cold}$ state to the $D0_{uninititialized}$ state, and the power-on defaults will be restored to the Function by hardware just as at initial power up. At this point, software must perform a full initialization of the Function in order to re-establish all functional context, completing the restoration of the Function to its $D0_{active}$ state.

Functions that support wakeup functionality from $D3_{cold}$ must maintain their PME context (in the PMCSR), When PME_En is Set, for inspection by PME service routine software during the course of the resume process. Retention of additional context is implementation specific.

IMPLEMENTATION NOTE

PME Context

Examples of PME context include, but are not limited to, a Function's PME_Status bit, the requesting agent's Requester ID, Caller ID if supported by a modem, IP information for IP directed network packets that trigger a resume event, etc.

A Function's PME assertion is acknowledged when system software performs a "write 1 to clear" configuration transaction to the asserting Function's PME_Status bit of its PCI-PM compatible PMCSR.

An auxiliary power source must be used to support PME event detection within a Function, Link reactivation, and to preserve PME context from within $D3_{cold}$. Note that once the I/O Hierarchy has been brought back to a fully communicating state, as a result of the Link reactivation, the waking agent then propagates a PME Message to the root of the Hierarchy indicating the source of the PME event. Refer to Section 1.1.1 for further PME specific detail.

5.3.2 PM Software Control of the Link Power Management State

The power management state of a Link is determined by the D-state of its Downstream component.

Table 5-2 depicts the relationships between the power state of a component (with an Upstream Port) and its Upstream Link.

Table 5-2: Relation Between Power Management States of Link and Components

Downstream Component D-State	Permissible Upstream Component D-State	Permissible Interconnect State
D0	D0	L0, L0s, L1[(1)], L2/L3 Ready
D1	D0-D1	L1, L2/L3 Ready
D2	D0-D2	L1, L2/L3 Ready
$D3_{hot}$	D0-$D3_{hot}$	L1, L2/L3 Ready
$D3_{cold}$	D0-$D3_{cold}$	L2[(2)], L3

Notes:

1. Requirements for ASPM L0s and ASPM L1 support are form factor specific.
2. If Vaux is provided by the platform, the Link sleeps in L2. In the absence of Vaux, the L-state is L3.

The following rules relate to PCI-PM compatible power management:

- Devices in D0, D1, D2, and $D3_{hot}$ must respond to the receipt of a PME_Turn_Off Message by the transmission of a PME_TO_Ack Message.
- In any device D state, following the execution of a PME_Turn_Off/PME_TO_Ack handshake sequence, a Downstream component must request a Link transition to L2/L3 Ready using the PM_Enter_L23 DLLP. Following the L2/L3 Ready entry transition protocol the Downstream component must be ready for loss of main power and reference clock.
- The Upstream Port of a single-Function device must initiate a Link state transition to L1 based solely upon its Function being programmed to D1, D2, or $D3_{hot}$. In the case of the Switch, system software bears the responsibility of ensuring that any D-state programming of a Switch's Upstream Port is done in a compliant manner with respect to hierarchy-wide PM policies (i.e., the Upstream Port cannot be programmed to a D-state that is any less active than the most active Downstream Port and Downstream connected component/Function(s)).

- The Upstream Port of a non-ARI Multi-Function Device must not initiate a Link state transition to L1 (on behalf of PCI-PM) until all of its Functions have been programmed to a non-D0 D-state.
- The Upstream Port of an ARI Device must not initiate a Link state transition to L1 (on behalf of PCI-PM) until at least one of its Functions has been programmed to a non-D0 state, and all of its Functions are either in a non-D0 state or the $D0_{uninitialized}$ state.

5.3.2.1 *Entry into the L1 State*

Figure 5-2 depicts the process by which a Link transitions into the L1 state as a direct result of power management software programming the Downstream connected component into a lower power state, (either D1, D2, or $D3_{hot}$ state). This figure and the subsequent description outline the transition process for a single -Function Downstream component that is being programmed to a non-D0 state.

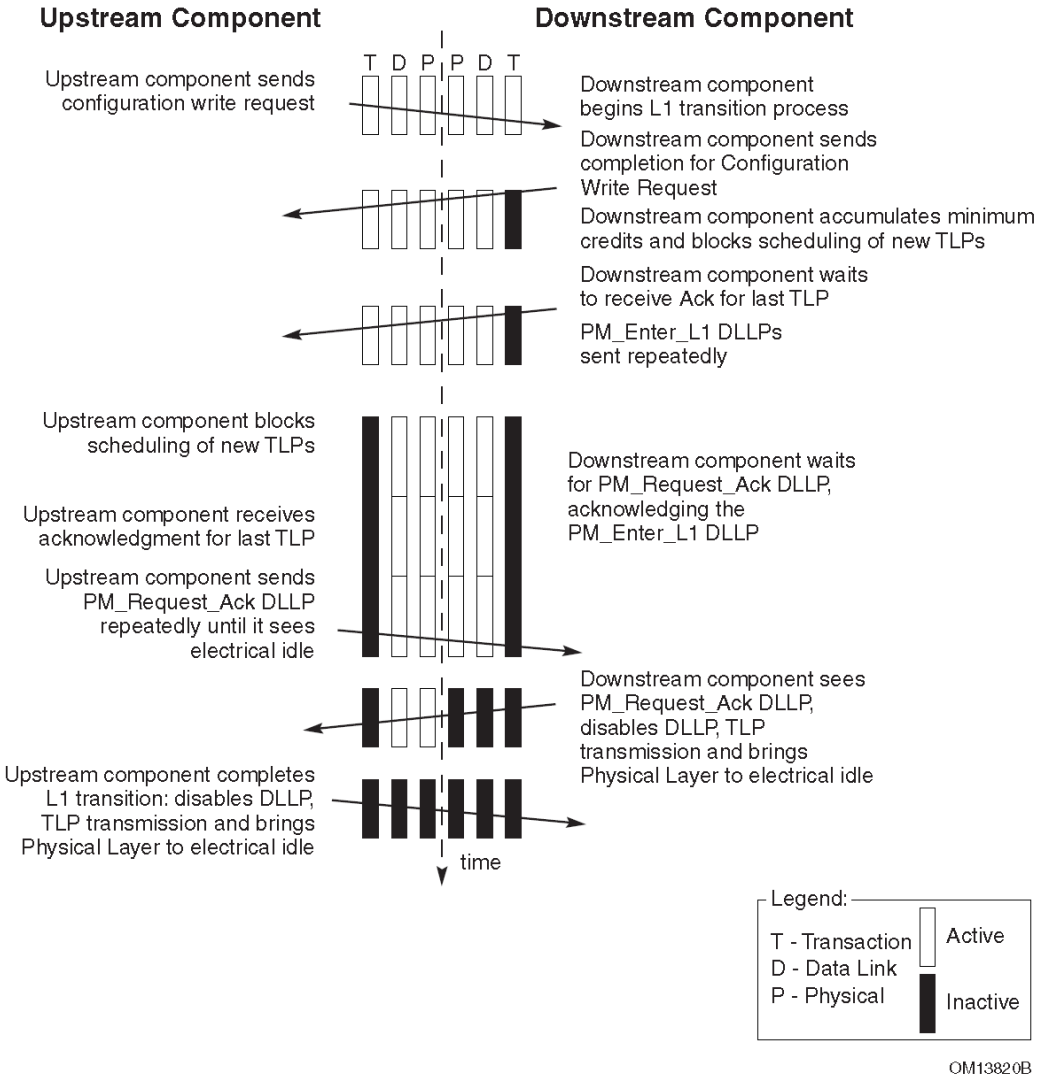


Figure 5-2: Entry into the L1 Link State

The following text provides additional detail for the Link state transition process shown in Figure 5-2.

PM Software Request:

1. PM software sends a Configuration Write Request TLP to the Downstream Function's PMCSR to change the Downstream Function's D-state (from D0 to D1 for example).

Downstream Component Link State Transition Initiation Process:

2. The Downstream component schedules the Completion corresponding to the Configuration Write Request to its PMCSR PowerState field and accounts for the completion credits required.
3. The Downstream component must then wait until it accumulates at least the minimum number of credits required to send the largest possible packet for any FC type for all enabled VCs (if it does not already have such credits). All Transaction Layer TLP scheduling is then suspended.
4. The Downstream component then waits until it receives a Link Layer acknowledgement for the PMCSR Write Completion, and any other TLPs it had previously sent. The component must retransmit a TLP out of its Data Link Layer Retry buffer if required to do so by Data Link Layer rules.
5. Once all of the Downstream components' TLPs have been acknowledged, the Downstream component starts to transmit PM_Enter_L1 DLLPs. The Downstream component sends this DLLP repeatedly with no more than eight (when using 8b/10b encoding) or 32 (when using 128b/130b encoding) Symbol times of idle between subsequent transmissions of the PM_Enter_L1 DLLP. The transmission of other DLLPs and SKP Ordered Sets is permitted at any time between PM_Enter_L1 transmissions, and do not contribute to this idle time limit.

   The Downstream component continues to transmit the PM_Enter_L1 DLLP as described above until it receives a response from the Upstream component[69] (PM_Request_Ack).

   The Downstream component must continue to accept TLPs and DLLPs from the Upstream component, and continue to respond with DLLPs, including FC update DLLPs and Ack/Nak DLLPs, as required. Any TLPs that are blocked from transmission (including responses to TLP(s) received) must be stored for later transmission, and must cause the Downstream component to initiate L1 exit as soon as possible following L1 entry.

Upstream Component Link State Transition Process:

6. Upon receiving the PM_Enter_L1 DLLP, the Upstream component blocks the scheduling of all TLP transmissions.
7. The Upstream component then must wait until it receives a Link Layer acknowledgement for the last TLP it had previously sent. The Upstream component must retransmit a TLP from its Link Layer retry buffer if required to do so by the Link Layer rules.
8. Once all of the Upstream component's TLPs have been acknowledged, the Upstream component must send PM_Request_Ack DLLPs Downstream, regardless of any outstanding Requests. The Upstream component sends this DLLP repeatedly with no more than eight

[69] If at this point the Downstream component needs to initiate a transfer on the Link, it must first complete the transition to L1. Once in L1 it is then permitted to initiate an exit L1 to handle the transfer.

(when using 8b/10b encoding) or 32 (when using 128b/130b encoding) Symbol times of idle between subsequent transmissions of the PM_Request_Ack DLLP. The transmission of SKP Ordered Sets is permitted at any time between PM_Request_Ack transmissions, and does not contribute to this idle time limit.

The Upstream component continues to transmit the PM_Request_Ack DLLP as described above until it observes its receive Lanes enter into the Electrical Idle state. Refer to Chapter 4 for more details on the Physical Layer behavior.

Completing the L1 Link State Transition:

9. Once the Downstream component has captured the PM_Request_Ack DLLP on its Receive Lanes (signaling that the Upstream component acknowledged the transition to L1 request), it then disables DLLP transmission and brings the Upstream directed physical Link into the Electrical Idle state.
10. When the Receive Lanes on the Upstream component enter the Electrical Idle state, the Upstream component stops sending PM_Request_Ack DLLPs, disables DLLP transmission, and brings its Transmit Lanes to Electrical Idle completing the transition of the Link to L1.

    When two components' interconnecting Link is in L1 as a result of the Downstream component being programmed to a non-D0 state, both components suspend the operation of their Flow Control Update and, if implemented, Update FCP Timer (see Section 2.6.1.2) counter mechanisms. Refer to Chapter 4 for more detail on the Physical Layer behavior.

Refer to Section 5.2 if the negotiation to L1 is interrupted.

Components on either end of a Link in L1 may optionally disable their internal PLLs in order to conserve more energy. Note, however, that platform supplied main power and reference clocks must continue to be supplied to components on both ends of an L1 Link in the L1.0 substate of L1.

Refer to Section 5.5 for entry into the L1 PM Substates.

5.3.2.2 *Exit from L1 State*

L1 exit can be initiated by the component on either end of a Link.

Upon exit from L1, it is recommended that the Downstream component send flow control update DLLPs for all enabled VCs and FC types starting within 1 µs of L1 exit.

The physical mechanism for transitioning a Link from L1 to L0 is described in detail in Chapter 4.

L1 exit must be initiated by a component if that component needs to transmit a TLP on the Link. An Upstream component must initiate L1 exit on a Downstream Port even if it does not have the flow control credits needed to transmit the TLP that it needs to transmit. Following L1 exit, the Upstream component must wait to receive the needed credit from the Downstream component. Figure 5-3 outlines an example sequence that would trigger an Upstream component to initiate transition of the Link to the L0 state.

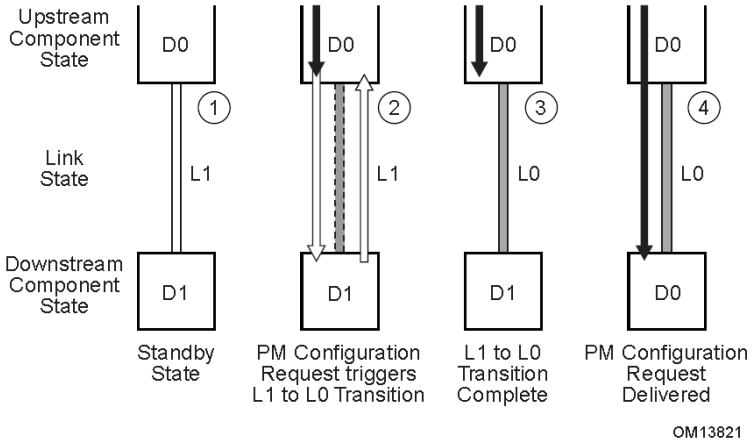


Figure 5-3: Exit from L1 Link State Initiated by Upstream Component

Sequence of events:

1. Power management software initiates a configuration cycle targeting a PM configuration register (the PowerState field of the PMCSR in this example) within a Function that resides in the Downstream component (e.g., to bring the Function back to the D0 state).
2. The Upstream component detects that a configuration cycle is intended for a Link that is currently in a low power state, and as a result, initiates a transition of that Link into the L0 state.
3. If the Link is in either L1.1 or L1.2 substates of L1, then the Upstream component initiates a transition of the Link into the L1.0 substate of L1.
4. In accordance with the Chapter 4 definition, both directions of the Link enter into Link training, resulting in the transition of the Link to the L0 state. The L1 → L0 transition is discussed in detail in Chapter 4.
5. Once both directions of the Link are back to the active L0 state, the Upstream Port sends the configuration Packet Downstream.

5.3.2.3 *Entry into the L2/L3 Ready State*

Transition to the L2/L3 Ready state follows a process that is similar to the L1 entry process. There are some minor differences between the two that are spelled out below.

- L2/L3 Ready entry transition protocol does not immediately result in an L2 or L3 Link state. The transition to L2/L3 Ready is effectively a handshake to establish the Downstream component's readiness for power removal. L2 or L3 is ultimately achieved when the platform removes the components' power and reference clock.
- The time for L2/L3 Ready entry transition is indicated by the completion of the PME_Turn_Off/PME_TO_Ack handshake sequence. Any actions on the part of the Downstream component necessary to ready itself for loss of power must be completed prior to initiating the transition to L2/L3 Ready. Once all preparations for loss of power and clock are completed, L2/L3 Ready entry is initiated by the Downstream component by sending the PM_Enter_L23 DLLP Upstream.

- L2/L3 Ready entry transition protocol uses the PM_Enter_L23 DLLP.

Note that the PM_Enter_L23 DLLPs are sent continuously until an acknowledgement is received or power is removed.

- Refer to Section 5.2 if the negotiation to L2/L3 Ready is interrupted.

5.3.3 Power Management Event Mechanisms

5.3.3.1 *Motivation*

The PCI Express PME mechanism is software compatible with the PME mechanism defined by the *PCI Bus Power Management Interface Specification*. Power Management Events are generated by Functions as a means of requesting a PM state change. Power Management Events are typically utilized to revive the system or an individual Function from a low power state.

Power management software may transition a Hierarchy into a low power state, and transition the Upstream Links of these devices into the non-communicating L2 state.[70] The PCI Express PME generation mechanism is, therefore, broken into two components:

- Waking a non-communicating Hierarchy (wakeup). This step is required only if the Upstream Link of the device originating the PME is in the non-communicating L2 state, since in that state the device cannot send a PM_PME Message Upstream.
- Sending a PM_PME Message to the root of the Hierarchy

PME indications that originate from PCI Express Endpoints or PCI Express Legacy Endpoints are propagated to the Root Complex in the form of TLP messages. PM_PME Messages identify the requesting agent within the Hierarchy (via the Requester ID of the PME Message header). Explicit identification within the PM_PME Message is intended to facilitate quicker PME service routine response, and hence shorter resume time.

If a Root Complex Event Collector is implemented, PME indications that originate from a Root Complex Integrated Endpoint (RCiEP) may optionally be reported in a Root Complex Event Collector residing on the same Logical Bus as the RCiEP. The Root Complex Event Collector must explicitly declare supported RCiEPs as part of its capabilities; each RCiEP must be associated with no more than one Root Complex Event Collector. Root Complex Event Collectors explicitly identify the logical location of the requesting agent to facilitate quicker PME service routine response.

PME indications that originate from a Root Port itself are reported through the same Root Port.

5.3.3.2 *Link Wakeup*

The Link wakeup mechanisms provide a means of signaling the platform to re-establish power and reference clocks to the components within its domain. There are two defined wakeup mechanisms: Beacon and WAKE#. The Beacon mechanism uses in-band signaling to implement wakeup functionality. For components that support wakeup functionality, the form factor specification(s)

[70] The L2 state is defined as “non-communicating” since component reference clock and main power supply are removed in that state.

targeted by the implementation determine the support requirements for the wakeup mechanism. Switch components targeting applications where Beacon is used on some Ports of the Switch and WAKE# is used for other Ports must translate the wakeup mechanism appropriately (see the implementation note entitled "Example of WAKE# to Beacon Translation"in Section, 5.3.3.2). In applications where WAKE# is the only wakeup mechanism used, the Root Complex is not required to support the receipt of Beacon.

The WAKE# mechanism uses sideband signaling to implement wakeup functionality. WAKE# is an "open drain" signal asserted by components requesting wakeup and observed by the associated power controller. WAKE# is only defined for certain form factors, and the detailed specifications for WAKE# are included in the relevant form factor specifications. Specific form factor specifications may require the use of either Beacon or WAKE# as the wakeup mechanism.

When WAKE# is used as a wakeup mechanism, once WAKE# has been asserted, the asserting Function must continue to drive the signal low until main power has been restored to the component as indicated by Fundamental Reset going inactive.

The system is not required to route or buffer WAKE# in such a way that an Endpoint is guaranteed to be able to detect that the signal has been asserted by another Function.

Before using any wakeup mechanism, a Function must be enabled by software to do so by setting the Function's PME_En bit in the PMCSR. The PME_Status bit is sticky, and Functions must maintain the value of the PME_Status bit through reset if Aux power is available and they are enabled for wakeup events (this requirement also applies to the PME_En bit in the PMCSR and the Aux Power PM Enable bit in the Device Control register).

Systems that allow PME generation from $D3_{cold}$ state must provide auxiliary power to support Link wakeup when the main system power rails are off. A component may only consume auxiliary power if software has enabled it to do so as described in Section 5.6. Software is required to enable auxiliary power consumption in all components that participate in Link wakeup, including all components that must propagate the Beacon signal. In the presence of legacy system software, this is the responsibility of system firmware.

Regardless of the wakeup mechanism used, once the Link has been re-activated and trained, the requesting agent then propagates a PM_PME Message Upstream to the Root Complex. From a power management point of view, the two wakeup mechanisms provide the same functionality, and are not distinguished elsewhere in this chapter.

IMPLEMENTATION NOTE

Example of WAKE# to Beacon Translation

Switch components targeting applications that connect "Beacon domains" and "WAKE# domains" must translate the wakeup mechanism appropriately. Figure 5-4 shows two example systems, each including slots that use the WAKE# wakeup mechanism. In Case 1, WAKE# is input directly to the Power Management Controller, and no translation is required. In Case 2, WAKE# is an input to the Switch, and in response to WAKE# being asserted the Switch must generate a Beacon that is propagated to the Root Complex/Power Management Controller.

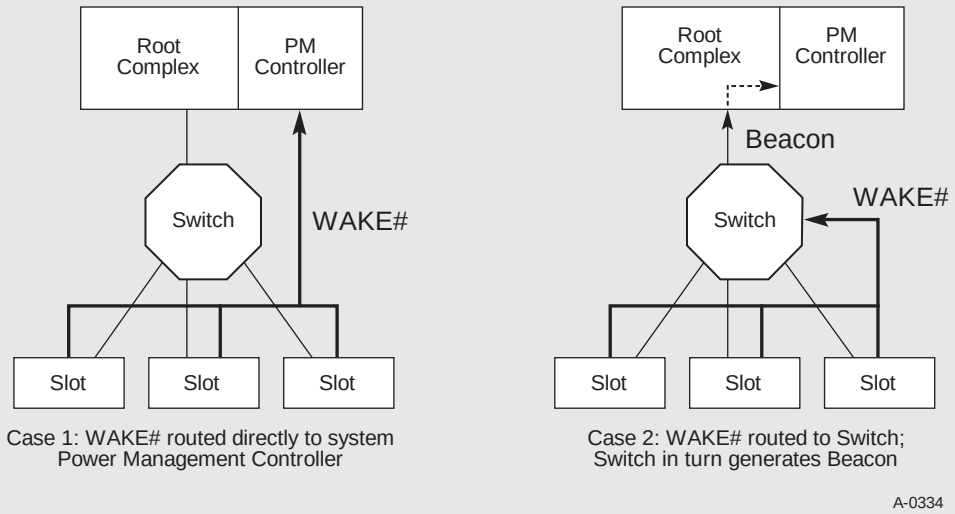


Figure 5-4: Conceptual Diagrams Showing Two Example Cases of WAKE# Routing

5.3.3.2.1 PME Synchronization

PCI Express-PM introduces a fence mechanism that serves to initiate the power removal sequence while also coordinating the behavior of the platform's power management controller and PME handling by PCI Express agents.

PME_Turn_Off Broadcast Message

Before main component power and reference clocks are turned off, the Root Complex or Switch Downstream Port must issue a broadcast Message that instructs all agents Downstream of that point within the hierarchy to cease initiation of any subsequent PM_PME Messages, effective immediately upon receipt of the PME_Turn_Off Message.

Each PCI Express agent is required to respond with a TLP "acknowledgement" Message, PME_TO_Ack that is always routed Upstream. In all cases, the PME_TO_Ack Message must terminate at the PME_Turn_Off Message's point of origin.[71]

[71] Point of origin for the PME_Turn_Off Message could be all of the Root Ports for a given Root Complex (full platform sleep state transition), an individual Root Port, or a Switch Downstream Port.

A Switch must report an "aggregate" acknowledgement only after having received PME_TO_Ack Messages from each of its Downstream Ports. Once a PME_TO_Ack Message has arrived on each Downstream Port, the Switch must then send a PME_TO_Ack packet on its Upstream Port. The occurrence of any one of the following must reset the aggregation mechanism: the transmission of the PME_TO_Ack Message from the Upstream Port, the receipt of any TLP at the Upstream Port, the removal of main power to the Switch, or Fundamental Reset.

All components with an Upstream Port must accept and acknowledge the PME_Turn_Off Message regardless of the D state of the associated device or any of its Functions for a Multi-Function Device. Once a component has sent a PME_TO_Ack Message, it must then prepare for removal of its power and reference clocks by initiating a transition to the L2/L3 Ready state.

A Switch must transition its Upstream Link to the L2/L3 Ready state after all of its Downstream Ports have entered the L2/L3 Ready state.

The Links attached to the originator of the PME_Turn_Off Message are the last to assume the L2/L3 Ready state. This state transition serves as an indication to the power delivery manager[72] that all Links within that portion of the Hierarchy have successfully retired all in flight PME Messages to the point of PME_Turn_Off Message origin and have performed any necessary local conditioning in preparation for power removal.

In order to avoid deadlock in the case where one or more devices do not respond with a PME_TO_Ack Message and then put their Links into the L2/L3 Ready state, the power manager must implement a timeout after waiting for a certain amount of time, after which it proceeds as if the Message had been received and all Links put into the L2/L3 Ready state. The recommended limit for this timer is in the range of 1 ms to 10 ms.

The power delivery manager must wait a minimum of 100 ns after observing all Links corresponding to the point of origin of the PME_Turn_Off Message enter L2/L3 Ready before removing the components' reference clock and main power. This requirement does not apply in the case where the above mentioned timer triggers.

[72] Power delivery control within this context relates to control over the entire Link hierarchy, or over a subset of Links ranging down to a single Link and associated Endpoint for sub hierarchies supporting independently managed power and clock distribution.

IMPLEMENTATION NOTE

PME_TO_Ack Message Proxy by Switches

One of the PME_Turn_Off/PME_TO_Ack handshake's key roles is to ensure that all in flight PME Messages are flushed from the PCI Express fabric prior to sleep state power removal. This is guaranteed to occur because PME Messages and the PME_TO_Ack Messages both use the posted request queue within VC0 and so all previously injected PME Messages will be made visible to the system before the PME_TO_Ack is received at the Root Complex. Once all Downstream Ports of the Root Complex receive a PME_TO_Ack Message the Root Complex can then signal the power manager that it is safe to remove power without loss of any PME Messages.

Switches create points of hierarchical expansion and, therefore, must wait for all of their connected Downstream Ports to receive a PME_TO_Ack Message before they can send a PME_TO_Ack Message Upstream on behalf of the sub-hierarchy that it has created Downstream. This can be accomplished very simply using common score boarding techniques. For example, once a PME_Turn_Off broadcast Message has been broadcast Downstream of the Switch, the Switch simply checks off each Downstream Port having received a PME_TO_Ack. Once the last of its active Downstream Ports receives a PME_TO_Ack, the Switch will then send a single PME_TO_Ack Message Upstream as a proxy on behalf of the entire sub-hierarchy Downstream of it. Note that once a Downstream Port receives a PME_TO_Ack Message and the Switch has scored its arrival, the Port is then free to drop the packet from its internal queues and free up the corresponding posted request queue FC credits.

5.3.3.3 *PM_PME Messages*

PM_PME Messages are posted Transaction Layer Packets (TLPs) that inform the power management software which agent within the Hierarchy requests a PM state change. PM_PME Messages, like all other Power Management system Messages, must use the general purpose Traffic Class, TC0.

PM_PME Messages are always routed in the direction of the Root Complex. To send a PM_PME Message on its Upstream Link, a device must transition the Link to the L0 state (if the Link was not in that state already). Unless otherwise noted, the device will keep the Link in the L0 state following the transmission of a PM_PME Message.

5.3.3.3.1 PM_PME “Backpressure” Deadlock Avoidance

A Root Complex is typically implemented with local buffering to store temporarily a finite number of PM_PME Messages that could potentially be simultaneously propagating through the Hierarchy. Given a limited number of PM_PME Messages that can be stored within the Root Complex, there can be backpressure applied to the Upstream directed posted queue in the event that the capacity of this temporary PM_PME Message buffer is exceeded.

Deadlock can occur according to the following example scenario:

1. Incoming PM_PME Messages fill the Root Complex’s temporary storage to its capacity while there are additional PM_PME Messages still in the Hierarchy making their way Upstream.

2. The Root Complex, on behalf of system software, issues a Configuration Read Request targeting one of the PME requester's PMCSR (e.g., reading its PME_Status bit).
3. The corresponding split completion Packet is required, as per producer/consumer ordering rules, to push all previously posted PM_PME Messages ahead of it, which in this case are PM_PME Messages that have no place to go.
4. The PME service routine cannot make progress; the PM_PME Message storage situation does not improve.
5. Deadlock occurs.

Precluding potential deadlocks requires the Root Complex to always enable forward progress under these circumstances. This must be done by accepting any PM_PME Messages that posted queue flow control credits allow for, and discarding any PM_PME Messages that create an overflow condition. This required behavior ensures that no deadlock will occur in these cases; however, PM_PME Messages will be discarded and hence lost in the process.

To ensure that no PM_PME Messages are lost permanently, all agents that are capable of generating PM_PME must implement a PME Service Timeout mechanism to ensure that their PME requests are serviced in a reasonable amount of time.

If after 100 ms (+ 50%/- 5%), the PME_Status bit of a requesting agent has not yet been cleared, the PME Service Timeout mechanism expires triggering the PME requesting agent to re-send the temporarily lost PM_PME Message. If at this time the Link is in a non-communicating state, then, prior to re-sending the PM_PME Message, the agent must reactivate the Link as defined in Section 5.3.3.2.

5.3.3.4 *PME Rules*

- All device Functions must implement the PCI-PM Power Management Capabilities (PMC) register and the PMCSR in accordance with the PCI-PM specification. These registers reside in the PCI-PM compliant PCI Capability List format.
  - PME capable Functions must implement the PME_Status bit, and underlying functional behavior, in their PMCSR.
  - When a Function initiates Link wakeup, or issues a PM_PME Message, it must set its PME_Status bit.
- Switches must route a PM_PME received on any Downstream Port to their Upstream Port
- On receiving a PME_Turn_Off Message, the device must block the transmission of PM_PME Messages and transmit a PME_TO_Ack Message Upstream. The component is permitted to send a PM_PME Message after the Link is returned to an L0 state through LDn.
- Before a Link or a portion of a Hierarchy is transferred into a non-communicating state (i.e., a state from which it cannot issue a PM_PME Message), a PME_Turn_Off Message must be broadcast Downstream.

5.3.3.5 *PM_PME Delivery State Machine*

The following diagram conceptually outlines the PM_PME delivery control state machine. This state machine determines the ability of a Link to service PME events by issuing PM_PME immediately vs. requiring Link wakeup.

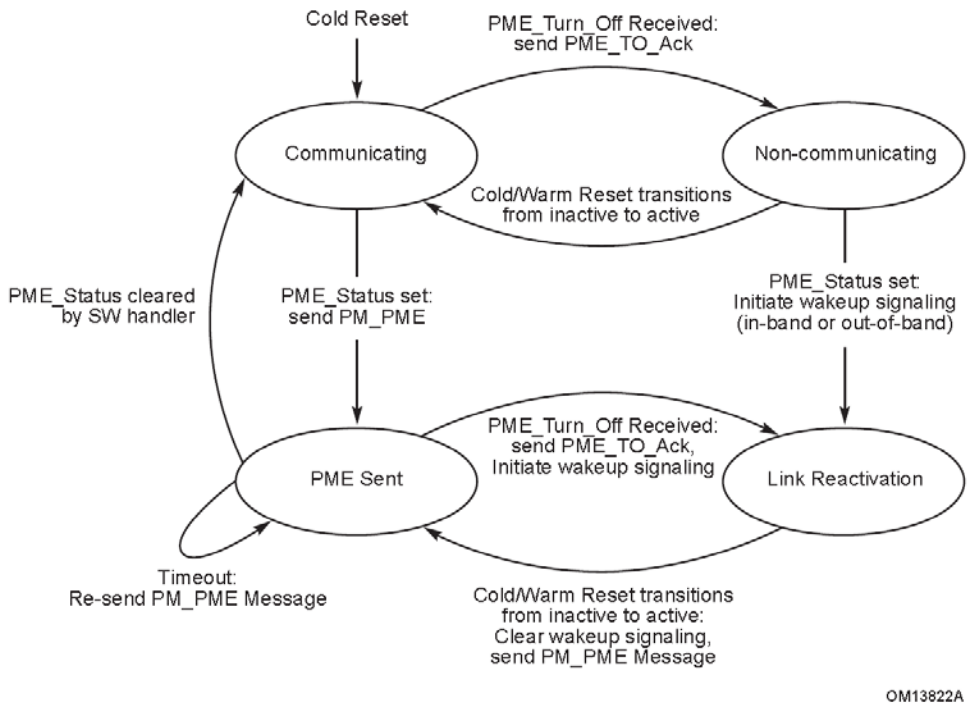


Figure 5-5: A Conceptual PME Control State Machine

Communicating State:

At initial power-up and associated reset, the Upstream Link enters the Communicating state

- If PME_Status is asserted (assuming PME delivery is enabled), a PM_PME Message will be issued Upstream, terminating at the root of the Hierarchy. The next state is the PME Sent state
- If a PME_Turn_Off Message is received, the Link enters the Non-communicating state following its acknowledgment of the Message and subsequent entry into the L2/L3 Ready state.

Non-communicating State:

- Following the restoration of power and clock, and the associated reset, the next state is the Communicating state.
- If PME_Status is asserted, the Link will transition to the Link Reactivation state, and activate the wakeup mechanism.

PME Sent State

- If PME_Status is cleared, the Function becomes PME Capable again. Next state is the Communicating state.

- If the PME_Status bit is not Clear by the time the PME service timeout expires, a PM_PME Message is re-sent Upstream. Refer to Section 5.3.3.3.1 for an explanation of the timeout mechanism.
- If a PME Message has been issued but the PME_Status has not been cleared by software when the Link is about to be transitioned into a messaging incapable state (a PME_Turn_Off Message is received), the Link transitions into Link Reactivation state after sending a PME_TO_Ack Message. The device also activates the wakeup mechanism.

Link Reactivation State

- Following the restoration of power and clock, and the associated reset, the Link resumes a transaction-capable state. The device clears the wakeup signaling, if necessary, and issues a PM_PME Upstream and transitions into the PME Sent state.

5.4 Native PCI Express Power Management Mechanisms

The following sections define power management features that require new software. While the presence of these features in new PCI Express designs will not break legacy software compatibility, taking the full advantage of them requires new code to manage them.

These features are enumerated and configured using PCI Express native configuration mechanisms as described in Chapter 7 of this specification. Refer to Chapter 7 for specific register locations, bit assignments, and access mechanisms associated with these PCI Express-PM features.

5.4.1 Active State Power Management (ASPM)

All Ports not associated with an Internal Root Complex Link or system Egress Port are required to support the minimum requirements defined herein for Active State Link PM. This feature must be treated as being orthogonal to the PCI-PM software compatible features from a minimum requirements perspective. For example, the Root Complex is exempt from the PCI-PM software compatible features requirements; however, it must implement the minimum requirements of ASPM.

Components in the D0 state (i.e., fully active state) normally keep their Upstream Link in the active L0 state, as defined in Section 5.3.2. ASPM defines a protocol for components in the D0 state to reduce Link power by placing their Links into a low power state and instructing the other end of the Link to do likewise. This capability allows hardware-autonomous, dynamic Link power reduction beyond what is achievable by software-only controlled (i.e., PCI-PM software driven) power management.

Two low power "standby" Link states are defined for ASPM. The L0s low power Link state is optimized for short entry and exit latencies, while providing substantial power savings. If the L0s state is enabled in a device, it is recommended that the device bring its Transmit Link into the L0s state whenever that Link is not in use (refer to Section 5.4.1.1.1 for details relating to the L0s invocation policy). Component support of the L0s Link state from within the D0 device state is optional unless the applicable form factor specification for the Link explicitly requires it.

The L1 Link state is optimized for maximum power savings at a cost of longer entry and exit latencies. L1 reduces Link power beyond the L0s state for cases where very low power is required and longer transition times are acceptable. ASPM support for the L1 Link state is optional unless specifically required by a particular form factor.

Optional L1 PM Substates L1.1 and L1.2 are defined. These substates can further reduce Link power for cases where very low idle power is required, and longer transition times are acceptable.

Each component must report its level of support for ASPM in the ASPM Support field. As applicable, each component shall also report its L0s and L1 exit latency (the time that it requires to transition from the L0s or L1 state to the L0 state). Endpoint Functions must also report the worst-case latency that they can withstand before risking, for example, internal FIFO overruns due to the transition latency from L0s or L1 to the L0 state. Power management software can use the provided information to then enable the appropriate level of ASPM.

The L0s exit latency may differ significantly if the reference clock for opposing sides of a given Link is provided from the same source, or delivered to each component from a different source. PCI Express-PM software informs each device of its clock configuration via the Common Clock Configuration bit in its Capability structure's Link Control register. This bit serves as the determining factor in the L0s exit latency value reported by the device. ASPM may be enabled or disabled by default depending on implementation specific criteria and/or the requirements of the associated form factor specification(s). Software can enable or disable ASPM using a process described in Section 5.4.1.3.1.

Power management software enables or disables ASPM in each Port of a component by programming the ASPM Control field. Note that new BIOS code can effectively enable or disable ASPM functionality when running with a legacy operating system, but a PCI Express-aware operating system might choose to override ASPM settings configured by the BIOS.

IMPLEMENTATION NOTE

Isochronous Traffic and ASPM

Isochronous traffic requires bounded service latency. ASPM may add latency to isochronous transactions beyond expected limits. A possible solution would be to disable ASPM for devices that are configured with an Isochronous Virtual Channel.

For ARI Devices, ASPM Control is determined solely by the setting in Function 0, regardless of Function 0's D-state. The ASPM Control settings in other Functions are ignored by the component.

An Upstream Port of a non-ARI Multi-Function Device may be programmed with different values in their respective ASPM Control fields of each Function. The policy for such a component will be dictated by the most active common denominator among all D0 Functions according to the following rules:

- Functions in a non-D0 state (D1 and deeper) are ignored in determining the ASPM policy
- If any of the Functions in the D0 state has its ASPM disabled (ASPM Control field = 00b) or if at least one of the Functions in the D0 state is enabled for L0s only (ASPM Control field = 01b) and at least one other Function in the D0 state is enabled for L1 only (ASPM Control field = 10b), then ASPM is disabled for the entire component

- Else, if at least one of the Functions in the D0 state is enabled for L0s only (ASPM Control field = 01b), then ASPM is enabled for L0s only
- Else, if at least one of the Functions in the D0 state is enabled for L1 only (ASPM Control field = 10b), then ASPM is enabled for L1 only
- Else, ASPM is enabled for both L0s and L1 states

Note that the components must be capable of changing their behavior during runtime as device Functions enter and exit low power device states. For example, if one Function within a Multi-Function Device is programmed to disable ASPM, then ASPM must be disabled for that device while that Function is in the D0 state. Once the Function transitions to a non-D0 state, ASPM can be enabled if all other Functions are enabled for ASPM.

5.4.1.1 *L0s ASPM State*

Device support of the L0s low power Link state is optional unless the applicable form factor specification for the Link explicitly requires it.

IMPLEMENTATION NOTE

Potential Issues With Legacy Software When L0s is Not Supported

In earlier versions of this specification, device support of L0s was mandatory, and software could legitimately assume that all devices support L0s. Newer hardware components that do not support L0s may encounter issues with such "legacy software". Such software might not even check the ASPM Support field in the Link Capabilities register, might not recognize the subsequently defined values (00b and 10b) for the ASPM Support field, or might not follow the policy of enabling L0s only if components on both sides of the Link each support L0s.

Legacy software (either operating system or firmware) that encounters the previously reserved value 00b (No ASPM Support), will most likely refrain from enabling L1, which is intended behavior. Legacy software will also most likely refrain from enabling L0s for that component's Transmitter (also intended behavior), but it is unclear if such software will also refrain from enabling L0s for the component on the other side of the Link. If software enables L0s on one side when the component on the other side does not indicate that it supports L0s, the result is undefined. Situations where the resulting behavior is unacceptable may need to be handled by updating the legacy software, resorting to "blacklists" or similar mechanisms directing the legacy software not to enable L0s, or simply not supporting the problematic system configurations.

On some platforms, firmware controls ASPM, and the operating system may either preserve or override the ASPM settings established by firmware. This will be influenced by whether the operating system supports controlling ASPM, and in some cases by whether the firmware permits the operating system to take control of ASPM. Also, ASPM control with hot-plug operations may be influenced by whether native PCI Express hot-plug versus ACPI hot-plug is used. Addressing any legacy software issues with L0s may require updating the firmware, the operating system, or both.

When a component does not advertise that it supports L0s, as indicated by its ASPM Support field value being 00b or 10b, it is recommended that the component's L0s Exit Latency field return a value of 111b, indicating the maximum latency range. Advertising this maximum latency range may help discourage legacy software from enabling L0s if it otherwise would do so, and thus may help avoid problems caused by legacy software mistakenly enabling L0s on this component or the component on the other side of the Link.

Transaction Layer and Link Layer timers are not affected by a transition to the L0s state (i.e., they must follow the rules as defined in their respective chapters).

IMPLEMENTATION NOTE

Minimizing L0s Exit Latency

L0s exit latency depends mainly on the ability of the Receiver to quickly acquire bit and Symbol synchronization. Different approaches exist for high-frequency clocking solutions which may differ significantly in their L0s exit latency, and therefore in the efficiency of ASPM. To achieve maximum power savings efficiency with ASPM, L0s exit latency should be kept low by proper selection of the clocking solution.

5.4.1.1.1 Entry into the L0s State

Entry into the L0s state is managed separately for each direction of the Link. It is the responsibility of each device at either end of the Link to initiate an entry into the L0s state on its transmitting Lanes. Software must not enable L0s in either direction on a given Link unless components on both sides of the Link each support L0s; otherwise, the result is undefined.

A Port that is disabled for the L0s state must not transition its transmitting Lanes to the L0s state. However, if the Port advertises that it supports L0s, Port must be able to tolerate having its Receiver Port Lanes enter L0s, (as a result of the device at the other end bringing its transmitting Lanes into L0s state), and then later returning to the L0 state.

L0s Invocation Policy

Ports that are enabled for L0s entry generally should transition their Transmit Lanes to the L0s state if the defined idle conditions (below) are met for a period of time, recommended not to exceed 7 µs. Within this time period, the policy used by the Port to determine when to enter L0s is implementation specific. It is never mandatory for a Transmitter to enter L0s.

Definition of Idle

The definition of an "idle" Upstream Port varies with device Function category. An Upstream Port of a Multi-Function Device is considered idle only when all of its Functions are idle.

A non-Switch Port is determined to be idle if the following conditions are met:

- No TLP is pending to transmit over the Link, or no FC credits are available to transmit any TLPs
- No DLLPs are pending for transmission

A Switch Upstream Port Function is determined to be idle if the following conditions are met:

- None of the Switch's Downstream Port Receive Lanes are in the L0, Recovery, or Configuration state
- No pending TLPs to transmit, or no FC credits are available to transmit anything
- No DLLPs are pending for transmission

A Switch's Downstream Port is determined to be idle if the following conditions are met:

- The Switch's Upstream Port's Receive Lanes are not in the L0, Recovery, or Configuration state

- No pending TLPs to transmit on this Link, or no FC credits are available
- No DLLPs are pending for transmission

Refer to Section 4.2 for details on L0s entry by the Physical Layer.

5.4.1.1.2 Exit from the L0s State

A component with its Transmitter in L0s must initiate L0s exit when it has a TLP or DLLP to transmit across the Link. Note that a transition from the L0s Link state does not depend on the status (or availability) of FC credits. The Link must be able to reach the L0 state, and to exchange FC credits across the Link. For example, if all credits of some type were consumed when the Link entered L0s, then any component on either side of the Link must still be able to transition the Link to the L0 state when new credits need to be sent across the Link. Note that it may be appropriate for a component to anticipate the end of the idle condition and initiate L0s transmit exit; for example, when a NP request is received.

Downstream Initiated Exit

The Upstream Port of a component is permitted to initiate an exit from the L0s low-power state on its Transmit Link, (Upstream Port Transmit Lanes in the case of a Downstream Switch), if it needs to communicate through the Link. The component initiates a transition to the L0 state on Lanes in the Upstream direction as described in Section 4.2.

If the Upstream component is a Switch (i.e., it is not the Root Complex), then it must initiate a transition on its Upstream Port Transmit Lanes (if the Upstream Port's Transmit Lanes are in a low-power state) as soon as it detects an exit from L0s on any of its Downstream Ports.

Upstream Initiated Exit

A Downstream Port is permitted to initiate an exit from L0s low power state on any of its Transmit Links if it needs to communicate through the Link. The component initiates a transition to the L0 state on Lanes in the Downstream direction as described in Chapter 4.

If the Downstream component contains a Switch, it must initiate a transition on all of its Downstream Port Transmit Lanes that are in L0s at that time as soon as it detects an exit from L0s on its Upstream Port. Links that are already in the L0 state are not affected by this transition. Links whose Downstream component is in a low-power state (i.e., D1-$D3_{hot}$ states) are also not affected by the exit transitions.

For example, consider a Switch with an Upstream Port in L0s and a Downstream device in a D1 state. A configuration request packet travels Downstream to the Switch, intending ultimately to reprogram the Downstream device from D1 to D0. The Switch's Upstream Port Link must transition to the L0 state to allow the packet to reach the Switch. The Downstream Link connecting to the device in D1 state will not transition to the L0 state yet; it will remain in the L1 state. The captured packet is checked and routed to the Downstream Port that shares a Link with the Downstream device that is in D1. As described in Section 4.2, the Switch now transitions the Downstream Link to the L0 state. Note that the transition to the L0 state was triggered by the packet being routed to that particular Downstream L1 Link, and not by the transition of the Upstream Port's Link to the L0 state. If the packet's destination was targeting a different Downstream Link, then that particular Downstream Link would have remained in the L1 state.

5.4.1.2 *L1 ASPM State*

A component may optionally support the ASPM L1 state; a state that provides greater power savings at the expense of longer exit latency. L1 exit latency is visible to software, and reported via the L1 Exit Latency field.

IMPLEMENTATION NOTE

Potential Issues With Legacy Software When Only L1 is Supported

In earlier versions of this specification, device support of L0s was mandatory, and there was no architected ASPM Support field value to indicate L1 support without L0s support. Newer hardware components that support only L1 may encounter issues with "legacy software", i.e., software that does not recognize the subsequently defined value for the ASPM Support field.

Legacy software that encounters the previously reserved value 10b (L1 Support), may refrain from enabling both L0s and L1, which unfortunately avoids using L1 with new components that support only L1. While this may result in additional power being consumed, it should not cause any functional misbehavior. However, the same issues with respect to legacy software enabling L0s exist for this 10b case as are described in the Implementation Note "Potential Issues With Legacy Software When L0s is Not Supported" in Section 5.4.1.1.

When supported, L1 entry is disabled by default in the ASPM Control field. Software must enable ASPM L1 on the Downstream component only if it is supported by both components on a Link. Software must sequence the enabling and disabling of ASPM L1 such that the Upstream component is enabled before the Downstream component and disabled after the Downstream component.

5.4.1.2.1 Entry into the L1 State

An Upstream Port on a component enabled for L1 ASPM entry may initiate entry into the L1 Link state.

See Section 5.5.1 for details on transitions into either the L1.1 or L1.2 substates.

IMPLEMENTATION NOTE

Initiating L1

This specification does not dictate when a component with an Upstream Port must initiate a transition to the L1 state. The interoperable mechanisms for transitioning into and out of L1 are defined within this specification; however, the specific ASPM policy governing when to transition into L1 is left to the implementer.

One possible approach would be for the Downstream device to initiate a transition to the L1 state once the device has both its Receiver and Transmitter in the L0s state (RxL0s and TxL0s) for a set amount of time. Another approach would be for the Downstream device to initiate a transition to the L1 state once the Link has been idle in L0 for a set amount of time. This is particularly useful if L0s entry is not enabled. Still another approach would be for the Downstream device to initiate a transition to the L1 state if it has completed its assigned tasks. Note that a component's L1 invocation policy is in no way limited by these few examples.

Three power management Messages provide support for the ASPM L1 state:

- PM_Active_State_Request_L1 (DLLP)
- PM_Request_Ack (DLLP)
- PM_Active_State_Nak (TLP)

Downstream components enabled for ASPM L1 entry negotiate for L1 entry with the Upstream component on the Link.

A Downstream Port must accept a request to enter L1 if all of the following conditions are true:

- The Port supports ASPM L1 entry, and ASPM L1 entry is enabled.[73]
- No TLP is scheduled for transmission
- No Ack or Nak DLLP is scheduled for transmission

A Switch Upstream Port may request L1 entry on its Link provided all of the following conditions are true:

- The Upstream Port supports ASPM L1 entry and it is enabled
- All of the Switch's Downstream Port Links are in the L1 state (or deeper)
- No pending TLPs to transmit

[73] Software must enable ASPM L1 for the Downstream component only if it is also enabled for the Upstream component.

- No pending DLLPs to transmit
- The Upstream Port's Receiver is idle for an implementation specific set amount of time

Note that it is legitimate for a Switch to be enabled for the ASPM L1 Link state on any of its Downstream Ports and to be disabled or not even supportive of ASPM L1 on its Upstream Port. In that case, Downstream Ports may enter the L1 Link state, but the Switch will never initiate an ASPM L1 entry transition on its Upstream Port.

ASPM L1 Negotiation Rules (see Figure 5-6 and Figure 5-7)

- The Downstream component must not initiate ASPM L1 entry until it accumulates at least the minimum number of credits required to send the largest possible packet for any FC type for all enabled VCs.
- Upon deciding to enter a low-power Link state, the Downstream component must block movement of all TLPs from the Transaction Layer to the Data Link Layer for transmission (including completion packets).

If any TLPs become available from the Transaction Layer for transmission during the L1 negotiation process, the transition to L1 must first be completed and then the Downstream component must initiate a return to L0. Refer to Section 5.2 if the negotiation to L1 is interrupted.

- The Downstream component must wait until it receives a Link Layer acknowledgement for the last TLP it had previously sent (i.e., the retry buffer is empty). The component must retransmit a TLP out of its Data Link Layer Retry buffer if required by the Data Link Layer rules.
- The Downstream component then initiates ASPM negotiation by sending a PM_Active_State_Request_L1 DLLP onto its Transmit Lanes. The Downstream component sends this DLLP repeatedly with no more than eight (when using 8b/10b encoding) or 32 (when using 128b/130b encoding) Symbol times of idle between subsequent transmissions of the PM_Active_State_Request_L1 DLLP. The transmission of other DLLPs and SKP Ordered Sets must occur as required at any time between PM_Active_State_Request_L1 transmissions, and do not contribute to this idle time limit. Transmission of SKP Ordered Sets during L1 entry follows the clock tolerance compensation rules in Section 4.2.7.

- The Downstream component continues to transmit the PM_Active_State_Request_L1 DLLP as described above until it receives a response from the Upstream device (see below). The Downstream component remains in this loop waiting for a response from the Upstream component.

During this waiting period, the Downstream component must not initiate any Transaction Layer transfers. It must still accept TLPs and DLLPs from the Upstream component, storing for later transmission any TLP responses required. It continues to respond with DLLPs, including FC update DLLPs, as needed by the Link Layer protocol.

If the Downstream component for any reason needs to transmit a TLP on the Link, it must first complete the transition to the low-power Link state. Once in a lower power Link state, the Downstream component must then initiate exit of the low-power Link state to handle the transfer. Refer to Section 5.2 if the negotiation to L1 is interrupted.

- The Upstream component must immediately (while obeying all other rules in this specification) respond to the request with either an acceptance or a rejection of the request.

If the Upstream component is not able to accept the request, it must immediately (while obeying all other rules in this specification) reject the request.

- Refer to Section 5.2 if the negotiation to L1 is interrupted.

Rules in case of rejection:

- In the case of a rejection, the Upstream component must schedule, as soon as possible, a rejection by sending the PM_Active_State_Nak Message to the Downstream component. Once the PM_Active_State_Nak Message is sent, the Upstream component is permitted to initiate any TLP or DLLP transfers.
- If the request was rejected, it is generally recommended that the Downstream component immediately transition its Transmit Lanes into the L0s state, provided L0s is enabled and that conditions for L0s entry are met.
- Prior to transmitting a PM_Active_State_Request_L1 DLLP associated with a subsequent ASPM L1 negotiation sequence, the Downstream component must either enter and exit L0s on its Transmitter, or it must wait at least 10 µs from the last transmission of the PM_Active_State_Request_L1 DLLP associated with the preceding ASPM L1 negotiation. This 10 µs timer must count only time spent in the LTSSM L0 and L0s states. The timer must hold in the LTSSM Recovery state. If the Link goes down and comes back up, the timer is ignored and the component is permitted to issue new ASPM L1 request after the Link has come back up.

IMPLEMENTATION NOTE

ASPM L1 Accept/Reject Considerations for the Upstream Component

When the Upstream component has responded to the Downstream component's ASPM L1 request with a PM_Request_Ack DLLP to accept the L1 entry request, the ASPM L1 negotiation protocol clearly and unambiguously ends with the Link entering L1. However, if the Upstream component responds with a PM_Active_State_Nak Message to reject the L1 entry request, the termination of the ASPM L1 negotiation protocol is less clear. Therefore, both components need to be designed to unambiguously terminate the protocol exchange. If this is not done, there is the risk that the two components will get out of sync with each other, and the results may be undefined. For example, consider the following case:

- The Downstream component requests ASPM L1 entry by transmitting a sequence of PM_Active_State_Request_L1 DLLPs.
- Due to a temporary condition, the Upstream component responds with a PM_Active_State_Nak Message to reject the L1 request.
- The Downstream component continues to transmit the PM_Active_State_Request_L1 DLLPs for some time before it is able to respond to the PM_Active_State_Nak Message.
- Meanwhile, the temporary condition that previously caused the Upstream component to reject the L1 request is resolved, and the Upstream component erroneously sees the continuing PM_Active_State_Request_L1 DLLPs as a new request to enter L1, and responds by transmitting PM_Request_Ack DLLPs Downstream.

At this point, the result is undefined, because the Downstream component views the L1 request as rejected and finishing, but the Upstream component views the situation as a second L1 request being accepted.

To avoid this situation, the Downstream component needs to provide a mechanism to distinguish between one ASPM L1 request and another. The Downstream component does this by entering L0s or by waiting a minimum of 10 µs from the transmission of the last PM_Active_State_Request_L1 DLLP associated with the first ASPM L1 request before starting transmission of the PM_Active_State_Request_L1 DLLPs associated with the second request (as described above).

If the Upstream component is capable of exhibiting the behavior described above, then it is necessary for the Upstream component to recognize the end of an L1 request sequence by detecting a transition to L0s on its Receiver or a break in the reception of PM_Active_State_Request_L1 DLLPs of 9.5 µs measured while in L0/L0s or more as a separation between ASPM L1 requests by the Downstream component.

If there is a possibility of ambiguity, the Upstream component should reject the L1 request to avoid potentially creating the ambiguous situation outlined above.

Rules in case of acceptance:

- If the Upstream component is ready to accept the request, it must block scheduling of any TLPs from the Transaction Layer.
- The Upstream component then must wait until it receives a Data Link Layer acknowledgement for the last TLP it had previously sent. The Upstream component must retransmit a TLP if required by the Data Link Layer rules.
- Once all TLPs have been acknowledged, the Upstream component sends a PM_Request_Ack DLLP Downstream. The Upstream component sends this DLLP repeatedly with no more than eight (when using 8b/10b encoding) or 32 (when using 128b/130b encoding) Symbol times of idle between subsequent transmissions of the PM_Request_Ack DLLP. The transmission of SKP Ordered Sets must occur as required at any time between PM_Request_Ack transmissions, and do not contribute to this idle time limit. Transmission of SKP Ordered Sets during L1 entry follows the clock tolerance compensation rules in Section 4.2.7.
- The Upstream component continues to transmit the PM_Request_Ack DLLP as described above until it observes its Receive Lanes enter into the Electrical Idle state. Refer to Chapter 4 for more details on the Physical Layer behavior.
- If the Upstream component needs, for any reason, to transmit a TLP on the Link after it sends a PM_Request_Ack DLLP, it must first complete the transition to the low-power state, and then initiate an exit from the low-power state to handle the transfer once the Link is back to L0. Refer to Section 5.2 if the negotiation to L1 is interrupted.
    - The Upstream component must initiate an exit from L1 in this case even if it does not have the required flow control credit to transmit the TLP(s).
- When the Downstream component detects a PM_Request_Ack DLLP on its Receive Lanes (signaling that the Upstream device acknowledged the transition to L1 request), the Downstream component then ceases sending the PM_Active_State_Request_L1 DLLP, disables DLLP, TLP transmission and brings its Transmit Lanes into the Electrical Idle state.
- When the Upstream component detects an Electrical Idle on its Receive Lanes (signaling that the Downstream component has entered the L1 state), it then ceases to send the PM_Request_Ack DLLP, disables DLLP, TLP transmission and brings the Downstream direction of the Link into the Electrical Idle state.

Notes:

1. The transaction Layer Completion Timeout mechanism is not affected by transition to the L1 state (i.e., it must keep counting).
2. Flow Control Update timers are frozen while the Link is in L1 state to prevent a timer expiration that will unnecessarily transition the Link back to the L0 state.

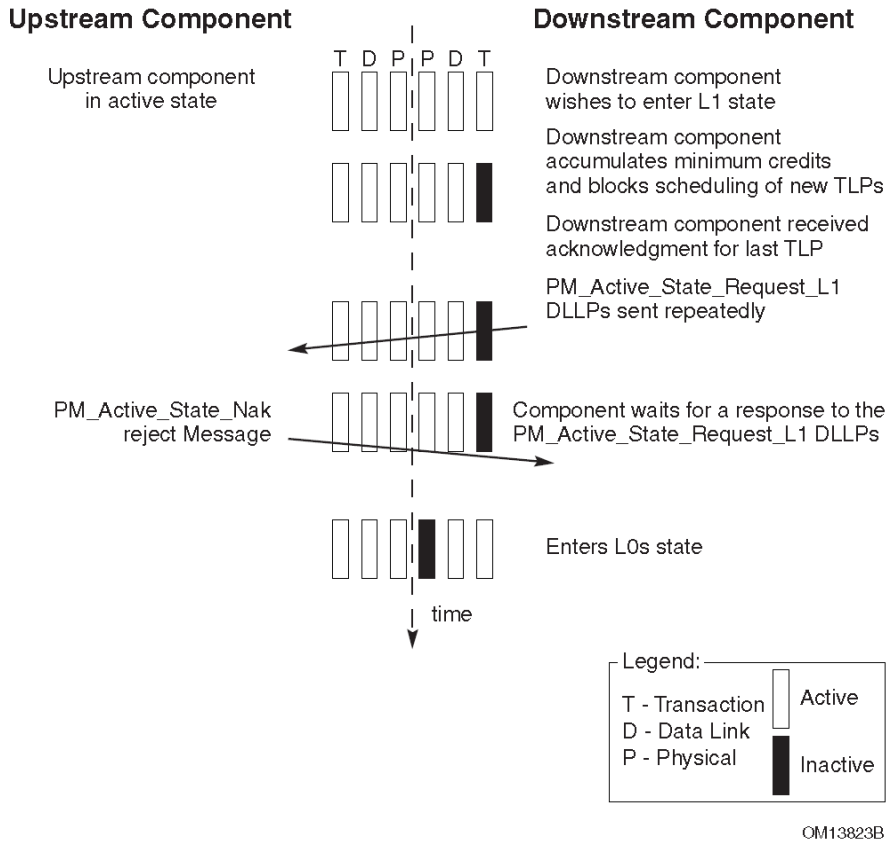


Figure 5-6: L1 Transition Sequence Ending with a Rejection (L0s Enabled)

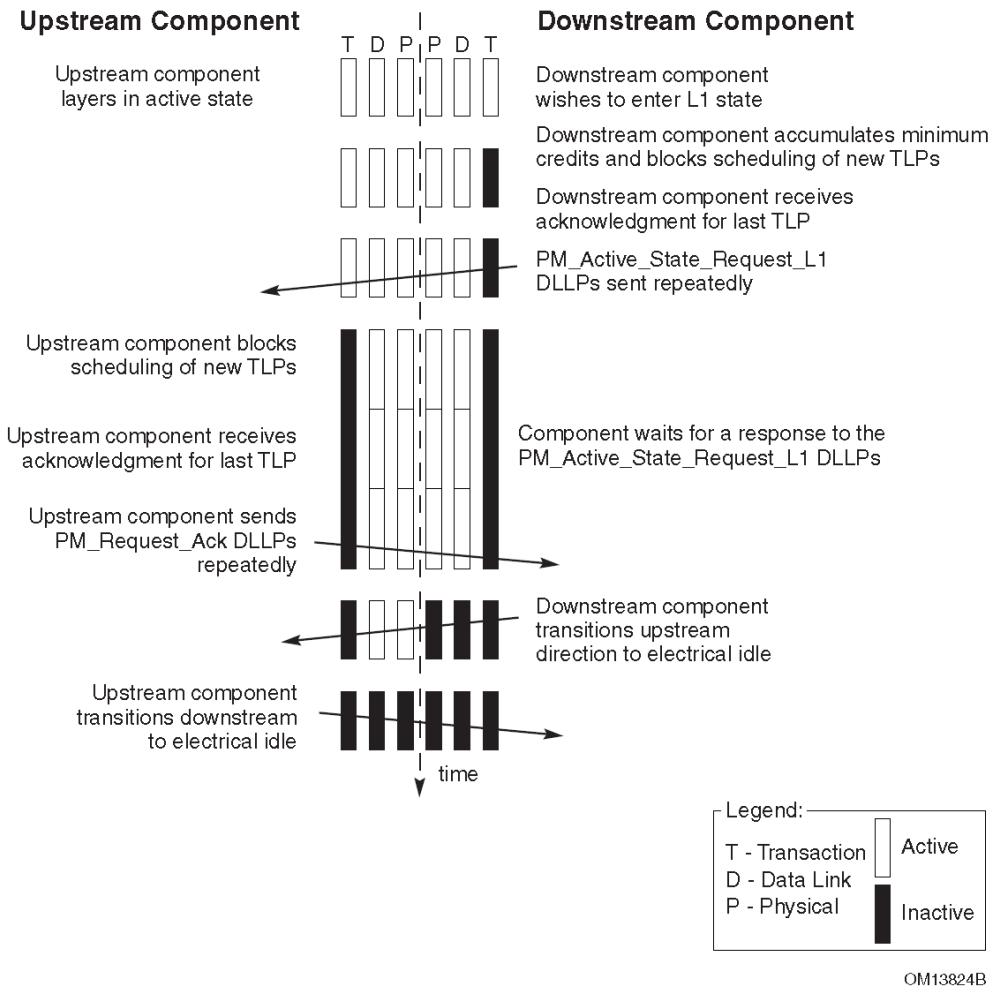


Figure 5-7: L1 Successful Transition Sequence

5.4.1.2.2 Exit from the L1 State

Components on either end of a Link may initiate an exit from the L1 Link state.

See Section 5.5.1 for details on transitions into either the L1.1 or L1.2 substates.

Upon exit from L1, it is recommended that the Downstream component send flow control update DLLPs for all enabled VCs and FC types starting within 1 µs of L1 exit.

Downstream Component Initiated Exit

An Upstream Port must initiate an exit from L1 on its Transmit Lanes if it needs to communicate through the Link. The component initiates a transition to the L0 state as described in Chapter 4. The Upstream component must respond by initiating a similar transition of its Transmit Lanes.

If the Upstream component is a Switch Downstream Port, (i.e., it is not a Root Complex Root Port), the Switch must initiate an L1 exit transition on its Upstream Port's Transmit Lanes, (if the Upstream Port's Link is in the L1 state), as soon as it detects the L1 exit activity on any of its

Downstream Port Links. Since L1 exit latencies are relatively long, a Switch must not wait until its Downstream Port Link has fully exited to L0 before initiating an L1 exit transition on its Upstream Port Link. Waiting until the Downstream Link has completed the L0 transition will cause a Message traveling through several Switches to experience accumulating latency as it traverses each Switch.

A Switch is required to initiate an L1 exit transition on its Upstream Port Link after no more than 1 µs from the beginning of an L1 exit transition on any of its Downstream Port Links. Refer to Section 4.2 for details of the Physical Layer signaling during L1 exit.

Consider the example in Figure 5-8. The numbers attached to each Port represent the corresponding Port's reported Transmit Lanes L1 exit latency in units of microseconds.

Links 1, 2, and 3 are all in the L1 state, and Endpoint C initiates a transition to the L0 state at time T. Since Switch B takes 32 µs to exit L1 on its Ports, Link 3 will transition to the L0 state at T+32 (longest time considering T+8 for the Endpoint C, and T+32 for Switch B).

Switch B is required to initiate a transition from the L1 state on its Upstream Port Link (Link 2) after no more than 1 µs from the beginning of the transition from the L1 state on Link 3. Therefore, transition to the L0 state will begin on Link 2 at T+1. Similarly, Link 1 will start its transition to the L0 state at time T+2.

Following along as above, Link 2 will complete its transition to the L0 state at time T+33 (since Switch B takes longer to transition and it started at time T+1). Link 1 will complete its transition to the L0 state at time T+34 (since the Root Complex takes 32 µs to transition and it started at time T+2).

Therefore, among Links 1, 2, and 3, the Link to complete the transition to the L0 state last is Link 1 with a 34 µs delay. This is the delay experienced by the packet that initiated the transition in Endpoint C.

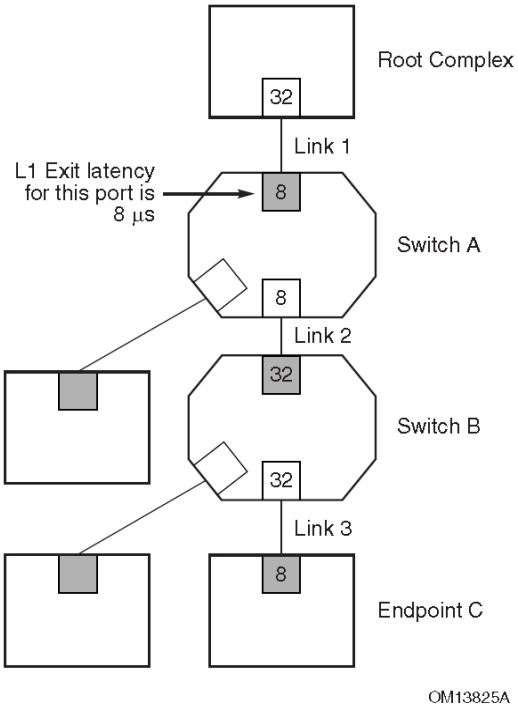


Figure 5-8: Example of L1 Exit Latency Computation

Switches are not required to initiate an L1 exit transition on any other of their Downstream Port Links.

Upstream Component Initiated Exit

A Root Complex, or a Switch must initiate an exit from L1 on any of its Root Ports, or Downstream Port Links if it needs to communicate through that Link. The Switch or Root Complex must be capable of initiating L1 exit even if it does not have the flow control credits needed to transmit a given TLP. The component initiates a transition to the L0 state as described in Chapter 4. The Downstream component must respond by initiating a similar transition on its Transmit Lanes.

If the Downstream component contains a Switch, it must initiate a transition on all of its Downstream Links (assuming the Downstream Link is in an ASPM L1 state) as soon as it detects an exit from L1 state on its Upstream Port Link. Since L1 exit latencies are relatively long, a Switch must not wait until its Upstream Port Link has fully exited to L0 before initiating an L1 exit transition on its Downstream Port Links. If that were the case, a Message traveling through multiple Switches would experience accumulating latency as it traverses each Switch.

A Switch is required to initiate a transition from L1 state on all of its Downstream Port Links that are currently in L1 after no more than 1 µs from the beginning of a transition from L1 state on its Upstream Port. Refer to Section 4.2 for details of the Physical Layer signaling during L1 exit. Downstream Port Links that are already in the L0 state do not participate in the exit transition. Downstream Port Links whose Downstream component is in a low power D-state (D1-$D3_{hot}$) are also not affected by the L1 exit transitions (i.e., such Links must not be transitioned to the L0 state).

5.4.1.3 *ASPM Configuration*

All Functions must implement the following configuration bits in support of ASPM. Refer to Chapter 7 for configuration register assignment and access mechanisms.

Each component reports its level of support for ASPM in the ASPM Support field below.

Table 5-3: Encoding of the ASPM Support Field

Field	Description
ASPM Support	00b – No ASPM support 01b – L0s supported 10b – L1 supported 11b – L0s and L1 supported

Software must not enable L0s in either direction on a given Link unless components on both sides of the Link each support L0s; otherwise, the result is undefined.

Each component reports the source of its reference clock in its Slot Clock Configuration bit located in its Capability structure's Link Status register.

Table 5-4: Description of the Slot Clock Configuration Bit

Bit	Description
Slot Clock Configuration	This bit, when set, indicates that the component uses the same physical reference clock that the platform provides on the connector. If the component uses an independent clock irrespective of the presence of a reference on the connector, this bit must be clear. For Root and Switch Downstream Ports, this bit, when set, indicates that the Downstream Port is using the same reference clock as the Downstream component or the slot. For Switch and Bridge Upstream Ports, this bit when set, indicates that the Upstream Port is using the same reference clock that the platform provides. Otherwise it is clear.

Each component must support the Common Clock Configuration bit in their Capability structure's Link Control register. Software writes to this register bit to indicate to the device whether it is sharing the same clock source as the device on the other end of the Link.

Table 5-5: Description of the Common Clock Configuration Bit

Bit	Description
Common Clock Configuration	This bit when set indicates that this component and the component at the opposite end of the Link are operating with a common clock source. A value of 0b indicates that this component and the component at the opposite end of the Link are operating with separate reference clock sources. Default value of this bit is 0b. Components utilize this common clock configuration information to report the correct L0s and L1 Exit Latencies.

Each Port reports the L0s and L1 exit latency (the time that they require to transition their Receive Lanes from the L0s or L1 state to the L0 state) in the L0s Exit Latency and the L1 Exit Latency

configuration fields, respectively. If a Port does not support L0s or ASPM L1, the value of the respective exit latency field is undefined.

Table 5-6: Encoding of the L0s Exit Latency Field

Field	Description
L0s Exit Latency	000b – Less than 64 ns 001b – 64 ns to less than 128 ns 010b – 128 ns to less than 256 ns 011b – 256 ns to less than 512 ns 100b – 512 ns to less than 1 μs 101b – 1 μs to less than 2 μs 110b – 2 μs to 4 μs 111b – More than 4 μs

Table 5-7: Encoding of the L1 Exit Latency Field

Field	Description
L1 Exit Latency	000b – Less than 1 μs 001b – 1 μs to less than 2 μs 010b – 2 μs to less than 4 μs 011b – 4 μs to less than 8 μs 100b – 8 μs to less than 16 μs 101b – 16 μs to less than 32 μs 110b – 32 μs to 64 μs 111b – More than 64 μs

Endpoints also report the additional latency that they can absorb due to the transition from L0s state or L1 state to the L0 state. This is reported in the Endpoint L0s Acceptable Latency and Endpoint L1 Acceptable Latency fields, respectively.

Power management software, using the latency information reported by all components in the Hierarchy, can enable the appropriate level of ASPM by comparing exit latency for each given path from Root to Endpoint against the acceptable latency that each corresponding Endpoint can withstand.

Table 5-8: Encoding of the Endpoint L0s Acceptable Latency Field

Field	Description
Endpoint L0s Acceptable Latency	000b – Maximum of 64 ns 001b – Maximum of 128 ns 010b – Maximum of 256 ns 011b – Maximum of 512 ns 100b – Maximum of 1 μs 101b – Maximum of 2 μs 110b – Maximum of 4 μs 111b – No limit

Table 5-9: Encoding of the Endpoint L1 Acceptable Latency Field

Field	Description
Endpoint L1 Acceptable Latency	000b – Maximum of 1 μs 001b – Maximum of 2 μs 010b – Maximum of 4 μs 011b – Maximum of 8 μs 100b – Maximum of 16 μs 101b – Maximum of 32 μs 110b – Maximum of 64 μs 111b – No limit

Power management software enables or disables ASPM in each component by programming the ASPM Control field.

Table 5-10: Encoding of the ASPM Control Field

Field	Description
ASPM Control	00b – Disabled 01b – L0s Entry Enabled 10b – L1 Entry Enabled 11b – L0s and L1 Entry enabled

ASPM Control = 00b

Port's Transmitter must not enter L0s.

Ports connected to the Downstream end of the Link must not issue a PM_Active_State_Request_L1 DLLP on its Upstream Link.

Ports connected to the Upstream end of the Link receiving a L1 request must respond with negative acknowledgement.

ASPM Control = 01b

Port must bring a Link into L0s state if all conditions are met.

Ports connected to the Downstream end of the Link must not issue a PM_Active_State_Request_L1 DLLP on its Upstream Link.

Ports connected to the Upstream end of the Link receiving a L1 request must respond with negative acknowledgement.

ASPM Control = 10b

Port's Transmitter must not enter L0s.

Ports connected to the Downstream end of the Link may issue PM_Active_State_Request_L1 DLLPs.

Ports connected to the Upstream end of the Link must respond with positive acknowledgement to a L1 request and transition into L1 if the conditions for the Root Complex Root Port or Switch Downstream Port in Section 5.4.1.2.1 are met.

ASPM Control = 11b

Port must bring a Link into the L0s state if all conditions are met.

Ports connected to the Downstream end of the Link may issue PM_Active_State_Request_L1 DLLPs.

Ports connected to the Upstream end of the Link must respond with positive acknowledgement to a L1 request and transition into L1 if the conditions for the Root Complex Root Port or Switch Downstream Port in Section 5.4.1.2.1 are met.

5.4.1.3.1 Software Flow for Enabling or Disabling ASPM

Following is an example software algorithm that highlights how to enable or disable ASPM in a component.

- PCI Express components power up with an appropriate value in their Slot Clock Configuration bit. The method by which they initialize this bit is device-specific.
- PCI Express system software scans the Slot Clock Configuration bit in the components on both ends of each Link to determine if both are using the same reference clock source or reference clocks from separate sources. If the Slot Clock Configuration bits in both devices are Set, they are both using the same reference clock source, otherwise they're not.
- PCI Express software updates the Common Clock Configuration bits in the components on both ends of each Link to indicate if those devices share the same reference clock and triggers Link retraining by writing 1b to the Retrain Link bit in the Link Control register of the Upstream component.
- Devices must reflect the appropriate L0s/L1 exit latency in their L0s/L1 Exit Latency fields, per the setting of the Common Clock Configuration bit.
- PCI Express system software then reads and calculates the L0s/L1 exit latency for each Endpoint based on the latencies reported by each Port. Refer to Section 5.4.1.2.2 for an example.
- For each component with one or more Endpoint Functions, PCI Express system software examines the Endpoint L0s/L1 Acceptable Latency, as reported by each Endpoint Function in its Link Capabilities register, and enables or disables L0s/L1 entry (via the ASPM Control field in the Link Control register) accordingly in some or all of the intervening device Ports on that hierarchy.

5.5 L1 PM Substates

L1 PM Substates establish a Link power management regime that creates lower power substates of the L1 Link state (see Figure 5-9), and associated mechanisms for using those substates. The L1 PM Substates are:

- L1.0 substate
  - The L1.0 substate corresponds to the conventional L1 Link state. This substate is entered whenever the Link enters L1. The L1 PM Substate mechanism defines transitions from this substate to and from the L1.1 and L1.2 substates.
  - The Upstream and Downstream Ports must be enabled to detect Electrical Idle exit as required in Section 4.2.6.7.2.

- L1.1 substate
  - Link common mode voltages are maintained.
  - Uses a bidirectional open-drain clock request (CLKREQ#) signal for entry to and exit from this state.
  - The Upstream and Downstream Ports are not required to be enabled to detect Electrical Idle exit.
- L1.2 substate
  - Link common mode voltages are not required to be maintained.
  - Uses a bidirectional open-drain clock request (CLKREQ#) signal for entry to and exit from this state.
  - The Upstream and Downstream Ports are not required to be enabled to detect Electrical Idle exit.

Ports that support L1 PM Substates must not require a reference clock while in L1 PM Substates other than L1.0.

Ports that support the L1.2 substate for ASPM L1 must support Latency Tolerance Reporting (LTR).

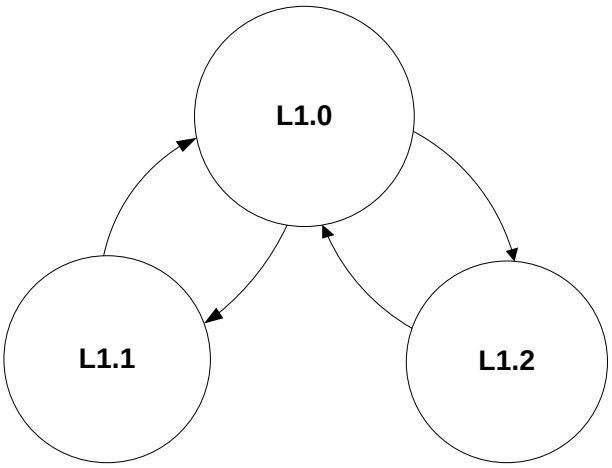


Figure 5-9: State Diagram for L1 PM Substates

- When enabled, the L1 PM Substates mechanism applies the following additional requirements to the CLKREQ# signal:The CLKREQ# signal must be supported as a bi-directional open drain signal by both the Upstream and Downstream Ports of the Link. Each Port must have a unique instance of the signal, and the Upstream and Downstream Port CLKREQ# signals must be connected.
- It is permitted for the Upstream Port to deassert CLKREQ# when the Link is in the PCI-PM L1 or ASPM L1 states, or when the Link is in the L2/L3 Ready pseudo-state; CLKREQ# must be asserted by the Upstream Port when the Link is in any other state.

- All other specifications related to the CLKREQ# signal that are not specifically defined or modified by L1 PM Substates continue to apply.

If these requirements cannot be satisfied in a particular system, then L1 PM Substates must not be enabled.

IMPLEMENTATION NOTE

CLKREQ# Connection Topologies

For an Upstream component the connection topologies for the CLKREQ# signal can vary. A few examples of CLKREQ# connection topologies are described below. For the Downstream component these cases are essentially the same, however from the Upstream component's perspective, there are some key differences that are described below.

Example 1: Single Downstream Port with a single PLL connected to a single Upstream Port (see Figure 5-10).

In this platform configuration the Upstream component (A) has only a single CLKREQ# signal. The Upstream and Downstream Ports' CLKREQ# (A and B) signals are connected to each other. In this case, Upstream component (A), must assert CLKREQ# signal whenever it requires a reference clock.

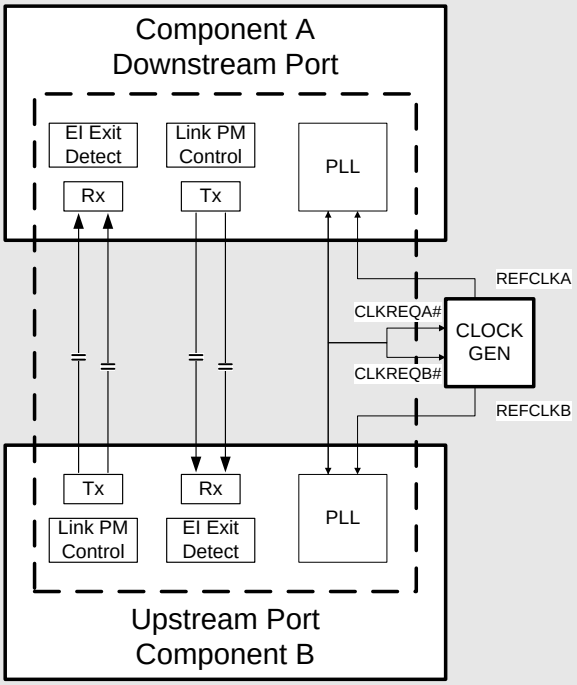


Figure 5-10: Downstream Port with a Single PLL

Example 2: Upstream component with multiple Downstream Ports, with a common shared PLL, connected to separate Downstream components (see Figure 5-11).

In this example configuration, there are three instances of CLKREQ# signal for the Upstream component (A), one per Downstream Port and a common shared CLKREQ# signal for the Upstream component (A). In this topology the Downstream Port CLKREQ# (CLKREQB#,

CLKREQC#) signals are used to connect to the CLKREQ# signal of the Upstream Port of the Downstream components (B and C). The common shared CLKREQ# (CLKREQA#) signal for the Upstream component is used to request the reference clock for the shared PLL. The PLL control logic in Upstream component (A) can only be turned off and CLKREQA# be deasserted when both the Downstream Ports are in L1.1 or L1.2 Substates, and all internal (A) consumers of the PLL don't require a clock.

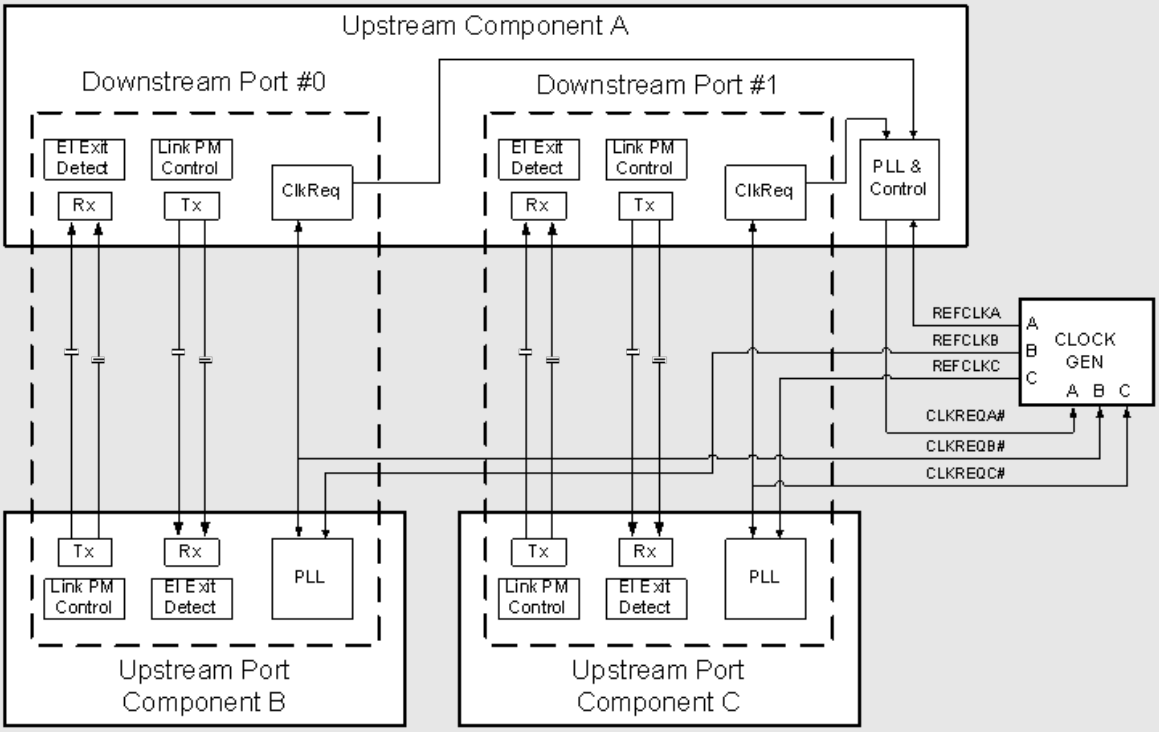


Figure 5-11: Multiple Downstream Ports with a shared PLL

It is necessary for board implementers to consider what CLKREQ# topologies will be supported by components in order to make appropriate board level connections to support L1 PM Substates and for the reference clock generation.

IMPLEMENTATION NOTE

Avoiding Unintended Interactions Between L1 PM Substates and the LTSSM

It is often the case that implementation techniques which save power will also increase the latency to return to normal operation. When implementing L1 PM Substates, it is important for the implementer to ensure that any added delays will not negatively interact with other elements of the platform. It is particularly important to ensure that LTSSM timeout conditions are not unintentionally triggered. Although typical implementations will not approach the latencies that would cause such interactions, the responsibility lies with the implementer to ensure that correct overall operation is achieved.

5.5.1 Entry conditions for L1 PM Substates and L1.0 Requirements

The Link is considered to be in PCI-PM L1.0 when the L1 PM Substate is L1.0 and the LTSSM entered L1 through PCI-PM compatible power management. The Link is considered to be in ASPM L1.0 when the L1 PM Substate is in L1.0 and LTSSM entered L1 through ASPM.

The following rules define how the L1.1 and L1.2 substates are entered:

- Both the Upstream and Downstream Ports must monitor the logical state of the CLKREQ# signal.
- When in PCI-PM L1.0 and the PCI-PM L1.2 Enable bit is Set, the L1.2 substate must be entered when CLKREQ# is deasserted.
- When in PCI-PM L1.0 and the PCI-PM L1.1 Enable bit is Set, the L1.1 substate must be entered when CLKREQ# is deasserted and the PCI-PM L1.2 Enable bit is Clear.
- When in ASPM L1.0 and the ASPM L1.2 Enable bit is Set, the L1.2 substate must be entered when CLKREQ# is deasserted and all of the following conditions are true:
  - The reported snooped LTR value last sent or received by this Port is greater than or equal to the value set by the LTR_L1.2_THRESHOLD Value and Scale fields, or there is no snoop service latency requirement.
  - The reported non-snooped LTR last sent or received by this Port value is greater than or equal to the value set by the LTR_L1.2_THRESHOLD Value and Scale fields, or there is no non-snoop service latency requirement.
- When in ASPM L1.0 and the ASPM L1.1 Enable bit is Set, the L1.1 substate must be entered when CLKREQ# is deasserted and the conditions for entering the L1.2 substate are not satisfied.

When the entry conditions for L1.2 are satisfied, the following rules apply:

- Both the Upstream and Downstream Ports must monitor the logical state of the CLKREQ# input signal.
- An Upstream Port must not deassert CLKREQ# until the Link has entered L1.0.
- It is permitted for either Port to assert CLKREQ# to prevent the Link from entering L1.2.
- A Downstream Port intending to block entry into L1.2 must assert CLKREQ# before the Link enters L1.
- When CLKREQ# is deasserted the Ports enter the L1.2.Entry substate of L1.2.

If a Downstream Port is in PCI-PM L1.0 and PCI-PM L1.1 Enable and/or PCI-PM L1.2 Enable are Set, or if a Downstream Port is in ASPM L1.0 and ASPM L1.1 Enable and/or ASPM L1.2 Enable are Set, and the Downstream Port initiates an exit to Recovery without having entered L1.1 or L1.2, the Downstream Port must assert CLKREQ# until the Link exits Recovery.

5.5.2 L1.1 Requirements

Both Upstream and Downstream Ports are permitted to deactivate mechanisms for electrical idle (EI) exit detection and Refclk activity detection if implemented, however both ports must maintain common mode.

5.5.2.1 *Exit from L1.1*

If either the Upstream or Downstream Port needs to initiate exit from L1.1, it must assert CLKREQ# until the Link exits Recovery. The Upstream Port must assert CLKREQ# on entry to Recovery, and must continue to assert CLKREQ# until the next entry into L1, or other state allowing CLKREQ# deassertion.

- Next state is L1.0 if CLKREQ# is asserted.
  - The Refclk will eventually be turned on as defined in the PCI Express Mini CEM spec, which may be delayed according to the LTR advertized by the Upstream Port.

Figure 5-12 illustrates entry into L1.1 with exit driven by the Upstream Port.

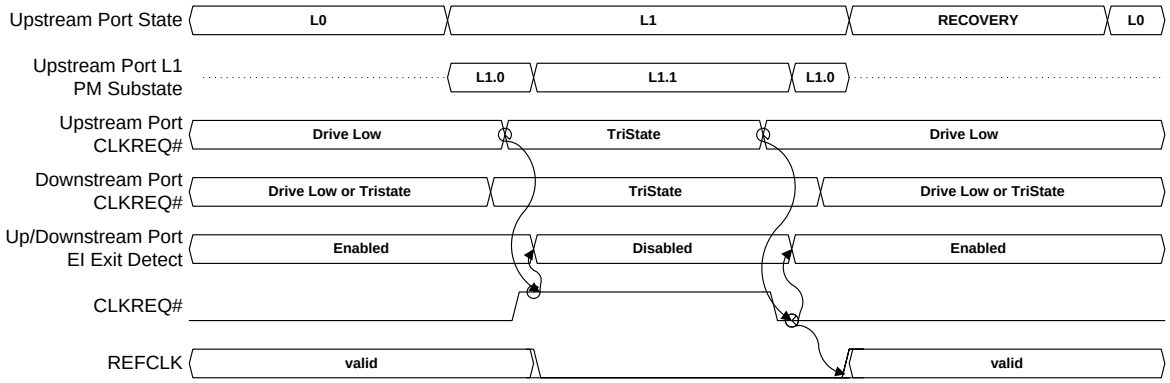


Figure 5-12: Example: L1.1 Waveforms Illustrating Upstream Port Initiated Exit

Figure 5-13 illustrates entry into L1.1 with exit driven by the Downstream Port.

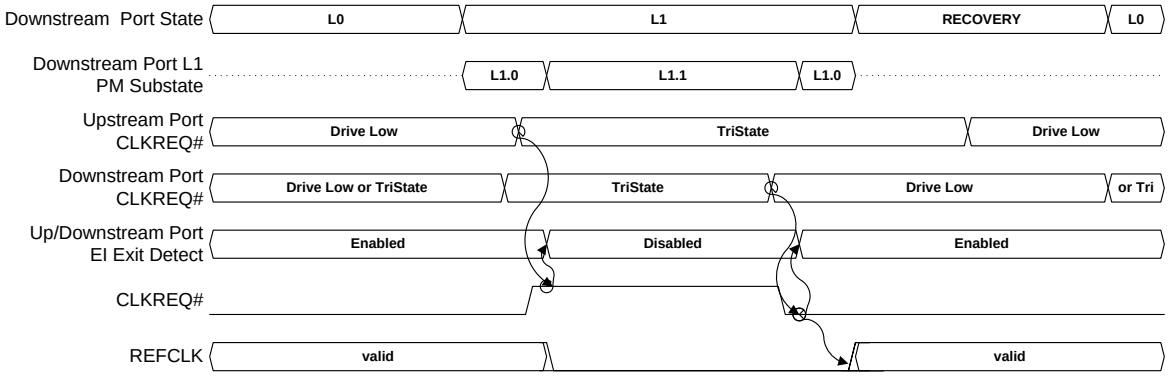


Figure 5-13: Example: L1.1 Waveforms Illustrating Downstream Port Initiated Exit

5.5.3 L1.2 Requirements

All Link and PHY state must be maintained during L1.2, or must be restored upon exit using implementation-specific means, and the LTSSM and corresponding Port state upon exit from L1.2 must be indistinguishable from the L1.0 LTSSM and Port state.

L1.2 has additional requirements that do not apply to L1.1. These requirements are documented in this section.

L1.2 has three substates, which are defined below (see Figure 5-14).

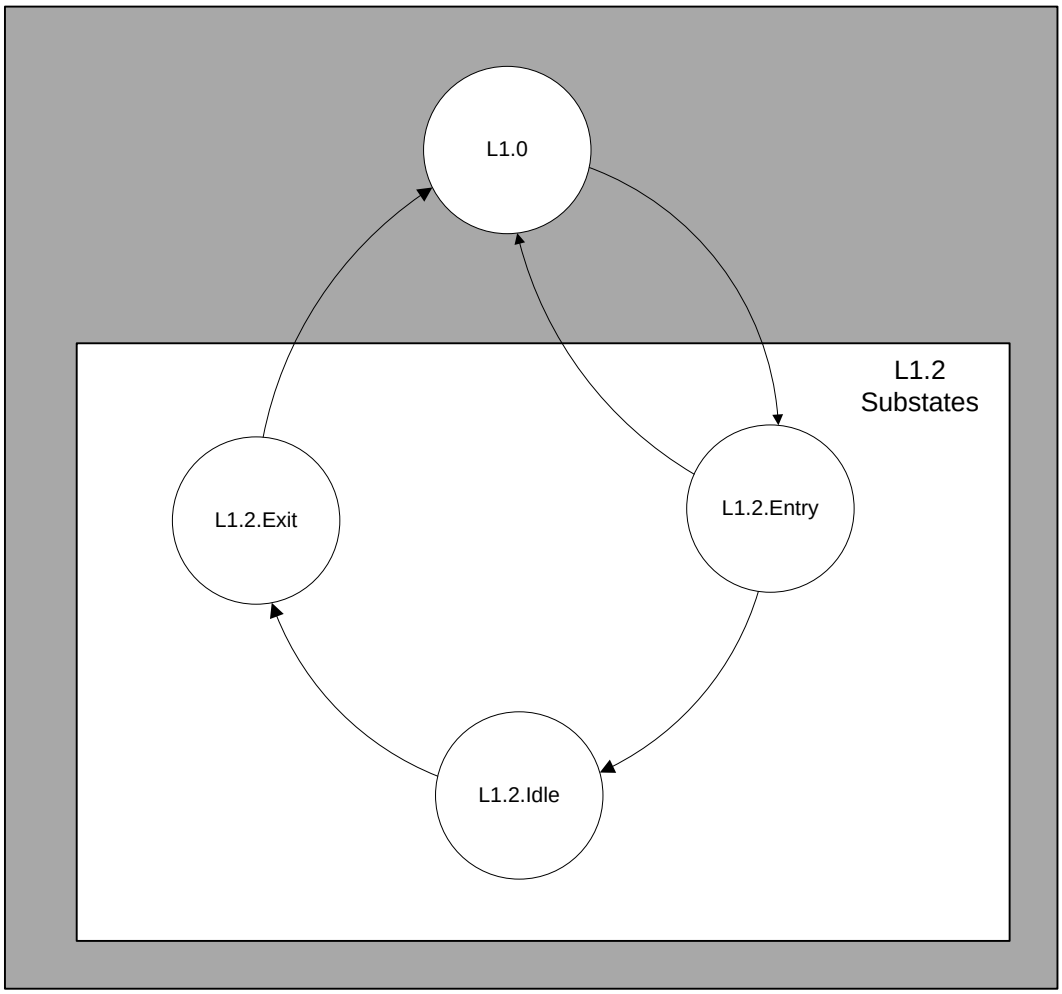


Figure 5-14: L1.2 Substates

5.5.3.1 *L1.2.Entry*

L1.2.Entry is a transitional state on entry into L1.2 to allow time for Refclk to turn off and to ensure both Ports have observed CLKREQ# deasserted. The following rules apply to L1.2.Entry:

- Both Upstream and Downstream Ports continue to maintain common mode.
- Both Upstream and Downstream Ports may turn off their electrical idle (EI) exit detect circuitry.

- The Upstream and Downstream Ports must not assert CLKREQ# in this state.
- Refclk must be turned off within $T_{L1O_REFCLK_OFF}$.
- Next state is L1.0 if CLKREQ# is asserted, else the next state is L1.2.Idle after waiting for T_{POWER_OFF}.

Note that there is a boundary condition which can occur when one Port asserts CLKREQ# shortly after the other Port deasserts CLKREQ#, but before the first Port has observed CLKREQ# deasserted. This is an unavoidable boundary condition that implementations must handle correctly. An example of this condition is illustrated in Figure 5-15.

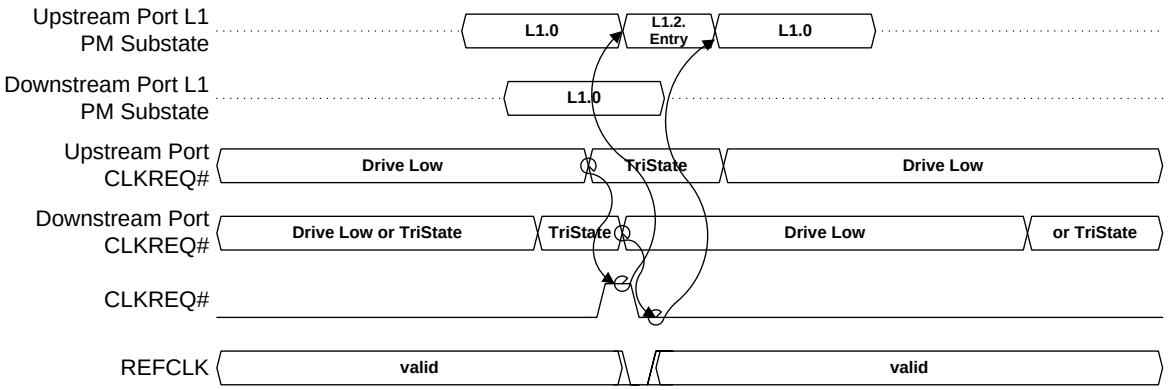


Figure 5-15: Example: Illustration of Boundary Condition due to Different Sampling of CLKREQ#

5.5.3.2 *L1.2.Idle*

When requirements for the entry into L1.2.Idle state (see Section 5.5.1) have been satisfied then the Ports enter the L1.2.Idle substate. The following rules apply in L1.2.Idle:

- Both Upstream and Downstream Ports may power-down any active logic, including circuits required to maintain common mode.
- The PHY of both Upstream and Downstream Ports may have their power removed.

The following rules apply for L1.2.Idle state when using the CLKREQ#-based mechanism:

- If either the Upstream or Downstream Port needs to exit L1.2, it must assert CLKREQ# after ensuring that $T_{L1.2}$ has been met.
- If the Downstream Port is initiating exit from L1, it must assert CLKREQ# until the Link exits Recovery. The Upstream Port must assert CLKREQ# on entry to Recovery, and must continue to assert CLKREQ# until the next entry into L1, or other state allowing CLKREQ# deassertion.
- If the Upstream Port is initiating exit from L1, it must continue to assert CLKREQ# until the next entry into L1, or other state allowing CLKREQ# deassertion.

- Both the Upstream and Downstream Ports must monitor the logical state of the CLKREQ# input signal.
- Next state is L1.2.Exit if CLKREQ# is asserted.

5.5.3.3 *L1.2.Exit*

This is a transitional state on exit from L1.2 to allow time for both devices to power up. In L1.2.Exit, the following rules apply:

- The PHYs of both Upstream and Downstream Ports must be powered.
- It must not be assumed that common mode has been maintained.

5.5.3.3.1 Exit from L1.2

- The following rules apply for L1.2 Exit using the CLKREQ#-based mechanism:
- Both Upstream and Downstream Ports must power up any circuits required for L1.0, including circuits required to maintain common mode.
- The Upstream and Downstream Ports must not change their driving state of CLKREQ# in this state.
- Refclk must be turned on no earlier than $T_{L1O_REFCLK_ON}$ minimum time, and may take up to the amount of time allowed according to the LTR advertized by the Endpoint before becoming valid.
- Next state is L1.0 after waiting for T_{POWER_ON}.
    - Common mode is permitted to be established passively during L1.0, and actively during Recovery. In order to ensure common mode has been established, the Downstream Port must maintain a timer, and the Downstream Port must continue to send TS1 training sequences until a minimum of $T_{COMMONMODE}$ has elapsed since the Downstream Port has started transmitting TS1 training sequences and has detected electrical idle exit on any Lane of the configured Link.

Figure 5-16 illustrates the signal relationships and timing constraints associated with L1.2 entry and Upstream Port initiated exit.

Figure 5-17 illustrates the signal relationships and timing constraints associated with L1.2 entry and Downstream Port initiated exit.

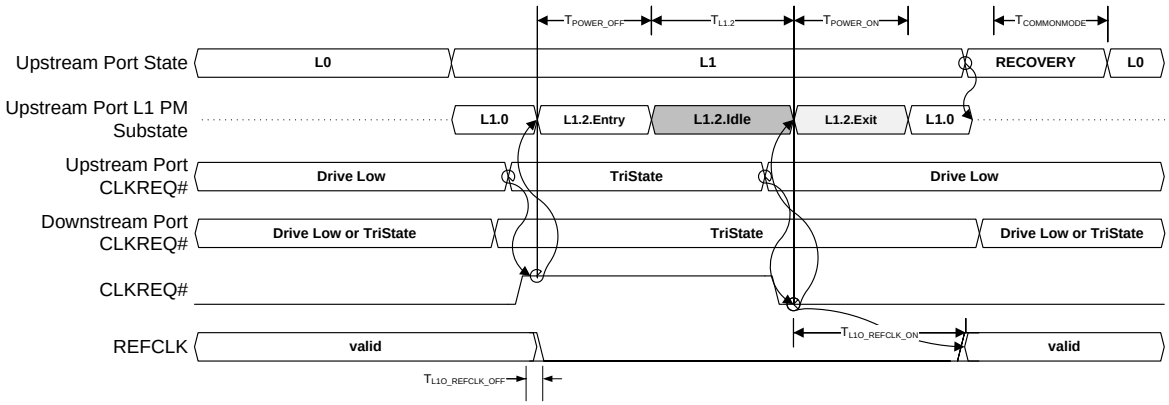


Figure 5-16: Example: L1.2 Waveforms Illustrating Upstream Port Initiated Exit

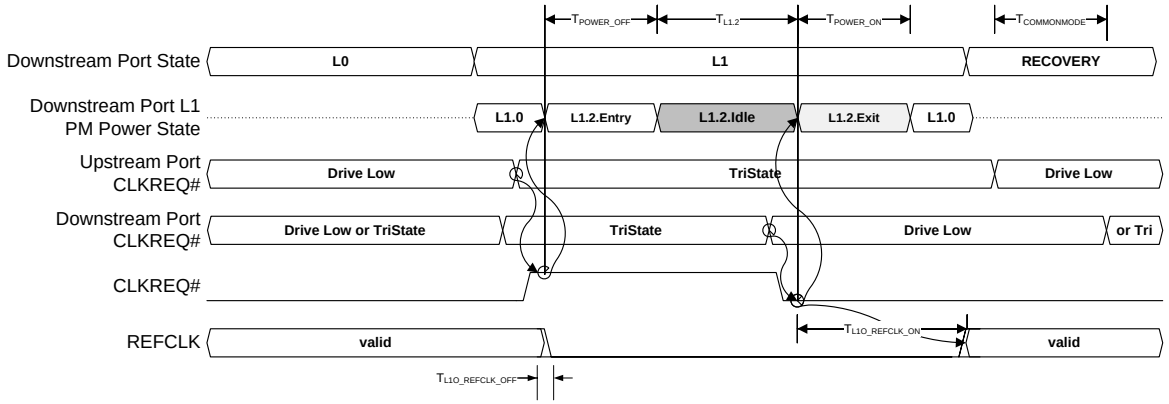


Figure 5-17: Example: L1.2 Waveforms Illustrating Downstream Port Initiated Exit

5.5.4 L1 PM Substates Configuration

L1 PM Substates is considered enabled on a Port when any combination of the ASPM L1.1 Enable, ASPM L1.2 Enable, PCI-PM L1.1 Enable and PCI-PM L1.2 Enable bits associated with that Port are Set.

An L1 PM Substate enable bit must only be Set in the Upstream and Downstream Ports on a Link when the corresponding supported capability bit is Set by both the Upstream and Downstream Ports on that Link, otherwise the behavior is undefined.

The Setting of any enable bit must be performed at the Downstream Port before the corresponding bit is permitted to be Set at the Upstream Port. If any L1 PM Substates enable bit is at a later time to be cleared, the enable bit(s) must be cleared in the Upstream Port before the corresponding enable bit(s) are permitted to be cleared in the Downstream Port.

If setting either or both of the enable bits for ASPM L1 PM Substates, both ports must be configured as described in this section while ASPM L1 is disabled.

If setting either or both of the enable bits for PCI-PM L1 PM Substates, both ports must be configured as described in this section while in D0.

Prior to setting either or both of the enable bits for L1.2, the values for T_{POWER_ON}, Common_Mode_Restore_Time, and, if the ASPM L1.2 Enable bit is to be Set, the LTR_L1.2_THRESHOLD (both Value and Scale fields) must be programmed.

The T_{POWER_ON} and Common_Mode_Restore_Time fields must be programmed to the appropriate values based on the components and AC coupling capacitors used in the connection linking the two components. The determination of these values is design implementation specific.

When both the ASPM L1.2 Enable and PCI-PM L1.2 Enable bits are cleared, it is not required to program the T_{POWER_ON}, Common_Mode_Restore_Time, and LTR_L1.2_THRESHOLD Value and Scale fields, and hardware must not rely on these fields to have any particular values.

When programming LTR_L1.2_THRESHOLD Value and Scale fields, identical values must be programmed in both Ports.

5.5.5 L1 PM Substates Timing Parameters

The following table defines the timing parameters associated with the L1.2 substates mechanism.

Table 5-11: Timing Parameters

Parameter	Description	Min	Max	Units
T_{POWER_OFF}	CLKREQ# deassertion to entry into the L1.2.Idle substate		2	µs
$T_{COMMONMODE}$	Restoration of Refclk to restoration of common mode established through active transmission of TS1 training sequences (see Section 5.5.3.3.1)	Programmable in range from 0 to 255		µs
$T_{L1O_REFCLK_OFF}$	CLKREQ# deassertion to Refclk reaching idle electrical state when entering L1.2	0	100	ns
$T_{L1O_REFCLK_ON}$	CLKREQ# assertion to Refclk valid when exiting L1.2	T_{POWER_ON}	LTR value advertized by the Endpoint	µs
T_{POWER_ON}	The minimum amount of time that each component must wait in L1.2.Exit after sampling CLKREQ# asserted before actively driving the interface to ensure no device is ever actively driving into an unpowered component.	Set in the L1 PM Substates Control 2 Register (range from 0 to 3100)		µs
$T_{L1.2}$	Time a Port must stay in L1.2 when CLKREQ# must remain inactive	4		µs

5.6 Auxiliary Power Support

The specific definition and requirements associated with auxiliary power are form-factor specific, and the terms "auxiliary power" and "Vaux" should be understood in reference to the specific form factor in use. The following text defines recruitments that apply in all form factors.

PCI Express PM provides a Aux Power PM Enable bit in the Device Control register that provides the means for enabling a Function to draw the maximum allowance of auxiliary current independent of its level of support for PME generation.

A Function requests Aux power allocation by specifying a non-zero value in the Aux_Current field of the PMC register. Refer to Chapter 7 for the Aux Power PM Enable register bit assignment, and access mechanism.

Allocation of Aux power using Aux Power PM Enable is determined as follows:

Aux Power PM Enable = 1b:

Aux power is allocated as requested in the Aux_Current field of the PMC register, independent of the PME_En bit in the PMSCR. The PME_En bit still controls the ability to master PME.

Aux Power PM Enable = 0b:

Aux power allocation is controlled by the PME_En bit as defined in Section 7.5.2.2.

The Aux Power PM Enable bit is sticky (see Section 7.4) so its state is preserved in the $D3_{cold}$ state, and is not affected by the transitions from the $D3_{cold}$ state to the $D0_{uninitialized}$ state.

5.7 Power Management System Messages and DLLPs

Table 5-12 defines the location of each PM packet in the PCI Express stack.

Table 5-12: Power Management System Messages and DLLPs

Packet	Type
PM_Enter_L1	DLLP
PM_Enter_L23	DLLP
PM_Active_State_Request_L1	DLLP
PM_Request_Ack	DLLP
PM_Active_State_Nak	Transaction Layer Message
PM_PME	Transaction Layer Message
PME_Turn_Off	Transaction Layer Message
PME_TO_Ack	Transaction Layer Message

For information on the structure of the power management DLLPs, refer to Section 3.5.

Power Management Messages follow the general rules for all Messages. Power Management Message fields follow the following rules:

- Length field is Reserved.
- Attribute field must be set to the default values (all 0's).
- Address field is Reserved.
- Requester ID – see Table 2-20 in Section 2.2.8.2.
- Traffic Class field must use the default class (TC0).

5.8 PCI Function Power State Transitions

All PCI-PM power management state changes are explicitly controlled by software except for Fundamental Reset which brings all Functions to the $D0_{uninitialized}$ state. Figure 5-18 shows all supported state transitions. The unlabeled arcs represent a software initiated state transition (Set Power State operation).

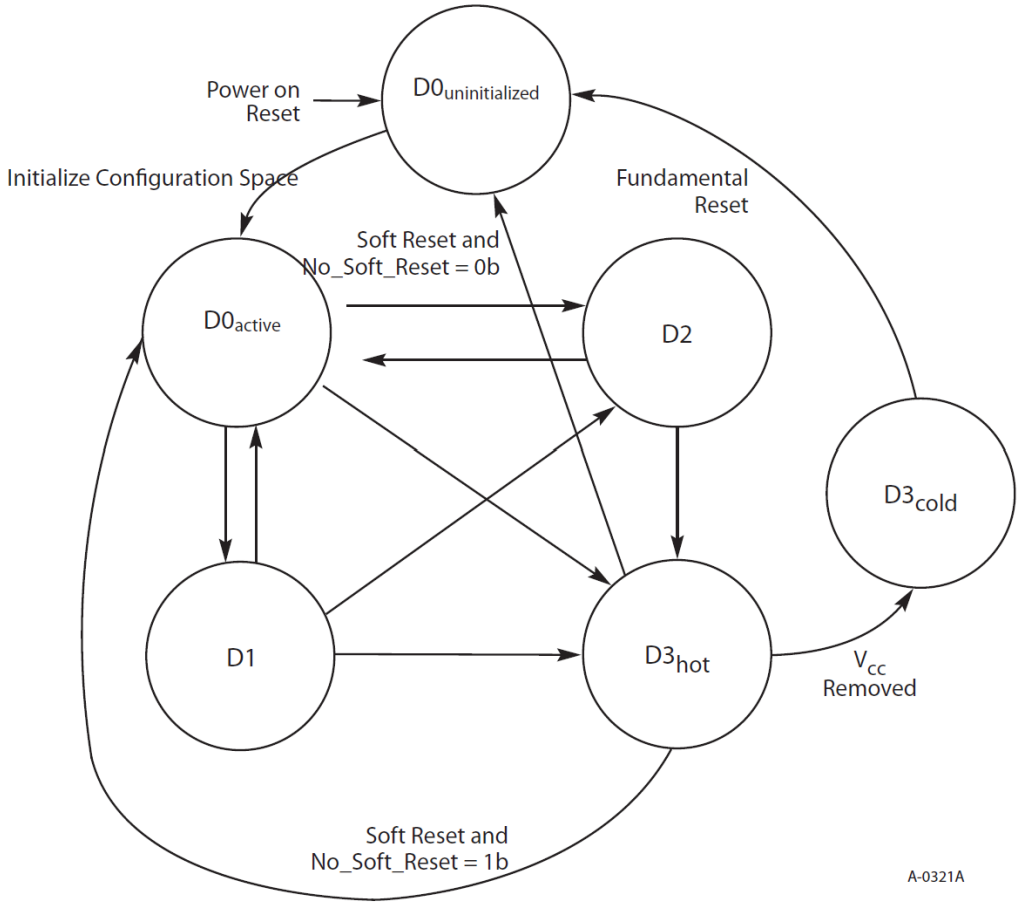


Figure 5-18: Function Power Management State Transitions

5.9 Function Power Management Policies

This section defines the behavior for Functions. Figure 5-19 illustrates the areas discussed in this section.

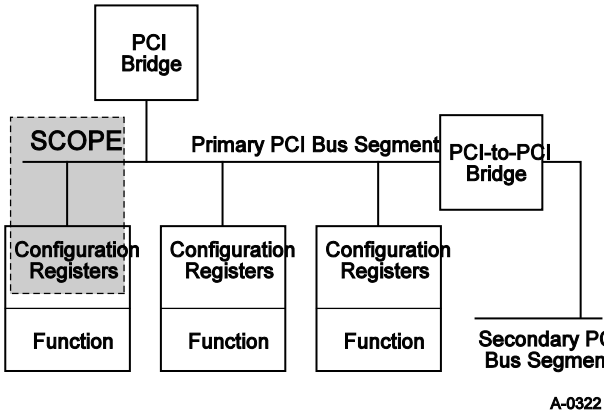


Figure 5-19: Non-Bridge Function Power Management Diagram

Tables 5-1 to 5-5 define the behavior for a Function while operating in each combination of bus and functional power management states.

The columns of Table 5-13 to Table 5-17 are defined as follows:

Function PCI-PM State Current Function power management state.

Context Configuration register and functional state information that are required to be valid for the given power management state. The registers that must remain valid, and the features that must remain available for a given class of device, are typically dictated by the corresponding Device-Class Power Management specification

Power Power consumption.

Access Delay

Restore Time The total time from when a Function transitions from its current power management state to the fully configured $D0_{active}$ state. (Measurement beginning from either a write to the function's *PMCSR* or a bus segment reset.)

Actions to Function Valid PCIe transactions that can be conducted with the Function as the target of the transaction.

Actions from Function Valid PCIe transactions and/or operations that can be initiated by the Function.

Table 5-13: D0 Power Management Policies

PCI Bus PM State	Function PM State	Context	Power	Access Delay	Restore Time	Actions to Function	Actions from Function
B0	$D0_{uninitialized}$	PME Context *	<10 W	None	None	configuration transactions	None
B0	$D0_{active}$	Full	Full	None	None	Any PCIe transaction	Any PCIe transaction or interrupt, PME*
B1-B3	$D0_{active}$	N/A**	N/A**	N/A**	N/A**	N/A**	N/A**

Notes:

* If PME is supported in this state.

** This combination of Function and power management states is not allowed.

Table 5-14: D1 Power Management Policies

PCI Bus PM State	Function PCI-PM State	Context	Power	Access Delay	Restore Time	Actions to Function	Actions from Function
B0	D1	Configur-ation PME Context, Device-Class Specific*	≤ $D0_{uninitialized}$	None	Device-Class specific	configuration transactions and Device-Class specific	PME*
B1	D1	Configur-ation PME Context, Device-Class Specific*	≤ $D0_{uninitialized}$	Bus restorati on time	Device-Class specific	None	PME only*
B2-B3	D1	N/A**	N/A**	N/A**	N/A**	N/A**	N/A**

Notes:

* If PME is supported in this state.

** This combination of Function and power management states is not allowed.

Table 5-15: D2 Power Management Policies

PCI Bus PM State	Function PCI-PM State	Context	Power	Access Delay	Restore Time	Actions to Function	Actions from Function
B0	D2	Configur-ation PME Context, Device-Class Specific*	≤ Next lower supported PM state or ≤ $D0_{uninitialized}$	200 μs (Note 1)	Device-Class specific	configurat ion transactio ns	PME only*
B1	D2	Configur-ation PME Context, Device-Class Specific*	≤ Next lower supported PM state or ≤ $D0_{uninitialized}$	Greater of either the bus restoration time or 200 μs (Note 2)	Device-Class specific	none	PME only*
B2	D2	Configur-ation PME Context, Device-Class Specific*	≤ Next lower supported PM state or ≤ $D0_{uninitialized}$	Greater of either the bus restoration time or 200 μs (Note 2)	Device-Class specific	none	PME only*
B3	D2	N/A**	N/A**	N/A**	N/A**	N/A**	N/A**

Notes:

1. This condition is not typical. It specifies the case where the system software has programmed the Function's PowerState field and then immediately decides to change its power state again. Typically, the state transition recovery time will have expired prior to a power state change request by software.
2. The more typical case where the bus must first be restored to B0 before being able to access the Function residing on the bus to request a change of its power state. State transition recovery time begins from the time of the last write to the Function's PowerState field. In this case, the bus restoration time is dictated by state transition recovery times incurred in programming the bus's Originating Device to D0 which then transitions its bus to B0. Bus restoration time is typically the deciding factor in access delay for this case (refer to Section 5.6.1).

* If PME is supported and enabled in this state, all context is saved. Configuration context is always saved.

** This combination of Function and power management states is not allowed.

Table 5-16: $D3_{hot}$ Power Management Policies

PCI Bus PM State	Function PCI-PM State	Context	Power	Access Delay	Restore Time	Actions to Function	Actions from Function
B0	$D3_{hot}$	Device-Class specific, PME Configur-ation Context***	≤ Next lower supported PM state or ≤ $D0_{uninitialized}$	10 ms (Note 1)	Device-Class specific	configurati on transaction s	PME only*
B1	$D3_{hot}$	Device-Class specific, PME Configur-ation Context***	≤ Next lower supported PM state or ≤ $D0_{uninitialized}$	Greater of either the bus restoration time or 10 ms (Note 2)	Device-Class specific	None	PME only*
B2	$D3_{hot}$	Device-Class specific, PME Configur-ation Context***	≤ next lower supported PM state or ≤ $D0_{uninitialized}$	Greater of either the bus restoration time or 10 ms (Note 2)	Device-Class specific	None	PME only*
B3	$D3_{hot}$	N/A**	N/A**	N/A**	N/A**	N/A**	N/A**

Notes:

1. This condition is not typical. It specifies the case where the system software has programmed the Function's PowerState field and then immediately decides to change its power state again. Typically, the state transition recovery time will have expired prior to a power state change request by software.
2. The more typical case where the bus must first be restored to B0 before being able to access the Function residing on the bus to request a change of its power state. State transition recovery time begins from the time of the last write to the Function's PowerState field. In this case, the bus restoration time is dictated by state transition recovery times incurred in programming the bus's Originating Device to D0 which then transitions its bus to B0. Bus restoration time is typically the deciding factor in access delay for this case (refer to Section 5.6.1).

* If PME is supported in this state.

** This combination of Function and power management states is not allowed.

***Configuration Context may be preserved if supporting No_Soft_Reset.

Table 5-17: $D3_{cold}$ Power Management Policies

PCI Bus PM State	Function PCI-PM State	Context	Power	Access Delay	Restore Time	Actions to Function	Actions from Function
B0-B2	$D3_{cold}$	N/A***	N/A***	N/A***	N/A***	N/A***	N/A***
B3	$D3_{cold}$	PME Context only*	No power from the bus	N/A	Full context restore or boot latency	Bus Segment Reset only	PME only*
B3	Legacy PCI function (D3)	None	No power	N/A	Full context restore or boot latency	Bus Segment Reset only	None

Notes:

* If PME is supported and enabled in this state.

***Implies device specific or slot specific power supplies which are outside the scope of this specification.

When in D2 or $D3_{hot}$, a Function must not respond to transactions targeting its I/O or memory spaces or assert a functional interrupt request.

A Function cannot tell the state of its PCI bus; therefore, it must always be ready to accept a configuration access when in D1, D2, or $D3_{hot}$.

5.9.1 State Transition Recovery Time Requirements

Table 5-18 shows the minimum recovery times (delays) that must be guaranteed, by hardware in some cases and by system software in others, between the time that a Function is programmed to change state and the time that the function is next accessed (including Configuration Space). Note that for bridges, this delay also constitutes a minimum delay between when the bridge's state is upgraded to D0 and when any Function on the bus that it originates can be accessed.

Table 5-18: PCI Function State Transition Delays

Initial State	Next State	Minimum System Software Guaranteed Delays
D0	D1	0
D0 or D1	D2	200 μs
D0, D1 or D2	$D3_{hot}$	10 ms
D1	D0	0
D2	D0	200 μs
$D3_{hot}$	D0	10 ms

5.10 PCI Bridges and Power Management

With power management under the direction of the operating system, each class of Functions must have a clearly defined criteria for feature availability as well as what functional context must be preserved when operating in each of the power management states. Some example Device-Class specifications have been proposed as part of the ACPI specification for various Functions ranging from audio to network add-in cards. While defining Device-Class specific behavioral policies for most Functions is outside the scope of this specification, defining the required behavior for PCI bridge functions is within the scope of this specification. The definitions here apply to all three types of PCIe Bridges:

- Host bridge, PCI Express to expansion bus bridge, or other ACPI enumerated bridge
- Switches
- PCI Express to PCI bridge
- PCI-to-CardBus bridge

The mechanisms for controlling the state of these Functions vary somewhat depending on which type of Originating Device is present. The following sections describe how these mechanisms work for the three types of bridges.

This chapter details the power management policies for PCI Express Bridge Functions. The PCI Express Bridge Function can be characterized as an Originating Device with a secondary bus downstream of it. The relationship of the bridge function's power management state to that of its secondary bus has been mentioned in Section 4.7.1 and will be developed further in this chapter.

The shaded regions in Figure 5-20 illustrate what is discussed in this chapter.

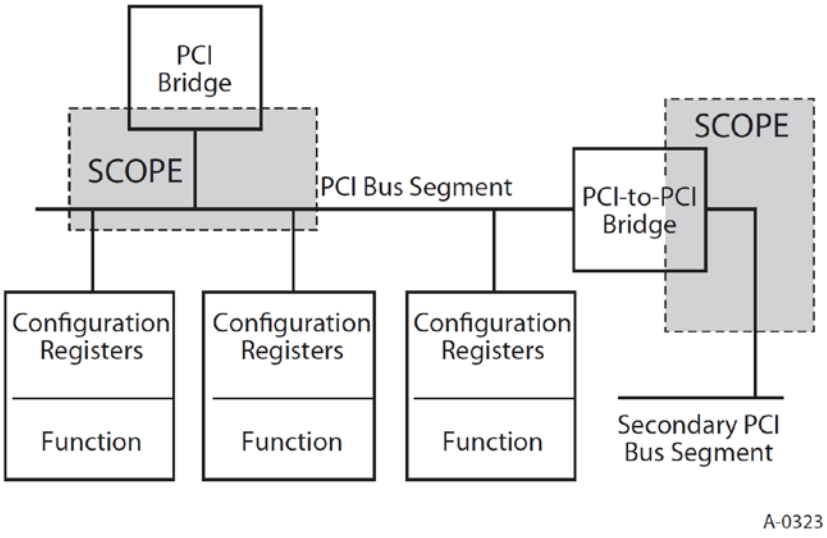


Figure 5-20: PCI Express Bridge Power Management Diagram

As can be seen from Figure 5-20, the PCI Express Bridge behavior described in this chapter is common, from the perspective of the operating system, to host bridges, Switches, and PCI Express to PCI bridges.

It is the responsibility of the system software to ensure that only valid, workable combinations of bus and downstream Function power management states are used for a given bus and all Functions residing on that bus.

5.10.1 Switches and PCI Express to PCI Bridges

The power management policies for the secondary bus of a Switch or PCI Express to PCI bridge are identical to those defined for any Bridge Function.

The BPCC_En and B2_B3# bus power/clock control fields in the Bridge Function's PMCSR_BSE register support the same functionality as for any other Bridges.

5.11 Power Management Events

There are two varieties of Power Management Events:

- Wakeup Events
- PME Generation

A Wakeup Event is used to request that power be turned on.

A PME Generation Event is used to identify to the system the Function requesting that power be turned on.

In conventional PCI, both events are associated with the PME# signal. The PME# signal is asserted by a Function to request a change in its power management state. When the PME_En bit is Set and the event occurs, the Function sets the PME_Status bit and asserts the PME# signal. It keeps the PME# signal asserted until either the PME_En bit or the PME_Status are Cleared (typically by software).

In PCI Express, the Wakeup Event is associated with the WAKE# signal. If supported, the WAKE# signal is defined in the associated form factor specification and is used by a Function to request a change in its PCI-PM power management state when the Function is in $D3_{cold}$ and PME_En is Set.

In PCI Express, after main power has been restored and the Link is trained, the Function(s) that initiated the wakeup (e.g., that asserted WAKE#), sends a PM_PME Message to the Root Complex. The PM_PME Message provides the Root Complex with the identity of the requesting Function(s) without requiring software to poll for the PME_Status bit being Set.

This page intentionally blank

6

6 System Architecture

This chapter addresses various aspects of PCI Express interconnect architecture in a platform context.

6.1 Interrupt and PME Support

The PCI Express interrupt model supports two mechanisms:

- INTx emulation
- Message Signaled Interrupt (MSI/MSI-X)

For legacy compatibility, PCI Express provides a PCI INTx emulation mechanism to signal interrupts to the system interrupt controller (typically part of the Root Complex). This mechanism is compatible with existing PCI software, and provides the same level and type of service as the corresponding PCI interrupt signaling mechanism and is independent of system interrupt controller specifics. This legacy compatibility mechanism allows boot device support without requiring complex BIOS-level interrupt configuration/control service stacks. It virtualizes PCI physical interrupt signals by using an in-band signaling mechanism.

In addition to PCI INTx compatible interrupt emulation, PCI Express requires support of MSI or MSI-X or both. The PCI Express MSI and MSI-X mechanisms are compatible with those originally defined in the *PCI Local Bus Specification*.

6.1.1 Rationale for PCI Express Interrupt Model

PCI Express takes an evolutionary approach from PCI with respect to interrupt support.

As required for PCI/PCI-X interrupt mechanisms, each device Function is required to differentiate between INTx and MSI/MSI-X modes of operation. The device complexity required to support both schemes is no different than that for PCI/PCI-X devices. The advantages of this approach include:

- Compatibility with existing PCI Software Models
- Direct support for boot devices
- Easier End of Life (EOL) for INTx legacy mechanisms.

The existing software model is used to differentiate INTx vs. MSI/MSI-X modes of operation; thus, no special software support is required for PCI Express.

6.1.2 PCI-compatible INTx Emulation

PCI Express supports the PCI interrupts as defined in the P*CI Local Bus Specification* including the Interrupt Pin and Interrupt Line registers of the PCI Configuration Space for PCI device Functions. PCI Express devices support these registers for backwards compatibility; actual interrupt signaling uses in-band Messages rather than being signaled using physical pins.

Two types of Messages are defined, Assert_INTx and Deassert_INTx, for emulation of PCI INTx signaling, where x is A, B, C, and D for respective PCI interrupt signals. These Messages are used to provide "virtual wires" for signaling interrupts across a Link. Switches collect these virtual wires and present a combined set at the Switch's Upstream Port. Ultimately, the virtual wires are routed to the Root Complex which maps the virtual wires to system interrupt resources. Devices must use assert/deassert Messages in pairs to emulate PCI interrupt level-triggered signaling. Actual mapping of PCI Express INTx emulation to system interrupts is implementation specific as is mapping of physical interrupt signals in conventional PCI.

The legacy INTx emulation mechanism may be deprecated in a future version of this specification.

6.1.3 INTx Emulation Software Model

The software model for legacy INTx emulation matches that of PCI. The system BIOS reporting of chipset/platform interrupt mapping and the association of each device Function's interrupt with PCI interrupt lines is handled in exactly the same manner as with conventional PCI systems. Legacy software reads from each device Function's Interrupt Pin register to determine if the Function is interrupt driven. A value between 01h and 04h indicates that the Function uses an emulated interrupt pin to generate an interrupt.

Note that similarly to physical interrupt signals, the INTx emulation mechanism may potentially cause spurious interrupts that must be handled by the system software.

6.1.4 MSI and MSI-X Operation

Message Signaled Interrupts (MSI) is an optional feature that enables a device Function to request service by writing a system-specified data value to a system-specified address (using a DWORD Memory Write transaction). System software initializes the message address and message data (from here on referred to as the "vector") during device configuration, allocating one or more vectors to each MSI-capable Function.

Interrupt latency (the time from interrupt signaling to interrupt servicing) is system dependent. Consistent with current interrupt architectures, Message Signaled Interrupts do not provide interrupt latency time guarantees.

MSI-X defines a separate optional extension to basic MSI functionality. Compared to MSI, MSI-X supports a larger maximum number of vectors per Function, the ability for software to control aliasing when fewer vectors are allocated than requested, plus the ability for each vector to use an independent address and data value, specified by a table that resides in Memory Space. However, most of the other characteristics of MSI-X are identical to those of MSI.

For the sake of software backward compatibility, MSI and MSI-X use separate and independent Capability structures. On Functions that support both MSI and MSI-X, system software that

supports only MSI can still enable and use MSI without any modification. MSI functionality is managed exclusively through the MSI Capability structure, and MSI-X functionality is managed exclusively through the MSI-X Capability structure.

A Function is permitted to implement both MSI and MSI-X, but system software is prohibited from enabling both at the same time. If system software enables both at the same time, the result is undefined.

All PCI Express device Functions that are capable of generating interrupts must support MSI or MSI-X or both. The MSI and MSI-X mechanisms deliver interrupts by performing Memory Write transactions. MSI and MSI-X are edge-triggered interrupt mechanisms; neither the *PCI Local Bus Specification* nor this specification support level-triggered MSI/MSI-X interrupts. Certain PCI devices and their drivers rely on INTx-type level-triggered interrupt behavior (addressed by the PCI Express legacy INTx emulation mechanism). To take advantage of the MSI or MSI-X capability and edge-triggered interrupt semantics, these devices and their drivers may have to be redesigned.

MSI and MSI-X each support Per-Vector Masking (PVM). PVM is an optional[74] extension to MSI, and a standard feature with MSI-X. A Function that supports the PVM extension to MSI is backward compatible with system software that is unaware of the extension. MSI-X also supports a Function Mask bit, which when Set masks all of the vectors associated with a Function.

A Legacy Endpoint that implements MSI is required to support either the 32-bit or 64-bit Message Address version of the MSI Capability structure. A PCI Express Endpoint that implements MSI is required to support the 64-bit Message Address version of the MSI Capability structure.

The Requester of an MSI/MSI-X transaction must set the No Snoop and Relaxed Ordering attributes of the Transaction Descriptor to 0b. A Requester of an MSI/MSI-X transaction is permitted to Set the ID-Based Ordering (IDO) attribute if use of the IDO attribute is enabled.

Note that, unlike INTx emulation Messages, MSI/MSI-X transactions are not restricted to the TC0 traffic class.

IMPLEMENTATION NOTE

Synchronization of Data Traffic and Message Signaled Interrupts

MSI/MSI-X transactions are permitted to use the TC that is most appropriate for the device's programming model. This is generally the same TC as is used to transfer data; for legacy I/O, TC0 should be used.

If a device uses more than one TC, it must explicitly ensure that proper synchronization is maintained between data traffic and interrupt Message(s) not using the same TC. Methods for ensuring this synchronization are implementation specific. One option is for a device to issue a zero-length Read (as described in Section 2.2.5) using each additional TC used for data traffic prior to issuing the MSI/MSI-X transaction. Other methods are also possible. Note, however, that platform software (e.g., a device driver) is generally only capable of issuing transactions using TC0.

[74] Exception: Within an SR-IOV Device, any PFs or VFs that implement MSI must implement MSI PVM.

Within a device, different Functions are permitted to implement different sets of the MSI/MSI-X/INTx interrupt mechanisms, and system software manages each Function's interrupt mechanisms independently.

6.1.4.1 *MSI Configuration*

In this section, all register and field references are in the context of the MSI Capability structure.

System software reads the Message Control register to determine the Function's MSI capabilities.

System software reads the Multiple Message Capable field (bits 3-1 of the Message Control register) to determine the number of requested vectors. MSI supports a maximum of 32 vectors per Function. System software writes to the Multiple Message Enable field (bits 6-4 of the Message Control register) to allocate either all or a subset of the requested vectors. For example, a Function can request four vectors and be allocated either four, two, or one vector. The number of vectors requested and allocated is aligned to a power of two (that is, a Function that requires three vectors must request four).

If the Per-Vector Masking Capable bit (bit 8 of the Message Control register) is Set and system software supports Per-Vector Masking, system software may mask one or more vectors by writing to the Mask Bits register.

If the 64-bit Address Capable bit (bit 7 of the Message Control register) is Set, system software initializes the MSI Capability structure's Message Address register (specifying the lower 32 bits of the message address) and the Message Upper Address register (specifying the upper 32 bits of the message address) with a system-specified message address. System software may program the Message Upper Address register to zero so that the Function uses a 32-bit address for the MSI transaction. If this bit is Clear, system software initializes the MSI Capability structure's Message Address register (specifying a 32-bit message address) with a system specified message address.

System software initializes the MSI Capability structure's Message Data register with the lower 16 bits of a system specified data value. When the Extended Message Data Capable bit is Clear, care must be taken to initialize only the Message Data register (i.e., a 2-byte value) and not modify the upper two bytes of that DWORD location.

If the Extended Message Data Capable bit is Set and system software supports 32-bit vector values, system software may initialize the MSI capability structure's Extended Message Data register with the upper 16 bits of a system specified data value, and then Set the Extended Message Data Enable bit.

6.1.4.2 *MSI-X Configuration*

In this section, all register and field references are in the context of the MSI-X Capability, MSI-X Table, and MSI-X PBA structures.

System software allocates address space for the Function's standard set of Base Address registers and sets the registers accordingly. One of the Function's Base Address registers includes address space for the MSI-X Table, though the system software that allocates address space does not need to be aware of which Base Address register this is, or the fact the address space is used for the MSI-X Table. The same or another Base Address register includes address space for the MSI-X PBA, and the same point regarding system software applies.

Depending upon system software policy, system software, device driver software, or each at different times or environments may configure a Function's MSI-X Capability and table structures with suitable vectors. For example, a booting environment will likely require only a single vector, whereas a normal operating system environment for running applications may benefit from multiple vectors if the Function supports an MSI-X Table with multiple entries. For the remainder of this section, "software" refers to either system software or device driver software.

Software reads the Table Size field from the Message Control register to determine the MSI-X Table size. The field encodes the number of table entries as N-1, so software must add 1 to the value read from the field to calculate the number of table entries N. MSI-X supports a maximum table size of 2048 entries.

Software calculates the base address of the MSI-X Table by reading the 32-bit value from the Table Offset/Table BIR register, masking off the lower 3 Table BIR bits, and adding the remaining QWORD-aligned 32-bit Table offset to the address taken from the Base Address register indicated by the Table BIR. Software calculates the base address of the MSI-X PBA using the same process with the PBA Offset/PBA BIR register.

For each MSI-X Table entry that will be used, software fills in the Message Address field, Message Upper Address field, Message Data field, and Vector Control field. The Vector Control field may contain optional Steering Tag fields. Software must not modify the Address, Data, or Steering Tag fields of an entry while it is unmasked. Refer to Section 6.1.4.5 for details.

IMPLEMENTATION NOTE

Special Considerations for QWORD Accesses

Software is permitted to fill in MSI-X Table entry DWORD fields individually with DWORD writes, or software in certain cases is permitted to fill in appropriate pairs of DWORDs with a single QWORD write. Specifically, software is always permitted to fill in the Message Address and Message Upper Address fields with a single QWORD write. If a given entry is currently masked (via its Mask bit or the Function Mask bit), software is permitted to fill in the Message Data and Vector Control fields with a single QWORD write, taking advantage of the fact the Message Data field is guaranteed to become visible to hardware no later than the Vector Control field. However, if software wishes to mask a currently unmasked entry (without Setting the Function Mask bit), software must Set the entry's Mask bit using a DWORD write to the Vector Control field, since performing a QWORD write to the Message Data and Vector Control fields might result in the Message Data field being modified before the Mask bit in the Vector Control field becomes Set.

For potential use by future specifications, the Reserved bits in the Vector Control field must have their default values preserved by software. If software does not preserve their values, the result is undefined.

For each MSI-X Table entry that software chooses not to configure for generating messages, software can simply leave the entry in its default state of being masked.

Software is permitted to configure multiple MSI-X Table entries with the same vector, and this may indeed be necessary when fewer vectors are allocated than requested.

IMPLEMENTATION NOTE

Handling MSI-X Vector Shortages

For the case where fewer vectors are allocated to a Function than desired, software-controlled aliasing as enabled by MSI-X is one approach for handling the situation. For example, if a Function supports five queues, each with an associated MSI-X table entry, but only three vectors are allocated, the Function could be designed for software still to configure all five table entries, assigning one or more vectors to multiple table entries. Software could assign the three vectors {A,B,C} to the five entries as ABCCC, ABBCC, ABCBA, or other similar combinations.

Alternatively, the Function could be designed for software to configure it (using a device specific mechanism) to use only three queues and three MSI-X table entries. Software could assign the three vectors {A,B,C} to the five entries as ABC--, A-B-C, A--CB, or other similar combinations.

6.1.4.3 *Enabling Operation*

To maintain backward compatibility, the MSI Enable bit in the MSI Message Control register and the MSI-X Enable bit in the MSI-X Message Control register are each Clear by default (MSI and MSI-X are both disabled). System configuration software Sets one of these bits to enable either MSI or MSI-X, but never both simultaneously. Behavior is undefined if both MSI and MSI-X are enabled simultaneously. A device driver is prohibited from writing this bit to mask a Function's service request. While enabled for MSI or MSI-X operation, a Function is prohibited from using INTx interrupts (if implemented) to request service (MSI, MSI-X, and INTx are mutually exclusive).

6.1.4.4 *Sending Messages*

Once MSI or MSI-X is enabled (the appropriate bit in one of the Message Control registers is Set), and one or more vectors is unmasked, the Function is permitted to send messages. To send a message, a Function does a DWORD Memory Write to the appropriate message address with the appropriate message data.

For MSI when the Extended Message Data Enable bit is Clear, the DWORD that is written is made up of the value in the MSI Message Data register in the lower two bytes and zeroes in the upper two bytes. For MSI when the Extended Message Data Enable bit is Set, the DWORD that is written is made up of the value in the MSI Message Data register in the lower two bytes and the value in the MSI Extended Message Data register in the upper two bytes.

For MSI, if the Multiple Message Enable field (bits 6-4 of the MSI Message Control register) is non-zero, the Function is permitted to modify the low order bits of the message data to generate multiple vectors. For example, a Multiple Message Enable encoding of 010b indicates the Function is permitted to modify message data bits 1 and 0 to generate up to four unique vectors. If the Multiple Message Enable field is 000b, the Function is not permitted to modify the message data.

For MSI-X, the MSI-X Table contains at least one entry for every allocated vector, and the 32-bit Message Data field value from a selected table entry is used in the message without any modification to the low-order bits by the Function.

How a Function uses multiple vectors (when allocated) is device dependent. A Function must handle being allocated fewer vectors than requested.

6.1.4.5 *Per-vector Masking and Function Masking*

Per-Vector Masking (PVM) is an optional[75] feature with MSI, and a standard feature in MSI-X.

Function Masking is a standard feature in MSI-X. When the MSI-X Function Mask bit is Set, all of the Function's entries must behave as being masked, regardless of the per-entry Mask bit values. Function Masking is not supported in MSI, but software can readily achieve a similar effect by Setting all MSI Mask bits using a single DWORD write.

PVM in MSI-X is controlled by a Mask bit in each MSI-X Table entry. While more accurately termed "per-entry masking", masking an MSI-X Table entry is still referred to as "vector masking" so similar descriptions can be used for both MSI and MSI-X. However, since software is permitted to program the same vector (a unique Address/Data pair) into multiple MSI-X table entries, all such entries must be masked in order to guarantee the Function will not send a message using that Address/Data pair.

For MSI and MSI-X, while a vector is masked, the Function is prohibited from sending the associated message, and the Function must Set the associated Pending bit whenever the Function would otherwise send the message. When software unmasks a vector whose associated Pending bit is Set, the Function must schedule sending the associated message, and Clear the Pending bit as soon as the message has been sent. Note that Clearing the MSI-X Function Mask bit may result in many messages needing to be sent.

If a masked vector has its Pending bit Set, and the associated underlying interrupt events are somehow satisfied (usually by software though the exact manner is Function-specific), the Function must Clear the Pending bit, to avoid sending a spurious interrupt message later when software unmasks the vector. However, if a subsequent interrupt event occurs while the vector is still masked, the Function must again Set the Pending bit.

Software is permitted to mask one or more vectors indefinitely, and service their associated interrupt events strictly based on polling their Pending bits. A Function must Set and Clear its Pending bits as necessary to support this "pure polling" mode of operation.

For MSI-X, a Function is permitted to cache Address and Data values from unmasked MSI-X Table entries. However, anytime software unmasks a currently masked MSI-X Table entry either by Clearing its Mask bit or by Clearing the Function Mask bit, the Function must update any Address or Data values that it cached from that entry. If software changes the Address or Data value of an entry while the entry is unmasked, the result is undefined.

[75] Exception: Within an SR-IOV Device, any PFs or VFs that implement MSI must implement MSI PVM.

IMPLEMENTATION NOTE

Per Vector Masking with MSI/MSI-X

Devices and drivers that use MSI or MSI-X have the challenge of coordinating exactly when new interrupt messages are generated. If hardware fails to send an interrupt message that software expects, an interrupt event might be "lost". If hardware sends an interrupt message that software is not expecting, a "spurious" interrupt might result.

Per-Vector Masking (PVM) can be used to assist in this coordination. For example, when a software interrupt service routine begins, it can mask the vector to help avoid "spurious" interrupts. After the interrupt service routine services all the interrupt conditions that it is aware of, it can unmask the vector. If any interrupt conditions remain, hardware is required to generate a new interrupt message, guaranteeing that no interrupt events are lost.

PVM is a standard feature with MSI-X and an optional[76] feature for MSI. For devices that implement MSI, implementing PVM as well is highly recommended.

6.1.4.6 *Hardware/Software Synchronization*

If a Function sends messages with the same vector multiple times before being acknowledged by software, only one message is guaranteed to be serviced. If all messages must be serviced, a device driver handshake is required. In other words, once a Function sends Vector A, it cannot send Vector A again until it is explicitly enabled to do so by its device driver (provided all messages must be serviced). If some messages can be lost, a device driver handshake is not required. For Functions that support multiple vectors, a Function can send multiple unique vectors and is guaranteed that each unique message will be serviced. For example, a Function can send Vector A followed by Vector B without any device driver handshake (both Vector A and Vector B will be serviced).

[76] Exception: Within an SR-IOV Device, any PFs or VFs that implement MSI must implement MSI PVM

IMPLEMENTATION NOTE

Servicing MSI and MSI-X Interrupts

When system software allocates fewer MSI or MSI-X vectors to a Function than it requests, multiple interrupt sources within the Function, each desiring a unique vector, may be required to share a single vector. Without proper handshakes between hardware and software, hardware may send fewer messages than software expects, or hardware may send what software considers to be extraneous messages.

A rather sophisticated but resource-intensive approach is to associate a dedicated event queue with each allocated vector, with producer and consumer pointers for managing each event queue. Such event queues typically reside in host memory. The Function acts as the producer and software acts as the consumer. Multiple interrupt sources within a Function may be assigned to each event queue as necessary. Each time an interrupt source needs to signal an interrupt, the Function places an entry on the appropriate event queue (assuming there's room), updates a copy of the producer pointer (typically in host memory), and sends an interrupt message with the associated vector when necessary to notify software that the event queue needs servicing. The interrupt service routine for a given event queue processes all entries it finds on its event queue, as indicated by the producer pointer. Each event queue entry identifies the interrupt source and possibly additional information about the nature of the event. The use of event queues and producer/consumer pointers can be used to guarantee that interrupt events won't get dropped when multiple interrupt sources are forced to share a vector. There's no need for additional handshaking between sending multiple messages associated with the same event queue, to guarantee that every message gets serviced. In fact, various standard techniques for "interrupt coalescing" can be used to avoid sending a separate message for every event that occurs, particularly during heavy bursts of events.

In more modest implementations, the hardware design of a Function's MSI or MSI-X logic sends a message any time a transition to assertion would have occurred on the virtual INTx wire if MSI or MSI-X had not been enabled. For example, consider a scenario in which two interrupt events (possibly from distinct interrupt sources within a Function) occur in rapid succession. The first event causes a message to be sent. Before the interrupt service routine has had an opportunity to service the first event, the second event occurs. In this case, only one message is sent, because the first event is still active at the time the second event occurs (a virtual INTx wire signal would have had only one transition to assertion).

One handshake approach for implementations like the above is to use standard Per-Vector Masking, and allow multiple interrupt sources to be associated with each vector. A given vector's interrupt service routine Sets the vector's Mask bit before it services any associated interrupting events and Clears the Mask bit after it has serviced all the events it knows about. (This could be any number of events.) Any occurrence of a new event while the Mask bit is Set results in the Pending bit being Set. If one or more associated events are still pending at the time the vector's Mask bit is Cleared, the Function immediately sends another message.

A handshake approach for MSI Functions that do not implement Per-Vector Masking is for a vector's interrupt service routine to re-inspect all of the associated interrupt events after Clearing what is presumed to be the last pending interrupt event. If another event is found to be active, it is serviced in the same interrupt service routine invocation, and the complete re-inspection is repeated until no pending events are found. This ensures that if an additional interrupting event occurs

before a previous interrupt event is Cleared, whereby the Function does not send an additional interrupt message, that the new event is serviced as part of the current interrupt service routine invocation.

This alternative has the potential side effect of one vector's interrupt service routine processing an interrupting event that has already generated a new interrupt message. The interrupt service routine invocation resulting from the new message may find no pending interrupt events. Such occurrences are sometimes referred to as spurious interrupts, and software using this approach must be prepared to tolerate them.

An MSI or MSI-X message, by virtue of being a Posted Request, is prohibited by transaction ordering rules from passing Posted Requests sent earlier by the Function. The system must guarantee that an interrupt service routine invoked as a result of a given message will observe any updates performed by Posted Requests arriving prior to that message. Thus, the interrupt service routine of a device driver is not required to read from a device register in order to ensure data consistency with previous Posted Requests. However, if multiple MSI-X Table entries share the same vector, the interrupt service routine may need to read from some device specific register to determine which interrupt sources need servicing.

6.1.4.7 *Message Transaction Reception and Ordering Requirements*

As with all Memory Write transactions, the device that includes the target of the interrupt message (the interrupt receiver) is required to complete all interrupt message transactions as a Completer without requiring other transactions to complete first as a Requester. In general, this means that the message receiver must complete the interrupt message transaction independent of when the CPU services the interrupt. For example, each time the interrupt receiver receives an interrupt message, it could Set a bit in an internal register indicating that this message had been received and then complete the transaction on the bus. The appropriate interrupt service routine would later be dispatched because this bit was Set. The message receiver would not be allowed to delay the completion of the interrupt message on the bus pending acknowledgement from the processor that the interrupt was being serviced. Such dependencies can lead to deadlock when multiple devices send interrupt messages simultaneously.

Although interrupt messages remain strictly ordered throughout the PCI Express Hierarchy, the order of receipt of the interrupt messages does not guarantee any order in which the interrupts will be serviced. Since the message receiver must complete all interrupt message transactions without regard to when the interrupt was actually serviced, the message receiver will generally not maintain any information about the order in which the interrupts were received. This is true both of interrupt messages received from different devices and multiple messages received from the same device. If a device requires one interrupt message to be serviced before another, the device must not send the second interrupt message until the first one has been serviced.

6.1.5 PME Support

PCI Express supports power management events from native PCI Express devices as well as PME-capable PCI devices.

PME signaling is accomplished using an in-band Transaction Layer PME Message (PM_PME) as described in Chapter 5.

6.1.6 Native PME Software Model

PCI Express-aware software can enable a mode where the Root Complex signals PME via an interrupt. When configured for native PME support, a Root Port receives the PME Message and sets the PME Status bit in its Root Status register. If software has set the PME Interrupt Enable bit in the Root Control register to 1b, the Root Port then generates an interrupt.

If the Root Port is enabled for level-triggered interrupt signaling using the INTx messages, the virtual INTx wire must be asserted whenever and as long as all of the following conditions are satisfied:

- The Interrupt Disable bit in the Command register is set to 0b.
- The PME Interrupt Enable bit in the Root Control register is set to 1b.
- The PME Status bit in the Root Status register is set.

Note that all other interrupt sources within the same Function will assert the same virtual INTx wire when requesting service.

If the Root Port is enabled for edge-triggered interrupt signaling using MSI or MSI-X, an interrupt message must be sent every time the logical AND of the following conditions transitions from FALSE to TRUE:

- The associated vector is unmasked (not applicable if MSI does not support PVM).
- The PME Interrupt Enable bit in the Root Control register is set to 1b.
- The PME Status bit in the Root Status register is set.

Note that PME and Hot-Plug Event interrupts (when both are implemented) always share the same MSI or MSI-X vector, as indicated by the Interrupt Message Number field in the PCI Express Capabilities register.

The software handler for this interrupt can determine which device sent the PME Message by reading the PME Requester ID field in the Root Status register in a Root Port. It dismisses the interrupt by writing a 1b to the PME Status bit in the Root Status register. Refer to Section 7.5.3.14 for more details.

Root Complex Event Collectors provide support for the above described functionality for Root Complex Integrated Endpoints (RCiEPs).

6.1.7 Legacy PME Software Model

Legacy software, however, will not understand this mechanism for signaling PME. In the presence of legacy system software, the system power management logic in the Root Complex receives the PME Message and informs system software through an implementation specific mechanism. The Root Complex may utilize the Requester ID in the PM_PME to inform system software which device caused the power management event.

Since it is delivered by a Message, PME has edge-triggered semantics in PCI Express, which differs from the level-triggered PME mechanism used for conventional PCI. It is the responsibility of the Root Complex to abstract this difference from system software to maintain compatibility with conventional PCI systems.

6.1.8 Operating System Power Management Notification

In order to maintain compatibility with non-PCI Express-aware system software, system power management logic must be configured by firmware to use the legacy mechanism of signaling PME by default. PCI Express-aware system software must notify the firmware prior to enabling native, interrupt-based PME signaling. In response to this notification, system firmware must, if needed, reconfigure the Root Complex to disable legacy mechanisms of signaling PME. The details of this firmware notification are beyond the scope of this specification, but since it will be executed at system run-time, the response to this notification must not interfere with system software. Therefore, following control handoff to the operating system, firmware must not write to available system memory or any PCI Express resources (e.g., Configuration Space structures) owned by the operating system.

6.1.9 PME Routing Between PCI Express and PCI Hierarchies

PME-capable conventional PCI and PCI-X devices assert the PME# pin to signal a power management event. The PME# signal from PCI or PCI-X devices may either be converted to a PCI Express in-band PME Message by a PCI Express-PCI Bridge or routed directly to the Root Complex.

If the PME# signal from a PCI or PCI-X device is routed directly to the Root Complex, it signals system software using the same mechanism used in present PCI systems. A Root Complex may optionally provide support for signaling PME from PCI or PCI-X devices to system software via an interrupt. In this scenario, it is recommended for the Root Complex to detect the Bus, Device and Function Number of the PCI or PCI-X device that asserted PME#, and use this information to fill in the PME Requester ID field in the Root Port that originated the hierarchy containing the PCI or PCI-X device. If this is not possible, the Root Complex may optionally write the Requester ID of the Root Port to this field.

Since RCiEPs are not contained in any of the hierarchy domains originated by Root Ports, RCiEPs not associated with a Root Complex Event Collector signal system software of a PME using the same mechanism used in present PCI systems. A Root Complex Event Collector, if implemented, enables the PCI Express Native PME model for associated RCiEPs.

6.2 Error Signaling and Logging

In this document, errors which must be checked and errors which may optionally be checked are identified. Each such error is associated either with the Port or with a specific device (or Function in a Multi-Function Device), and this association is given along with the description of the error. This section will discuss how errors are classified and reported.

6.2.1 Scope

This section explains the error signaling and logging requirements for PCI Express components. This includes errors which occur on the PCI Express interface itself, those errors which occur on behalf of transactions initiated on PCI Express, and errors which occur within a component and are related to the PCI Express interface. This section does not focus on errors which occur within the component that are unrelated to a PCI Express interface. This type of error signaling is better handled through proprietary methods employing device-specific interrupts.

PCI Express defines two error reporting paradigms: the baseline capability and the Advanced Error Reporting Capability. The baseline error reporting capabilities are required of all PCI Express devices and define the minimum error reporting requirements. The Advanced Error Reporting Capability is defined for more robust error reporting and is implemented with a specific PCI Express Capability structure (refer to Chapter 7 for a definition of this optional capability). This section explicitly calls out all error handling differences between the baseline and the Advanced Error Reporting Capability.

All PCI Express devices support existing, non-PCI Express-aware, software for error handling by mapping PCI Express errors to existing PCI reporting mechanisms, in addition to the PCI Express-specific mechanisms.

6.2.2 Error Classification

PCI Express errors can be classified as two types: Uncorrectable errors and Correctable errors. This classification separates those errors resulting in functional failure from those errors resulting in degraded performance. Uncorrectable errors can further be classified as Fatal or Non-Fatal (see Figure 6-1).

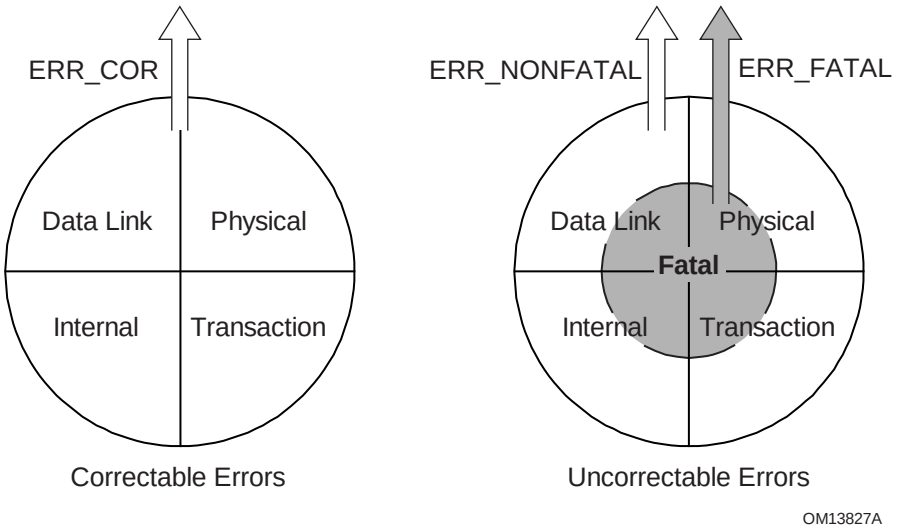


Figure 6-1: Error Classification

Classification of error severity as Fatal, Uncorrectable, and Correctable provides the platform with mechanisms for mapping the error to a suitable handling mechanism. For example, the platform might choose to respond to correctable errors with low priority, performance monitoring software.

Such software could count the frequency of correctable errors and provide Link integrity information. On the other hand, a platform designer might choose to map Fatal errors to a system-wide reset. It is the decision of the platform designer to map these PCI Express severity levels onto platform level severities.

6.2.2.1 *Correctable Errors*

Correctable errors include those error conditions where hardware can recover without any loss of information. Hardware corrects these errors and software intervention is not required. For example, an LCRC error in a TLP that might be corrected by Data Link Level Retry is considered a correctable error. Measuring the frequency of Link-level correctable errors may be helpful for profiling the integrity of a Link.

Correctable errors also include transaction-level cases where one agent detects an error with a TLP, but another agent is responsible for taking any recovery action if needed, such as re-attempting the operation with a separate subsequent transaction. The detecting agent can be configured to report the error as being correctable since the recovery agent may be able to correct it. If recovery action is indeed needed, the recovery agent must report the error as uncorrectable if the recovery agent decides not to attempt recovery.

The triggering of Downstream Port Containment (DPC) is not handled as an error, but it can be signaled as if it were a correctable error, since software that takes advantage of DPC can sometimes recover from the uncorrectable error that triggered DPC. See Section 6.2.10.

6.2.2.2 *Uncorrectable Errors*

Uncorrectable errors are those error conditions that impact functionality of the interface. There is no mechanism defined in this specification to correct these errors. Reporting an uncorrectable error is analogous to asserting SERR# in PCI/PCI-X. For more robust error handling by the system, this specification further classifies uncorrectable errors as Fatal and Non-fatal.

6.2.2.2.1 Fatal Errors

Fatal errors are uncorrectable error conditions which render the particular Link and related hardware unreliable. For Fatal errors, a reset of the components on the Link may be required to return to reliable operation. Platform handling of Fatal errors, and any efforts to limit the effects of these errors, is platform implementation specific.

6.2.2.2.2 Non-Fatal Errors

Non-fatal errors are uncorrectable errors which cause a particular transaction to be unreliable but the Link is otherwise fully functional. Isolating Non-fatal from Fatal errors provides Requester/Receiver logic in a device or system management software the opportunity to recover from the error without resetting the components on the Link and disturbing other transactions in progress. Devices not associated with the transaction in error are not impacted by the error.

6.2.3 Error Signaling

There are three complementary mechanisms which allow the agent detecting an error to alert the system or another device that an error has occurred. The first mechanism is through a Completion Status, the second method is with in-band error Messages, and the third is with Error Forwarding (also known as data poisoning).

Note that it is the responsibility of the agent detecting the error to signal the error appropriately.

Section 6.2.7 describes all the errors and how the hardware is required to respond when the error is detected.

6.2.3.1 *Completion Status*

The Completion Status field (when status is not Successful Completion) in the Completion header indicates that the associated Request failed (see Section 2.2.8.10). This is one method of error reporting which enables the Requester to associate an error with a specific Request. In other words, since Non-Posted Requests are not considered complete until after the Completion returns, the Completion Status field gives the Requester an opportunity to "fix" the problem at some higher level protocol (outside the scope of this specification). For example, if a Read is issued to prefetchable Memory Space and the Completion returns with an Unsupported Request Completion Status, the Requester would not be in violation of this specification if it chose to reissue the Read Request. Note that from a PCI Express point of view, the reissued Read Request is a distinct Request, and there is no relationship (on PCI Express) between the initial Request and the reissued Request.

6.2.3.2 *Error Messages*

Error Messages are sent to the Root Complex for reporting the detection of errors according to the severity of the error.

Error messages that originate from PCI Express or Legacy Endpoints are sent to corresponding Root Ports. Errors that originate from a Root Port itself are reported through the same Root Port.

If a Root Complex Event Collector is implemented, errors that originate from an RCiEP may optionally be sent to the corresponding Root Complex Event Collector. Errors that originate from an RCiEP are reported in a Root Complex Event Collector residing on the same Logical Bus as the RCiEP. The Root Complex Event Collector must explicitly declare supported RCiEPs as part of its capabilities; each RCiEP must be associated with no more than one Root Complex Event Collector.

When multiple errors of the same severity are detected, the corresponding error Messages with the same Requester ID may be merged for different errors of the same severity. At least one error Message must be sent for detected errors of each severity level. Note, however, that the detection of a given error in some cases will preclude the reporting of certain errors. Refer to Section 6.2.3.2.3. Also note special rules in Section 6.2.4 regarding non-Function-specific errors in Multi-Function Devices.

Table 6-1: Error Messages

Error Message	Description
ERR_COR	This Message is issued when the Function or Device detects a correctable error on the PCI Express interface. Refer to Section 6.2.2.1 for the definition of a correctable error.
ERR_NONFATAL	This Message is issued when the Function or Device detects a Non-fatal, uncorrectable error on the PCI Express interface. Refer to Section 6.2.2.2.2 for the definition of a Non-fatal, uncorrectable error.
ERR_FATAL	This Message is issued when the Function or Device detects a Fatal, uncorrectable error on the PCI Express interface. Refer to Section 6.2.2.2.1 for the definition of a Fatal, uncorrectable error.

For these Messages, the Root Complex identifies the initiator of the Message by the Requester ID of the Message header. The Root Complex translates these error Messages into platform level events.

IMPLEMENTATION NOTE

Use of ERR_COR, ERR_NONFATAL, and ERR_FATAL

In the *PCI Express Base Specification, Revisions* 1.0 and 1.0a, a given error was either correctable, non-fatal, or fatal. Assuming signaling was enabled, correctable errors were always signaled with ERR_COR, non-fatal errors were always signaled with ERR_NONFATAL, and fatal errors were always signaled with ERR_FATAL.

In subsequent specifications that support Role-Based Error Reporting, non-fatal errors are sometimes signaled with ERR_NONFATAL, sometimes signaled with ERR_COR, and sometimes not signaled at all, depending upon the role of the agent that detects the error and whether the agent implements AER (see Section 6.2.3.2.4). On some platforms, sending ERR_NONFATAL will preclude another agent from attempting recovery or determining the ultimate disposition of the error. For cases where the detecting agent is not the appropriate agent to determine the ultimate disposition of the error, a detecting agent with AER can signal the non-fatal error with ERR_COR, which serves as an advisory notification to software. For cases where the detecting agent is the appropriate one, the agent signals the non-fatal error with ERR_NONFATAL.

For a given uncorrectable error that's normally non-fatal, if software wishes to avoid continued hierarchy operation upon the detection of that error, software can configure detecting agents that implement AER to escalate the severity of that error to fatal. A detecting agent (if enabled) will always signal a fatal error with ERR_FATAL, regardless of the agent's role.

Software should recognize that a single transaction can be signaled by multiple agents using different types of error Messages. For example, a poisoned TLP might be signaled by intermediate Receivers with ERR_COR, while the ultimate destination Receiver might signal it with ERR_NONFATAL.

6.2.3.2.1 Uncorrectable Error Severity Programming (Advanced Error Reporting)

For device Functions implementing the Advanced Error Reporting Capability, the Uncorrectable Error Severity register allows each uncorrectable error to be programmed to Fatal or Non-Fatal. Uncorrectable errors are not recoverable using defined PCI Express mechanisms. However, some platforms or devices might consider a particular error fatal to a Link or device while another platform considers that error non-fatal. The default value of the Uncorrectable Error Severity register serves as a starting point for this specification but the register can be reprogrammed if the device driver or platform software requires more robust error handling.

Baseline error handling does not support severity programming.

6.2.3.2.2 Masking Individual Errors

Section 6.2.7 lists all the errors governed by this specification and describes when each of the above error Messages are issued. The transmission of these error Messages by class (correctable, non-fatal, fatal) is enabled using the Reporting Enable bits of the Device Control register (see Section 7.5.3.4) or the SERR# Enable bit in the PCI Command register (see Section 7.5.1.1.3).

For devices implementing the Advanced Error Reporting Capability the Uncorrectable Error Mask register and Correctable Error Mask register allows each error condition to be masked independently. If Messages for a particular class of error are not enabled by the combined settings in the Device Control register and the PCI Command register, then no Messages of that class will be sent regardless of the values for the corresponding mask register.

If an individual error is masked when it is detected, its error status bit is still affected, but no error reporting Message is sent to the Root Complex, and the error is not recorded in the Header Log, TLP Prefix Log, or First Error Pointer.

6.2.3.2.3 Error Pollution

Error pollution can occur if error conditions for a given transaction are not isolated to the most significant occurrence. For example, assume the Physical Layer detects a Receiver Error. This error is detected at the Physical Layer and an error is reported to the Root Complex. To avoid having this error propagate and cause subsequent errors at upper layers (for example, a TLP error at the Data Link Layer), making it more difficult to determine the root cause of the error, subsequent errors which occur for the same packet will not be reported by the Data Link or Transaction layers. Similarly, when the Data Link Layer detects an error, subsequent errors which occur for the same packet will not be reported by the Transaction Layer. This behavior applies only to errors that are associated with a particular packet – other errors are reported for each occurrence.

Corrected Internal Errors are errors whose effect has been masked or worked around by a component; refer to Section 6.2.9 for details. Therefore, Corrected Internal Errors do not contribute to error pollution and should be reported when detected.

For errors detected in the Transaction layer and Uncorrectable Internal Errors, it is permitted and recommended that no more than one error be reported for a single received TLP, and that the following precedence (from highest to lowest) be used:

- Uncorrectable Internal Error
- Receiver Overflow
- Flow Control Protocol Error
- Malformed TLP
- ECRC Check Failed
- AtomicOp Egress Blocked
- TLP Prefix Blocked
- ACS Violation
- MC Blocked TLP

- Unsupported Request (UR), Completer Abort (CA), or Unexpected Completion
- Poisoned TLP Received or Poisoned TLP Egress Blocked

The Completion Timeout error is not in the above precedence list, since it is not detected by processing a received TLP. Errors listed under the same bullet are mutually exclusive, so their relative order does not matter.

6.2.3.2.4 Advisory Non-Fatal Error Cases

In some cases the detector of a non-fatal error is not the most appropriate agent to determine whether the error is recoverable or not, or if it even needs any recovery action at all. For example, if software attempts to perform a configuration read from a non-existent device or Function, the resulting UR Status in the Completion will signal the error to software, and software does not need for the Completer in addition to signal the error by sending an ERR_NONFATAL Message. In fact, on some platforms, signaling the error with ERR_NONFATAL results in a System Error, which breaks normal software probing.

"Advisory Non-Fatal Error" cases are predominantly determined by the role of the detecting agent (Requester, Completer, or Receiver) and the specific error. In such cases, an agent with AER signals the non-fatal error (if enabled) by sending an ERR_COR Message as an advisory to software, instead of sending ERR_NONFATAL. An agent without AER sends no error Message for these cases, since software receiving ERR_COR would be unable to distinguish Advisory Non-Fatal Error cases from the correctable error cases used to assess Link integrity.

Following are the specific cases of Advisory Non-Fatal Errors. Note that multiple errors from the same or different error classes (correctable, non-fatal, fatal) may be present with a single TLP. For example, an unexpected Completion might also be poisoned. Refer to Section 6.2.3.2.3 for requirements and recommendations on reporting multiple errors. For the previous example, it is recommended that Unexpected Completion be reported, and that Poisoned TLP Received not be reported.

If software wishes for an agent with AER to handle what would normally be an Advisory Non-Fatal Error case as being more serious, software can escalate the severity of the uncorrectable error to fatal, in which case the agent (if enabled) will signal the error with ERR_FATAL.

This section covers Advisory Non-Fatal Error handling for errors managed by the PCI Express Extended Capability and AER. Section 6.2.10.3 covers the RP PIO error handling mechanism for Root Ports that support RP Extensions for DPC. RP PIO Advisory Non-Fatal Errors are similar in concept to AER Advisory Non-Fatal Errors, but apply to different error cases and are managed by different controls.

6.2.3.2.4.1 Completer Sending a Completion with UR/CA Status

A Completer generally sends a Completion with an Unsupported Request or Completer Abort (UR/CA) Status to signal an uncorrectable error for a Non-Posted Request.[77] If the severity of the

[77] If the Completer is returning data in a Completion, and the data is bad or suspect, the Completer is permitted to signal the error using the Error Forwarding (Data Poisoning) mechanism instead of handling it as a UR or CA.

UR/CA error[78] is non-fatal, the Completer must handle this case as an Advisory Non-Fatal Error.[79] A Completer with AER signals the non-fatal error (if enabled) by sending an ERR_COR Message. A Completer without AER sends no error Message for this case.

Even though there was an uncorrectable error for this specific transaction, the Completer must handle this case as an Advisory Non-Fatal Error, since the Requester upon receiving the Completion with UR/CA Status is responsible for reporting the error (if necessary) using a Requester-specific mechanism (see Section 6.2.3.2.5).

6.2.3.2.4.2 Intermediate Receiver

When a Receiver that's not serving as the ultimate PCI Express destination for a TLP detects[80] a non-fatal error with the TLP, this "intermediate" Receiver must handle this case as an Advisory Non-Fatal Error.[81] A Receiver with AER signals the error (if enabled) by sending an ERR_COR Message. A Receiver without AER sends no error Message for this case. An exception to the intermediate Receiver case for Root Complexes (RCs) is noted below.

An example where the intermediate Receiver case occurs is a Switch that detects poison or bad ECRC in a TLP that it is routing. Even though this was an uncorrectable (but non-fatal) error at this point in the TLP's route, the intermediate Receiver handles it as an Advisory Non-Fatal Error, so that the ultimate Receiver of the TLP (i.e., the Completer for a Request TLP, or the Requester for a Completion TLP) is not precluded from handling the error more appropriately according to its error settings. For example, a given Completer that detects poison in a Memory Write Request[82] might have the error masked (and thus go unsignaled), whereas a different Completer in the same hierarchy might signal that error with ERR_NONFATAL.

A Poisoned TLP Egress Blocked error is never handled as an intermediate Receiver case since it is not detected as a part of processing a received TLP.

If an RC detects a non-fatal error with a TLP it normally would forward peer-to-peer between Root Ports, but the RC does not support propagating the error related information (e.g., a TLP Digest, EP bit, or equivalent) with the forwarded transaction, the RC must signal the error (if enabled) with ERR_NONFATAL and also must not forward the transaction. An example is an RC needing to forward a poisoned TLP peer-to-peer between Root Ports, but the RC's internal fabric does not support poison indication.

---

[78] Certain other errors (e.g., ACS Violation) with a Non-Posted Request also result in the Completer sending a Completion with UR or CA Status. If the severity of the error (e.g., ACS Violation) is non-fatal, the Completer must also handle this case as an Advisory Non-Fatal Error. However, see Section 2.7.2.2 regarding certain Requests with Poisoned data that must be handled as uncorrectable errors.

[79] If the severity is fatal, the error is not an Advisory Non-Fatal Error, and must be signaled (if enabled) with ERR_FATAL.

[80] If the Receiver does not implement ECRC Checking or ECRC Checking is not enabled, the Receiver will not detect an ECRC Error.

[81] If the severity is fatal, the error is not an Advisory Non-Fatal Error, and must be signaled (if enabled) with ERR_FATAL.

[82] See Section 2.7.2.2 for special rules that apply for poisoned Memory Write Requests.

6.2.3.2.4.3 Ultimate PCI Express Receiver of a Poisoned TLP

When a poisoned TLP is received by its ultimate PCI Express destination, if the severity is non-fatal and the Receiver deals with the poisoned data in a manner that permits continued operation, the Receiver must handle this case[83] as an Advisory Non-Fatal Error.[84] A Receiver with AER signals the error (if enabled) by sending an ERR_COR Message. A Receiver without AER sends no error Message for this case. Refer to Section 2.7.2.2 for special rules that apply for poisoned Memory Write Requests.

An example is a Root Complex that receives a poisoned Memory Write TLP that targets host memory. If the Root Complex propagates the poisoned data along with its indication to host memory, it signals the error (if enabled) with an ERR_COR. If the Root Complex does not propagate the poison to host memory, it signals the error (if enabled) with ERR_NONFATAL.

Another example is a Requester that receives a poisoned Memory Read Completion TLP. If the Requester propagates the poisoned data internally or handles the error like it would for a Completion with UR/CA Status, it signals the error (if enabled) with an ERR_COR. If the Requester does not handle the poison in a manner that permits continued operation, it signals the error (if enabled) with ERR_NONFATAL.

6.2.3.2.4.4 Requester with Completion Timeout

This section applies to Requesters other than Root Ports performing programmed I/O (PIO). See Section 6.2.10.3 for related RP PIO functionality in Root Ports that support RP Extensions for DPC.

When the Requester of a Non-Posted Request times out while waiting for the associated Completion, the Requester is permitted to attempt to recover from the error by issuing a separate subsequent Request. The Requester is permitted to attempt recovery zero, one, or multiple (finite) times, but must signal the error (if enabled) with an uncorrectable error Message if no further recovery attempt will be made.

If the severity of the Completion Timeout is non-fatal, and the Requester elects to attempt recovery by issuing a new request, the Requester must first handle the current error case as an Advisory Non-Fatal Error.[85] A Requester with AER signals the error (if enabled) by sending an ERR_COR Message. A Requester without AER sends no error Message for this case.

Note that automatic recovery by the Requester from a Completion Timeout is generally possible only if the Non-Posted Request has no side-effects, but may also depend upon other considerations outside the scope of this specification.

---

[83] However, see Section 2.7.2.2 regarding certain Requests with Poisoned data that must be handled as uncorrectable errors.

[84] If the severity is fatal, the error is not an Advisory Non-Fatal Error, and must be signaled (if enabled) with ERR_FATAL.

[85] If the severity is fatal, the error is not an Advisory Non-Fatal Error, and must be signaled (if enabled) with ERR_FATAL. The Requester is strongly discouraged from attempting recovery since sending ERR_FATAL will often result in the entire hierarchy going down.

6.2.3.2.4.5 Receiver of an Unexpected Completion

When a Receiver receives an unexpected Completion and the severity of the Unexpected Completion error is non-fatal, the Receiver must handle this case as an Advisory Non-Fatal Error.[86] A Receiver with AER signals the error (if enabled) by sending an ERR_COR Message. A Receiver without AER sends no error Message for this case.

If the unexpected Completion was a result of misrouting, the Completion Timeout mechanism at the associated Requester will trigger eventually, and the Requester may elect to attempt recovery. Interference with Requester recovery can be avoided by having the Receiver of the unexpected Completion handle the error as an Advisory Non-Fatal Error.

6.2.3.2.5 Requester Receiving a Completion with UR/CA Status

When a Requester receives back a Completion with a UR/CA Status, generally the Completer has handled the error as an Advisory Non-Fatal Error, assuming the error severity was non-fatal at the Completer (see Section 6.2.3.2.4.1). The Requester must determine if any error recovery action is necessary, what type of recovery action to take, and whether or not to report the error.

If the Requester needs to report the error, the Requester must do so solely through a Requester-specific mechanism. For example, many devices have an associated device driver that can report errors to software. As another important example, the Root Complex on some platforms returns all 1's to software if a Configuration Read Completion has a UR/CA Status.

Section 6.2.10.3 covers RP PIO controls for Root Ports that support RP Extensions for DPC. Outside of the RP PIO mechanisms, Requesters are not permitted to report the error using PCI Express logging and error Message signaling.

6.2.3.3 *Error Forwarding (Data Poisoning)*

Error Forwarding, also known as data poisoning, is indicated by setting the EP bit in a TLP. Refer to Section 2.7.2. This is another method of error reporting in PCI Express that enables the Receiver of a TLP to associate an error with a specific Request or Completion. Unlike the Completion Status mechanism, Error Forwarding can be used with either Requests or Completions that contain data. In addition, "intermediate" Receivers along the TLP's route, not just the Receiver at the ultimate destination, are required to detect and report (if enabled) receiving the poisoned TLP. This can help software determine if a particular Switch along the path poisoned the TLP.

6.2.3.4 *Optional Error Checking*

This specification contains a number of optional error checks. Unless otherwise specified, behavior is undefined if an optional error check is not performed and the error occurs.

When an optional error check involves multiple rules, unless otherwise specified, each rule is independently optional. An implementation may check against all of the rules, none of them or any combination.

---

[86] If the severity is fatal, the error is not an Advisory Non-Fatal Error, and must be signaled (if enabled) with ERR_FATAL.

Unless otherwise specified, implementation specific criteria are used in determining whether an optional error check is performed.

6.2.4 Error Logging

Section 6.2.7 lists all the errors governed by this specification and for each error, the logging requirements are specified. Device Functions that do not support the Advanced Error Reporting Capability log only the Device Status register bits indicating that an error has been detected. Note that some errors are also reported using the reporting mechanisms in the PCI-compatible (Type 00h and 01h) configuration registers. Section 7.5.1 describes how these register bits are affected by the different types of error conditions described in this section.

For device Functions supporting the Advanced Error Reporting Capability, each of the errors in Table 6-3, Table 6-4, and Table 6-5 corresponds to a particular bit in the Uncorrectable Error Status register or Correctable Error Status register. These registers are used by software to determine more precisely which error and what severity occurred. For specific Transaction Layer errors and Uncorrectable Internal Errors, the associated TLP header is recorded.

In a Multi-Function Device, PCI Express errors that are not related to any specific Function within the device, are logged in the corresponding status and logging registers of all Functions in that device.

The following PCI Express errors are not Function-specific:

- All Physical Layer errors
- All Data Link Layer errors
- These Transaction Layer errors:
  - ECRC Check Failed
  - Unsupported Request, when caused by no Function claiming a TLP
  - Receiver Overflow
  - Flow Control Protocol Error
  - Malformed TLP
  - Unexpected Completion, when caused by no Function claiming a Completion
  - Unexpected Completion, when caused by a Completion that cannot be forwarded by a Switch, and the Ingress Port is a Switch Upstream Port associated with a Multi-Function Device
  - Some Transaction Layer errors (e.g., Poisoned TLP Received) may be Function-specific or not, depending upon whether the associated TLP targets a single Function or all Functions in that device.
- Some Internal Errors
  - The determination of whether an Internal Error is Function-specific or not is implementation specific.

On the detection of one of these errors, a Multi-Function Device should generate at most one error reporting Message of a given severity, where the Message must report the Requester ID of a

Function of the device that is enabled to report that specific type of error. If no Function is enabled to send a reporting Message, the device does not send a reporting Message. If all reporting-enabled Functions have the same severity level set for the error, only one error Message is sent. If all reporting-enabled Functions do not have the same severity level set for the error, one error Message for each severity level is sent. Software is responsible for scanning all Functions in a Multi-Function Device when it detects one of those errors.

6.2.4.1 *Root Complex Considerations (Advanced Error Reporting)*

6.2.4.1.1 Error Source Identification

In addition to the above logging, a Root Port or Root Complex Event Collector that supports the Advanced Error Reporting Capability is required to implement the Error Source Identification register, which records the Requester ID of the first ERR_NONFATAL/ERR_FATAL (uncorrectable errors) and ERR_COR (correctable errors) Messages received by the Root Port or Root Complex Event Collector. System software written to support Advanced Error Reporting can use the Root Error Status register to determine which fields hold valid information.

If a Root Complex Event Collector is implemented, errors from an RCiEP may optionally be reported in a Root Complex Event Collector residing on the same Logical Bus as the RCiEP. The Root Complex Event Collector must explicitly declare supported RCiEPs as part of its capabilities. Each RCiEP must be associated with no more than one Root Complex Event Collector.

For both Root Ports and Root Complex Event Collectors, in order for a received error Message or an internally generated error Message to be recorded in the Root Error Status register and the Error Source Identification register, the error Message must be "transmitted". Refer to Section 6.2.8.1 for information on how received Messages are forwarded and transmitted. Internally generated error Messages are enabled for transmission with the SERR# Enable bit in the Command register (ERR_NONFATAL and ERR_FATAL) or the Reporting Enable bits in the Device Control register (ERR_COR, ERR_NONFATAL, and ERR_FATAL).

6.2.4.1.2 Interrupt Generation

The Root Error Command register allows further control of Root Complex response to Correctable, Non-Fatal, and Fatal error Messages than the basic Root Complex capability to generate system errors in response to error Messages. Bit fields enable or disable generation of interrupts for the three types of error Messages. System error generation in response to error Messages may be disabled via the PCI Express Capability structure.

If a Root Port or Root Complex Event Collector is enabled for level-triggered interrupt signaling using the INTx messages, the virtual INTx wire must be asserted whenever and as long as all of the following conditions are satisfied:

- The Interrupt Disable bit in the Command register is set to 0b.
- At least one Error Reporting Enable bit in the Root Error Command register and its associated error Messages Received bit in the Root Error Status register are both set to 1b.

Note that all other interrupt sources within the same Function will assert the same virtual INTx wire when requesting service.

If a Root Port or Root Complex Event Collector is enabled for edge-triggered interrupt signaling using MSI or MSI-X, an interrupt message must be sent every time the logical AND of the following conditions transitions from FALSE to TRUE:

- The associated vector is unmasked (not applicable if MSI does not support PVM).
- At least one Error Reporting Enable bit in the Root Error Command register and its associated error Messages Received bit in the Root Error Status register are both set to 1b.

Note that Advanced Error Reporting MSI/MSI-X interrupts always use the vector indicated by the Advanced Error Interrupt Message Number field in the Root Error Status register.

6.2.4.2 *Multiple Error Handling (Advanced Error Reporting Capability)*

For the Advanced Error Reporting Capability, the Uncorrectable Error Status register and Correctable Error Status register accumulate the collection of errors which correspond to that particular PCI Express interface. The bits remain set until explicitly cleared by software or reset. Since multiple bits might be set in the Uncorrectable Error Status register, the First Error Pointer (when valid) points to the oldest uncorrectable error that is recorded. The First Error Pointer is valid when the corresponding bit of the Uncorrectable Error Status register is set. The First Error Pointer is invalid when the corresponding bit of the Uncorrectable Error Status register is not set, or is an undefined bit.

The Advanced Error Reporting Capability provides the ability to record headers[87] for errors that require header logging. An implementation may support the recording of multiple headers, but at a minimum must support the ability of recording at least one. The ability to record multiple headers is indicated by the state of the Multiple Header Recording Capable bit and enabled by the Multiple Header Recording Enable bit of the Advanced Error Capabilities and Control register. When multiple header recording is supported and enabled, errors are recorded in the order in which they are detected.

If no header recording resources are available when an unmasked uncorrectable error is detected, its error status bit is set, but the error is not recorded. If an uncorrectable error is masked when it is detected, its error status bit is set, but the error is not recorded.

When software is ready to dismiss a recorded error indicated by the First Error Pointer, software writes a 1b to the indicated error status bit to clear it, which causes hardware to free up the associated recording resources. If another instance of that error is still recorded, hardware is permitted but not required to leave that error status bit set. If any error instance is still recorded, hardware must immediately update the Header Log, TLP Prefix Log, TLP Prefix Log Present bit, First Error Pointer, and Uncorrectable Error Status register to reflect the next recorded error. If no

[87] If a Function supports TLP Prefixes, then its AER Capability also records any accompanying TLP Prefix along with each recorded header. References to header recording also imply TLP Prefix recording.

other error is recorded, it is recommended that hardware update the First Error Pointer to indicate a status bit that it will never set, e.g., a Reserved status bit. See the Implementation Note below.

If multiple header recording is supported and enabled, and the First Error Pointer is valid, it is recommended that software not write a 1b to any status bit other than the one indicated by the First Error Pointer[88]. If software writes a 1b to such non-indicated bits, hardware is permitted to clear any associated recorded errors, but is not required to do so.

If software observes that the First Error Pointer is invalid, and software wishes to clear any unmasked status bits that were set because of earlier header recording resource overflow, software should be aware of the following race condition. If any new instances of those errors happen to be recorded before software clears those status bits, one or more of the newly recorded errors might be lost.

If multiple header recording is supported and enabled, software must use special care when clearing the Multiple Header Recording Enable bit. Hardware behavior is undefined if software clears that bit while the First Error Pointer is valid. Before clearing the Multiple Header Recording Enable bit, it is recommended that software temporarily mask all uncorrectable errors, and then repetitively dismiss each error indicated by the First Error Pointer.

Since an implementation only has the ability to record a finite number of headers, it is important that software services the First Error Pointer, Header Log, and TLP Prefix Log registers in a timely manner, to limit the risk of missing this information for subsequent errors. A Header Log Overflow occurs when an error that requires header logging is detected and either the number of recorded headers supported by an implementation has been reached, or the Multiple Header Recording Enable bit is not Set and the First Error Pointer is valid.

Implementations may optionally check for this condition and report a Header Log Overflow error. This is a reported error associated with the detecting Function.

The setting of Multiple Header Recording Capable and the checking for Header Log Overflow are independently optional.

[88] Status bits for masked errors are an exception. Software can safely clear them if software is certain that they have no recorded headers, as would be the case if they have remained masked since the First Error Pointer was last invalid.

IMPLEMENTATION NOTE

First Error Pointer Register Being Valid

The First Error Pointer (FEP) field is defined to be valid when the corresponding bit of the Uncorrectable Error Status register is set. To avoid ambiguity with certain cases, the following is recommended:

- After an uncorrectable error has been recorded, when the associated bit in the Uncorrectable Error Status register is cleared by software writing a 1b to it, hardware should update the FEP to point to a status bit that it will never set, e.g., a Reserved status bit. (This assumes that the Function does not already have another recorded error to report, as could be the case if it supports multiple header recording.)
- The default value for the FEP should point to a status bit that hardware will never set, e.g., a Reserved status bit.

Here is an example case of ambiguity with Unsupported Request (UR) if the above recommendations are not followed:

- UR and Advisory Non-Fatal Error are unmasked while system firmware does its Configuration Space probing.
- The Function encounters a UR due to normal probing, logs it, and sets the FEP to point to UR.
- System firmware clears the UR Status bit, and hardware leaves the FEP pointing to UR.
- After the operating oystem has booted, it masks UR.
- Normal probing sets the UR Status bit, but the error is not recorded since UR is masked.

At this point, there's the ambiguity of the FEP pointing to a status bit that is set (thus being valid), when in fact, there is no recorded error that needs to be processed by software.

If hardware relies on this definition of the FEP being valid to determine when it's possible to record a new error, the Function can fail to record new unmasked errors, falsely determining that it has no available recording resources. Hardware implementations that rely on other internal state to determine when it's possible to record a new error might not have this problem; however, hardware implementations should still follow the above recommendations to avoid presenting this ambiguity to software.

6.2.4.3 *Advisory Non-Fatal Error Logging*

Section 6.2.3.2.4 describes Advisory Non-Fatal Error cases, under which an agent with AER detecting an uncorrectable error of non-fatal severity signals the error (if enabled) using ERR_COR instead of ERR_NONFATAL. For the same cases, an agent without AER sends no error Message. The remaining discussion in this section is in the context of agents that do implement AER.

For Advisory Non-Fatal Error cases, since an uncorrectable error is signaled using the correctable error Message, control/status/mask bits involving both uncorrectable and correctable errors apply. Figure 6-2 shows a flowchart of the sequence. Following are some of the unique aspects for logging Advisory Non-Fatal Errors.

First, the uncorrectable error needs to be of severity non-fatal, as determined by the associated bit in the Uncorrectable Error Severity register. If the severity is fatal, the error does not qualify as an Advisory Non-Fatal Error, and will be signaled (if enabled) with ERR_FATAL.

Next, the specific error case needs to be one of the Advisory Non-Fatal Error cases documented in Section 6.2.3.2.4. If not, the error does not qualify as an Advisory Non-Fatal Error, and will be signaled (if enabled) with an uncorrectable error Message.

Next, the Advisory Non-Fatal Error Status bit is set in the Correctable Error Status register to indicate the occurrence of the advisory error, and the Advisory Non-Fatal Error Mask bit in the Correctable Error Mask register is checked, and, if set, no further processing is done.

If the Advisory Non-Fatal Error Mask bit is clear, logging proceeds by setting the "corresponding" bit in the Uncorrectable Error Status register, based upon the specific uncorrectable error that's being reported as an advisory error. If the "corresponding" uncorrectable error bit in the Uncorrectable Error Mask register is clear and the error is one that requires header logging, then the prefix and header are recorded, subject to the availability of resources. See Section 6.2.4.2.

Finally, an ERR_COR Message is sent if the Correctable Error Reporting Enable bit is set in the Device Control register.

6.2.4.4 *TLP Prefix Logging*

For any device Function that supports both TLP Prefixes and Advanced Error Reporting the TLP Prefixes associated with the TLP in error are recorded in the TLP Prefix Log register according to the same rules as the Header Log register (such that both the TLP Prefix Log and Header Log registers always correspond to the error indicated in the First Error Pointer, when the First Error Pointer is valid).

The TLP Prefix Log Present bit (see Section 7.8.4.7) indicates that the TLP Prefix Log register (see Section 7.8.4.12) contains information.

Only End-End TLP Prefixes are logged by AER. Logging of Local TLP Prefixes may occur elsewhere using prefix specific mechanisms.[89]

End-End TLP Prefixes are logged in the TLP Prefix Log register. The underlying TLP Header is logged in the Header Log register subject to two exceptions:

---

[89] For example, errors involving MRI-IOV TLP Prefixes are logged in MR-IOV structures and are not logged in the AER Capability.

- If the Extended Fmt Field Supported bit is Set (see Section 7.5.3.15), a Function that does not support TLP Prefixes and receives a TLP containing a TLP Prefix will signal Malformed TLP and the Header Log register will contain the first four DWs of the TLP (TLP Prefixes followed by as much of the TLP Header as will fit).
- A Function that receives a TLP containing more End-End TLP Prefixes than are indicated by the Function's Max End-End TLP Prefixes field must handle the TLP as an error (see Section 2.2.10.2 for specifics) and store the first overflow End-End TLP Prefix in the 1st DW of the Header Log register with the remainder of the Header Log register being undefined.

6.2.5 Sequence of Device Error Signaling and Logging Operations

Figure 6-2 shows the sequence of operations related to signaling and logging of errors detected by a device.

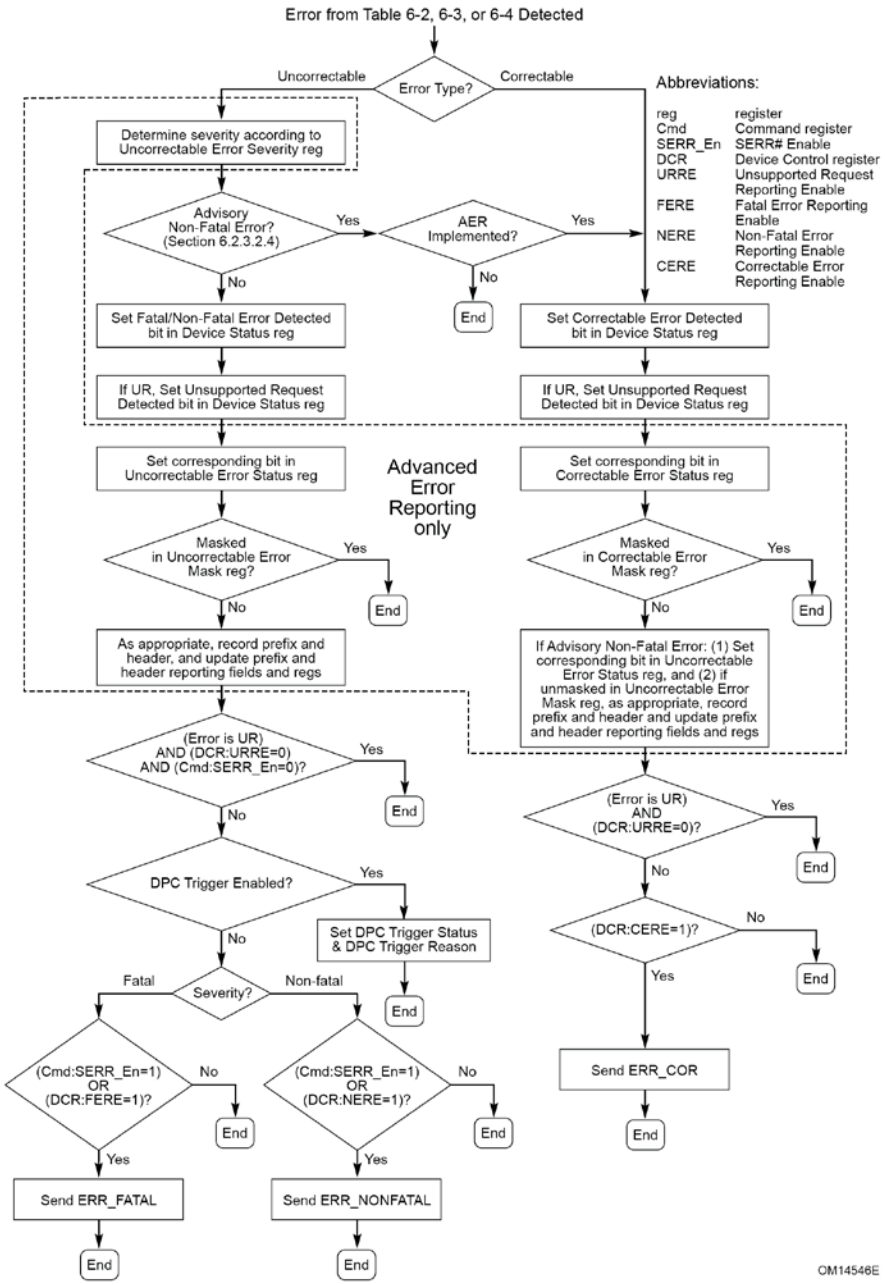


Figure 6-2: Flowchart Showing Sequence of Device Error Signaling and Logging Operations

6.2.6 Error Message Controls

Error Messages have a complex set of associated control and status bits. Figure 6-3 provides a conceptual summary in the form of a pseudo logic diagram for how error Messages are generated, logged, forwarded, and ultimately notified to the system. Not all logged status bits are shown. The logic gates shown in this diagram are intended for conveying general concepts, and not for direct implementation.

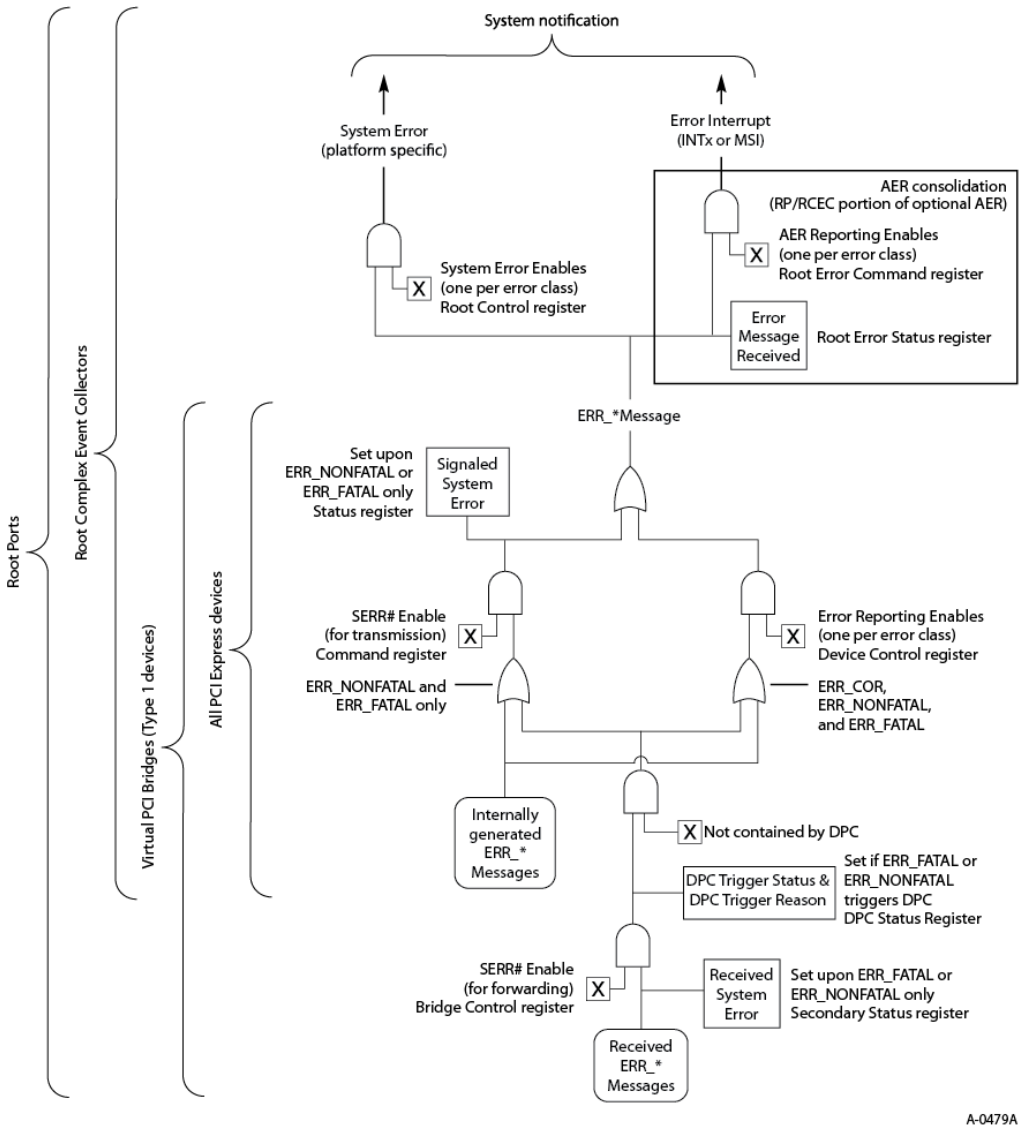


Figure 6-3: Pseudo Logic Diagram for Error Message Controls

6.2.7 Error Listing and Rules

Table 6-2 through Table 6-4 list all of the PCI Express errors that are defined by this specification. Each error is listed with a short-hand name, how the error is detected in hardware, the default

severity of the error, and the expected action taken by the agent which detects the error. These actions form the rules for PCI Express error reporting and logging.

The Default Severity column specifies the default severity for the error without any software reprogramming. For device Functions supporting the Advanced Error Reporting Capability, the uncorrectable errors are programmable to Fatal or Non-fatal with the Error Severity register. Device Functions without Advanced Error Reporting Capability use the default associations and are not reprogrammable.

The detecting agent action for Downstream Ports that implement Downstream Port Containment (DPC) and have it enabled will be different if the error triggers DPC. DPC behavior is not described in the following tables. See Section 6.2.10 for the description of DPC behavior.

Table 6-2: General PCI Express Error List

Error Name	Error Type (Default Severity)	Detecting Agent Action[90]	References
Corrected Internal Error	Correctable (masked by default)	Component: Send ERR_COR to Root Complex.	Section 6.2.9
Uncorrectable Internal Error	Uncorrectable (Fatal and masked by default)	*Component:* Send ERR_FATAL to Root Complex. Optionally, log the prefix/header of the first TLP associated with the error.	Section 6.2.9
Header Log Overflow	Correctable (masked by default)	Component: Send ERR_COR to Root Complex.	Section 6.2.4.2

Table 6-3: Physical Layer Error List

Error Name	Error Type (Default Severity)	Detecting Agent Action[91]	References
Receiver Error	Correctable	Receiver: Send ERR_COR to Root Complex.	Section 4.2.1.1.3 Section 4.2.1.2 Section 4.2.4.7 Section 4.2.6

[90] For these tables, detecting agent action is given as if all enable bits are set to “enable” and, for Advanced Error Handling, mask bits are disabled and severity bits are set to their default values. Actions must be modified according to the actual settings of these bits.

[91] For these tables, detecting agent action is given as if all enable bits are set to “enable” and, for Advanced Error Handling, mask bits are disabled and severity bits are set to their default values. Actions must be modified according to the actual settings of these bits.

Table 6-4: Data Link Layer Error List

Error Name	Error Type (Default Severity)	Detecting Agent Action[91]	References
Bad TLP	Correctable	Receiver: Send ERR_COR to Root Complex.	Section 3.6.3.1
Bad DLLP		Receiver: Send ERR_COR to Root Complex.	Section 3.6.2.2
Replay Timer Timeout		Transmitter: Send ERR_COR to Root Complex.	Section 3.6.2.1
REPLAY_NUM Rollover		Transmitter: Send ERR_COR to Root Complex.	Section 3.6.2.1
Data Link Protocol Error	Uncorrectable (Fatal)	If checking, send ERR_FATAL to Root Complex.	Section 3.6.2.2
Surprise Down		If checking, send ERR_FATAL to Root Complex.	Section 3.2.1

Table 6-5: Transaction Layer Error List

Error Name	Error Type (Default Severity)	Detecting Agent Action[91]	References
Poisoned TLP Received	Uncorrectable (Non-Fatal)	Receiver: Send ERR_NONFATAL to Root Complex or ERR_COR for the Advisory Non-Fatal Error cases described in Sections 6.2.3.2.4.2 and 6.2.3.2.4.3. Log the prefix/header of the Poisoned TLP.[92]	Section 2.7.2.2
Poisoned TLP Egress Blocked		Downstream Port Transmitter: Send ERR_NONFATAL to Root Complex. Log the prefix/header of the poisoned TLP.	Section 2.7.2.2
ECRC Check Failed		Receiver (if ECRC checking is supported): Send ERR_NONFATAL to Root Complex or ERR_COR for the Advisory Non-Fatal Error case described in Sections 6.2.3.2.4.1 and 6.2.3.2.4.2. Log the prefix/header of the TLP that encountered the ECRC error.	Section 2.7.1

[92] Advanced Error Handling only.

Error Name	Error Type (Default Severity)	Detecting Agent Action[91]	References
Unsupported Request (UR)	Uncorrectable (Non-Fatal)	Request Receiver: Send ERR_NONFATAL to Root Complex or ERR_COR for the Advisory Non-Fatal Error case described in Section 6.2.3.2.4.1. Log the prefix/header of the TLP that caused the error.	Table F-1, Section 2.3.1, Section 2.3.2, Section 2.7.2.2, Section 2.9.1, Section 5.3.1, Section 6.2.3.1, Section 6.2.6, Section 6.2.8.1, Section 6.5.7, Section 7.3.1, Section 7.3.3, Section 7.5.1.1.3, Section 7.5.1.1.4
Completion Timeout	Uncorrectable (Non-Fatal)	Requester: Send ERR_NONFATAL to Root Complex or ERR_COR for the Advisory Non-Fatal Error case described in Section 6.2.3.2.4.4. If the Completion Timeout Prefix/Header Log Capable bit is Set in the Advanced Error Capabilities and Control register, log the prefix/header of the Request TLP that encountered the error.	Section 2.8
Completer Abort		Completer: Send ERR_NONFATAL to Root Complex or ERR_COR for the Advisory Non-Fatal Error case described in Section 6.2.3.2.4.1. Log the prefix/header of the Request that encountered the error.	Section 2.3.1
Unexpected Completion		Receiver: Send ERR_COR to Root Complex. This is an Advisory Non-Fatal Error case described in Section 6.2.3.2.4.5. Log the prefix/header of the Completion that encountered the error.	Section 2.3.2
ACS Violation		Receiver (if checking): Send ERR_NONFATAL to Root Complex or ERR_COR for the Advisory Non-Fatal Error case described in Section 6.2.3.2.4.1. Log the prefix/header of the Request TLP that encountered the error.	
MC Blocked TLP		Receiver (if checking): Send ERR_NONFATAL to Root Complex. Log the prefix/header of the Request TLP that encountered the error.	Section 6.14.4

Error Name	Error Type (Default Severity)	Detecting Agent Action[91]	References
AtomicOp Egress Blocked	Uncorrectable (Non-Fatal)	Egress Port: Send ERR_COR to Root Complex. This is an Advisory Non-Fatal Error case described in Section 6.2.3.2.4.1. Log the prefix/header of the AtomicOp Request that encountered the error.	Section 6.15.2
TLP Prefix Blocked		Egress Port: Send ERR_NONFATAL to Root Complex or ERR_COR for the Advisory Non-Fatal Error case described in Section 6.2.3.2.4.1. Log the prefix/header of the TLP that encountered the error.	Section 2.2.10.2
Receiver Overflow	Uncorrectable (Fatal)	Receiver (if checking): Send ERR_FATAL to Root Complex.	Section 2.6.1.2
Flow Control Protocol Error		Receiver (if checking): Send ERR_FATAL to Root Complex.	Section 2.6.1
Malformed TLP		Receiver: Send ERR_FATAL to Root Complex. Log the prefix/header of the TLP that encountered the error.	Section 2.2.2, Section 2.2.3, Section 2.2.5, Section 2.2.7, Section 2.2.8.1, Section 2.2.8.2, Section 2.2.8.3, Section 2.2.8.4, Section 2.2.8.5, Section 2.2.8.10, Section 2.2.10, Section 2.2.10.1, Section 2.2.10.2, Section 2.3, Section 2.3.1, Section 2.3.1.1, Section 2.3.2, Section 2.5, Section 2.5.3, Section 2.6.1, Section 2.6.1.2, Section 6.2.4.4, Section 6.3.2

For all errors listed above, the appropriate status bit(s) must be set upon detection of the error. For Unsupported Request (UR), additional detection and reporting enable bits apply (see Section 6.2.5).

IMPLEMENTATION NOTE

Device UR Reporting Compatibility with Legacy and 1.0a Software

With 1.0a device Functions that do not implement Role-Based Error Reporting,[93] the Unsupported Request Reporting Enable bit in the Device Control register, when clear, prevents the Function from sending any error Message to signal a UR error. With Role-Based Error Reporting Functions, if the SERR# Enable bit in the Command register is set, the Function is implicitly enabled[94] to send ERR_NONFATAL or ERR_FATAL messages to signal UR errors, even if the Unsupported Request Reporting Enable bit is clear. This raises a backward compatibility concern with software (or firmware) written for 1.0a devices.

With software/firmware that sets the SERR# Enable bit but leaves the Unsupported Request Reporting Enable and Correctable Error Reporting Enable bits clear, a Role-Based Error Reporting Function that encounters a UR error will send no error Message if the Request was non-posted, and will signal the error with ERR_NONFATAL if the Request was posted. The behavior with non-posted Requests supports PC-compatible Configuration Space probing, while the behavior with posted Requests restores error reporting compatibility with PCI and PCI-X, avoiding the potential in this area for silent data corruption. Thus, Role-Based Error Reporting devices are backward compatible with envisioned legacy and 1.0a software and firmware.

6.2.7.1 *Conventional PCI Mapping*

In order to support conventional PCI driver and software compatibility, PCI Express error conditions, where appropriate, must be mapped onto the PCI Status register bits for error reporting.

In other words, when certain PCI Express errors are detected, the appropriate PCI Status register bit is set alerting the error to legacy PCI software. While the PCI Express error results in setting the PCI Status register, clearing the PCI Status register will not result in clearing bits in the Uncorrectable Error Status register and Correctable Error Status register. Similarly, clearing bits in the Uncorrectable Error Status register and Correctable Error Status register will not result in clearing the PCI Status register.

The PCI command register has bits which control PCI error reporting. However, the PCI Command register does not affect the setting of the PCI Express error register bits.

6.2.8 Virtual PCI Bridge Error Handling

Virtual PCI Bridge configuration headers are associated with each PCI Express Port in a Root Complex or a Switch. For these cases, PCI Express error concepts require appropriate mapping to the PCI error reporting structures.

---

[93] As indicated by the Role-Based Error Reporting bit in the Device Capabilities register. See Section 7.8.3.

[94] Assuming the Unsupported Request Error Mask bit is not set in the Uncorrectable Error Mask register if the device implements AER.

6.2.8.1 *Error Message Forwarding and PCI Mapping for Bridge - Rules*

In general, a TLP is either passed from one side of the Virtual PCI Bridge to the other, or is handled at the ingress side of the Bridge according to the same rules which apply to the ultimate recipient of a TLP. The following rules cover PCI Express specific error related cases. Refer to Section 6.2.6 for a conceptual summary of Error Message Controls.

- If a Request does not address a space mapped to either the Bridge's internal space, or to the egress side of the Bridge, the Request is terminated at the ingress side as an Unsupported Request
- Poisoned TLPs are forwarded according to the same rules as non-Poisoned TLPs
  - When forwarding a Poisoned Request Downstream:
    - Set the Detected Parity Error bit in the Status register
    - Set the Master Data Parity Error bit in the Secondary Status register if the Parity Error Response Enable bit in the Bridge Control register is set
  - When forwarding a Poisoned Completion Downstream:
    - Set the Detected Parity Error bit in the Status register
    - Set the Master Data Parity Error bit in the Status register if the Parity Error Response bit in the Command register is set
  - When forwarding a Poisoned Request Upstream:
    - Set the Detected Parity Error bit in the Secondary Status register
    - Set the Master Data Parity Error bit in the Status register if the Parity Error Response bit in the Command register is set
  - When forwarding a Poisoned Completion Upstream:
    - Set the Detected Parity Error bit in the Secondary Status register
    - Set the Master Data Parity Error bit in the Secondary Status register if the Parity Error Response Enable bit in the Bridge Control register is set
- ERR_COR, ERR_NONFATAL, and ERR_FATAL are forwarded from the secondary interface to the primary interface, if the SERR# Enable bit in the Bridge Control register is set. A Bridge forwarding an error Message must not set the corresponding Error Detected bit in the Device Status register. Transmission of forwarded error Messages by the primary interface is controlled by multiple bits, as shown in Figure 6-3.
- For a Root Port, error Messages forwarded from the secondary interface to the primary interface must be enabled for "transmission" by the primary interface in order to cause a System Error via the Root Control register or (when the Advanced Error Reporting Capability is present) reporting via the Root Error Command register and logging in the Root Error Status register and Error Source Identification register.
- For a Root Complex Event Collector (technically not a Bridge), error Messages "received" from associated RCiEPs must be enabled for "transmission" in order to cause a System Error via the

Root Control register or (when the Advanced Error Reporting Capability is present) reporting via the Root Error Command register and logging in the Root Error Status register and Error Source Identification register.

6.2.9 Internal Errors

An Internal Error is an error associated with a PCI Express interface that occurs within a component and which may not be attributable to a packet or event on the PCI Express interface itself or on behalf of transactions initiated on PCI Express. The determination of what is considered an Internal Error is implementation specific and is outside the scope of this specification.

Internal Errors may be classified as Corrected Internal Errors or Uncorrectable Internal Errors. A Corrected Internal Error is an error that occurs within a component that has been masked or worked around by hardware without any loss of information or improper operation. An example of a possible Corrected Internal Error is an internal packet buffer memory error corrected by an Error Correcting Code (ECC). An Uncorrectable Internal Error is an error that occurs within a component that results in improper operation of the component. An example of a possible Uncorrectable Internal Error is a memory error that cannot be corrected by an ECC. The only method of recovering from an Uncorrectable Internal Error is reset or hardware replacement.

Reporting of Corrected Internal Errors and Uncorrectable Internal Errors is independently optional. If either is reported, then AER must be implemented.

Header logging is optional for Uncorrectable Internal Errors. When a header is logged, the header is that of the first TLP that was lost or corrupted by the Uncorrectable Internal Error. When header logging is not implemented or a header is not available, a header of all ones is recorded.

Internal Errors that can be associated with a specific PCI Express interface are reported by the Function(s) associated with that Port. Internal Errors detected within Switches that cannot be associated with a specific PCI Express interface are reported by the Upstream Port. Reporting of Internal Errors that cannot be associated with a specific PCI Express interface in all other multi-Port components (e.g., Root Complexes) is outside the scope of this specification.

6.2.10 Downstream Port Containment (DPC)

Downstream Port Containment (DPC) is an optional normative feature of a Downstream Port. DPC halts PCI Express traffic below a Downstream Port after an unmasked uncorrectable error is detected at or below the Port, avoiding the potential spread of any data corruption, and permitting error recovery if supported by software. A Downstream Port indicates support for DPC by implementing a DPC Extended Capability structure, which contains all DPC control and status bits. See Section 7.9.15.

DPC is disabled by default, and cannot be triggered unless enabled by software using the DPC Trigger Enable field. When the DPC Trigger Enable field is set to 01b, DPC is enabled and is triggered when the Downstream Port detects an unmasked uncorrectable error or when the Downstream Port receives an ERR_FATAL Message. When the DPC Trigger Enable field is set to 10b, DPC is enabled and is triggered when the Downstream Port detects an unmasked uncorrectable error or when the Downstream Port receives an ERR_NONFATAL or ERR_FATAL Message. In addition to uncorrectable errors of the type managed by the PCI Express Extended Capability and Advanced Error Reporting (AER), RP PIO errors can be handled as

uncorrectable errors. See Section 6.2.10.3. There is also a mechanism described in Section 6.2.10.4 for software or firmware to trigger DPC.

When DPC is triggered due to receipt of an uncorrectable error Message, the Requester ID from the Message is recorded in the DPC Error Source ID register and that Message is discarded and not forwarded Upstream. When DPC is triggered by an unmasked uncorrectable error, that error will not be signaled with an uncorrectable error Message, even if otherwise enabled. However, when DPC is triggered, DPC can signal an interrupt or send an ERR_COR Message if enabled. See Sections 6.2.10.1 and 6.2.10.2.

When DPC is triggered, the Downstream Port immediately Sets the DPC Trigger Status bit and DPC Trigger Reason field to indicate the triggering condition (unmasked uncorrectable error, ERR_NONFATAL, ERR_FATAL, RP_PIO error, or software triggered), and disables its Link by directing the LTSSM to the Disabled state. Once the LTSSM reaches the Disabled state, it remains in that state until the DPC Trigger Status bit is Cleared. To ensure that the LTSSM has time to reach the Disabled state or at least to bring the Link down under a variety of error conditions, software must leave the Downstream Port in DPC until the Data Link Layer Link Active bit in the Link Status register reads 0b; otherwise, the result is undefined. See Section 7.5.3.8. See Section 2.9.3 for other important details on Transaction Layer behavior during DPC.

After DPC has been triggered in a Root Port that supports RP Extensions for DPC, the Root Port may require some time to quiesce and clean up its internal activities, such as those associated with DMA read Requests. When the DPC Trigger Status bit is Set and the DPC RP Busy bit is Set, software must leave the Root Port in DPC until the DPC RP Busy bit reads 0b.

After software releases the Downstream Port from DPC, the Port's LTSSM must transition to the Detect state, where the Link will attempt to retrain. Software can use Data Link Layer State Changed interrupts, DL_Active ERR_COR signaling, or both, to signal when the Link reaches the DL_Active state again. See Sections 6.7.3.3 and 6.2.10.5.

IMPLEMENTATION NOTE

Data Value of All 1's

Many platforms, including those supporting RP Extensions for DPC, can return a data value of all 1's to software when an error is associated with a PCI Express Configuration, I/O, or Memory Read Request. During DPC, the Downstream Port discards Requests destined for the Link and completes them with an error (i.e., either with an Unsupported Request (UR) or Completer Abort (CA) Completion Status). By ending a series of MMIO or configuration space operations with a read to an address with a known data value not equal to all 1's, software may determine if a Completer has been removed or DPC has been triggered.

Also see the Implementation Note "Use of RP PIO Advisory Non-Fatal Errors"

IMPLEMENTATION NOTE

Selecting Non-Posted Request Response During DPC

The DPC Completion Control bit determines how a Downstream Port responds to a Non-Posted Request (NPR) received during DPC. The selection needs to take into account how the rest of the platform handles PCI Express uncorrectable error recovery.

While specific PCI Express uncorrectable error recovery mechanisms in a platform are outside the scope of this specification, here are some guidelines based on general considerations.

If the platform or drivers do not support a PCI Express uncorrectable error recovery strategy, there's no envisioned benefit to enabling DPC, and thus no need to select the NPR response.

If the PCI Express uncorrectable error recovery strategy relies on software detecting containment by looking for all 1's returned by PIO reads, then a UR Completion may be the more appropriate selection, assuming the RP synthesizes an all 1's return value for PIO reads that return UR Completions. The all 1's synthesis would need to occur for PIO reads that target Configuration Space, Memory Space, and perhaps I/O Space.

If the PCI Express uncorrectable error recovery strategy utilizes a mechanism that handles UR and CA Completions differently for PIO reads, then a CA Completion might be the more appropriate selection. CA Completions coming back from a PCI Express device normally indicate a device programming model violation, which may need to trigger Root containment and error recovery.

IMPLEMENTATION NOTE

Selecting the DPC Trigger Condition

Non-Fatal Errors are uncorrectable errors that indicate that a particular TLP was unreliable, and in general the associated Function should not continue its normal operation. Fatal errors are uncorrectable errors that indicate that a particular Link and its related hardware are unreliable, and in general the entire hierarchy below that Link should not continue normal operation. This distinction between Non-Fatal and Fatal errors together with the Root Port error containment capabilities can sometimes be used to select the appropriate DPC trigger condition. The following assumes that there is no peer-to-peer traffic between devices.

Some RCs implement a proprietary feature that will be referred to generically as "Function Level Containment" (FLC). This is not an architected feature of PCI Express. A Root Port that implements FLC is capable of containing the traffic associated with a specific Function when a Non-Fatal Error is detected in that traffic. Switch Downstream Ports below a Root Port with FLC should be configured to trigger DPC when the Downstream Port detects an unmasked uncorrectable error itself or when the Downstream Port receives an ERR_FATAL Message. Under this mode, the Switch Downstream Port passes ERR_NONFATAL Messages it receives Upstream without triggering DPC. This enables Root Port FLC to handle Non-Fatal Errors that render a specific Function unreliable and Switch Downstream Port DPC to handle errors that render a sub-tree of the hierarchy domain unreliable. The Downstream Port still needs to trigger DPC for all unmasked uncorrectable errors it detects, since an ERR_NONFATAL it generates will have its own Requester ID, and the FLC hardware in the Root Port would not be able to determine which specific Function below the Switch Downstream Port was responsible for the Non-Fatal Error.

Switch Downstream Ports below a Root Port without FLC should be configured to trigger DPC when the Switch Downstream Port detects an unmasked uncorrectable error or when the Switch Downstream Port receives an ERR_NONFATAL or ERR_FATAL Message. This enables DPC to contain the error to the affected hierarchy below the Link and allow continued normal operation of the unaffected portion of the hierarchy domain.

IMPLEMENTATION NOTE

Software Polling the DPC RP Busy Bit

The DPC RP Busy bit is a means for hardware to indicate to software that the RP needs to remain in DPC containment while the RP does some internal cleanup and quiescing activities. While the details of these activities are implementation specific, the activities will typically complete within a few microseconds or less. However, under worst-case conditions such as those that might occur with certain internal errors in large systems, the busy period might extend substantially, possibly into multiple seconds. If software is unable to tolerate such lengthy delays within the current software context, software may need to rely on using timer interrupts to schedule polling under interrupt.

IMPLEMENTATION NOTE

Determination of DPC Control

DPC may be controlled in some configurations by platform firmware and in other configurations by the operating system. DPC functionality is strongly linked with the functionality in Advanced Error Reporting. To avoid conflicts over whether platform firmware or the operating system have control of DPC, it is recommended that platform firmware and operating systems always link the control of DPC to the control of Advanced Error Reporting.

6.2.10.1 *DPC Interrupts*

A DPC-capable Downstream Port must support the generation of DPC interrupts. DPC interrupts are enabled by the DPC Interrupt Enable bit in the DPC Control register. DPC interrupts are indicated by the DPC Interrupt Status bit in the DPC Status register.

If the Port is enabled for level-triggered interrupt signaling using INTx messages, the virtual INTx wire must be asserted whenever and as long as the following conditions are satisfied:

- The value of the Interrupt Disable bit in the Command register is 0b.
- The value of the DPC Interrupt Enable bit is 1b.
- The value of the DPC Interrupt Status bit is 1b.

Note that all other interrupt sources within the same Function will assert the same virtual INTx wire when requesting service.

If the Port is enabled for edge-triggered interrupt signaling using MSI or MSI-X, an interrupt message must be sent every time the logical AND of the following conditions transitions from FALSE to TRUE:

- The associated vector is unmasked (not applicable if MSI does not support PVM).
- The value of the DPC Interrupt Enable bit is 1b.
- The value of the DPC Interrupt Status bit is 1b.

The Port may optionally send an interrupt message if interrupt generation has been disabled, and the logical AND of the above conditions is TRUE when interrupt generation is subsequently enabled.

The interrupt message will use the vector indicated by the DPC Interrupt Message Number field in the DPC Capability register. This vector may be the same or may be different from the vectors used by other interrupt sources within this Function.

6.2.10.2 *DPC ERR_COR Signaling*

A DPC-capable Downstream Port must support ERR_COR signaling, independent of whether it supports Advanced Error Reporting (AER) or not. DPC ERR_COR signaling is enabled by the DPC ERR_COR Enable bit in the DPC Control register. DPC triggering is indicated by the DPC

Trigger Status bit in the DPC Status register. DPC ERR_COR signaling is managed independently of DPC interrupts, and it is permitted to use both mechanisms concurrently.

If the DPC ERR_COR Enable bit is Set, and the Correctable Error Reporting Enable bit in the Device Control register is Set, the Port must send an ERR_COR Message each time the DPC Trigger Status bit transitions from Clear to Set. DPC ERR_COR signaling must not Set the Correctable Error Detected bit in the Device Status register, since this event is not handled as an error.

For a given DPC trigger event, if a Port is going to send both an ERR_COR Message and an MSI/MSI-X transaction, then the Port must send the ERR_COR Message prior to sending the MSI/MSI-X transaction. There is no corresponding requirement if the INTx mechanism is being used to signal DPC interrupts, since INTx Messages won't necessarily remain ordered with respect to ERR_COR Messages when passing through routing elements.

IMPLEMENTATION NOTE

Use of DPC ERR_COR Signaling

It is recommended that operating systems use DPC interrupts for signaling when DPC has been triggered. While DPC ERR_COR signaling indicates the same event, DPC ERR_COR signaling is primarily intended for use by platform firmware, when it needs to be notified in order to do its own logging of the event or provide "firmware first" services.

6.2.10.3 *Root Port Programmed I/O (RP PIO) Error Controls*

The RP PIO error control registers enable fine-grained control over what happens when Non-Posted Requests that are tracked by the Root Port encounter certain uncorrectable errors or Advisory Non-Fatal Errors. See Section 2.9.3 for a description of which Non-Posted Requests are tracked. A set of control and status bits exists for receiving Completion with Unsupported Request status (UR Cpl), receiving Completion with Completer Abort status (CA Cpl), and Completion Timeout (CTO) errors. Independent sets of these error bits exist for Configuration Requests, I/O Requests, and Memory Requests. This finer granularity enables more precise error handling for this subset of uncorrectable errors (UR Cpl, CA Cpl, and CTO). As a key example, UR Cpl errors with Memory Read Requests can be configured to trigger DPC for proper containment and error handling, while UR Cpl errors with Configuration Requests can be configured to return all 1's (without trigging DPC) for normal probing and enumeration.

A UR or CA error logged in AER is the result of the Root Port operating in the role of a Completer, and for a received Non-Posted Request, returning a Completion. In contrast, a UR Cpl or CA Cpl error logged as an RP PIO error is the result of the Root Port operating in the role of a Requester, and for an outstanding Non-Posted Request, receiving a Completion. CTO errors logged in both AER and RP PIO are the result of the Root Port operating in the role of a Requester, though the RP PIO error controls support per-space granularity. Depending upon the control register settings, CTO errors can be logged in AER registers, in RP PIO registers, or both. If software unmasks CTO errors in RP PIO, it is recommended that software mask CTO errors in AER in order to avoid unintended interactions.

The RP PIO Header Log, RP PIO ImpSpec Log, and RP PIO TLP Prefix Log registers are referred to collectively as the RP PIO log registers. The RP PIO Header Log must be implemented; the RP PIO ImpSpec Log and RP PIO TLP Prefix Log are optional. The RP PIO Log Size field indicates how many DWORDs are allocated for the RP PIO log registers, and from this the allocated size for the RP PIO TLP Prefix Log can be calculated. See Section 7.9.15.2.

The RP PIO Status, Mask, and Severity registers behave similarly to the Uncorrectable Error Status, Mask, and Severity registers in AER. See Sections 7.8.4.2, 7.8.4.3, and 7.8.4.4. When an RP PIO error is detected while it is unmasked, the associated bit in the RP PIO Status register is Set, and the error is recorded in the RP PIO log registers (assuming that RP PIO error logging resources are available). When an RP PIO error is detected while it is masked, the associated bit is still Set in the RP PIO Status register, but the error does not trigger DPC and the error is not recorded in the RP PIO log registers.

Each unmasked RP PIO error is handled either as an uncorrectable error or an Advisory Non-Fatal Error, as determined by the value of the corresponding bit in the RP PIO Severity register. If the associated Severity bit is Set, the error is handled as uncorrectable, triggering DPC (assuming that DPC is enabled) and signaling this event with a DPC interrupt and/or ERR_COR (if enabled). If the associated Severity bit is Clear, the error is handled as an Advisory Non-Fatal Error (without triggering DPC) and signaled with ERR_COR (if enabled).

IMPLEMENTATION NOTE

Use of RP PIO Advisory Non-Fatal Errors

Each RP PIO error can be handled either as uncorrectable or an Advisory Non-Fatal error. Uncorrectable error handling usually logs the error, triggers DPC, and signals the event either with a DPC interrupt, an ERR_COR, or both. Advisory Non-Fatal Error handling usually logs the error and signals the event with ERR_COR.

RP PIO Advisory Non-Fatal Errors can be used by software in certain cases to handle RP PIO errors robustly without incurring the disruption caused if DPC is triggered in the RP. If an RP PIO Exception is not enabled for a given error, an all 1's value must be returned whenever the error occurs. If the error does not trigger DPC, software may be uncertain if the all 1's value returned by a given PIO read is the actual data value returned by the Completion versus indicating that an error occurred with that PIO read. If software enables Advisory Non-Fatal Error handling for that error, instances of that error will be logged, enabling software to distinguish the two cases.

The use of RP PIO Advisory Non-Fatal Errors is notably beneficial if DPC is triggered in a Switch Downstream Port, and that causes one or more Completion Timeouts in the RP as a side-effect, as described in Section 2.9.3. If the RP handles Completion Timeout errors as Advisory Non-Fatal, this avoids DPC being triggered in the RP, permitting continued operation with the other Switch Downstream Ports.

The RP PIO First Error Pointer, RP PIO Header Log, and RP PIO TLP Prefix Log behave similarly to the First Error Pointer, Header Log, and TLP Prefix Log in AER. The RP PIO First Error Pointer is defined to be valid when its value indicates a bit in the RP PIO Status register that is Set. When the RP PIO First Error Pointer is valid, the RP PIO log registers contain the information associated with the indicated error. The RP PIO ImpSpec Log, if implemented, contains implementation-specific information, e.g., the source of the Request TLP.

In contrast to AER, where the recording of CTO error information in the AER log registers is optional, RP PIO implementations must support recording RP PIO CTO error information in the RP PIO log registers.

The RP PIO SysError register provides a means to generate a System Error when an RP PIO error occurs. If an unmasked RP PIO error is detected while its associated bit in the RP PIO SysError register is Set, a System Error is generated.

The RP PIO Exception register provides a means to generate a synchronous processor exception[95] when an RP PIO error occurs with certain tracked Non-Posted Requests that are generated by a processor instruction. See Section 2.9.3. This exception must support all such tracked read Requests, and may optionally support Configuration write, I/O write, and AtomicOp Requests. If an RP PIO error with an exception-supported Non-Posted Request is detected while its associated bit in the RP PIO Exception register is Set, the processor instruction that generated the Non-Posted Request must take a synchronous exception. This is independent of whether the RP PIO error is masked or not.

The details of a processor instruction taking a synchronous exception are processor-specific, but at a minimum, the mechanism must be able to interrupt the normal processor instruction flow either before completion of the instruction that generated the Non-Posted Request, or immediately following that instruction. The intent is that exception handling routines in system firmware, the operating system, or both, can examine the cause of the exception and take corrective action if necessary.

If an RP PIO error occurs with a processor-generated read or AtomicOp Request, and the RP PIO Exception register value does not cause an exception, a value of all 1's must be returned for the instruction that generated the Request.

[95] "Exception" is used as a generic term for a variety of mechanisms used by processors, including interrupts, traps, machine checks, instruction aborts, etc.

IMPLEMENTATION NOTE

Synchronous Exception Implementation

The exact mechanism for implementing synchronous exceptions is processor and platform specific. One possible implementation is poisoning the data returned to the processor for a read or AtomicOp Request that encounters an error. While this approach is likely to work with those Requests, it might not work with Configuration and I/O write Requests since they return no data.

Another possible implementation is marking the response transaction for processor-generated Non-Posted Requests with some other type of indication of the Request having failed, e.g., a "hard fail" response. This approach is more likely to work with all processor-generated Non-Posted Requests.

IMPLEMENTATION NOTE

RP PIO Mask Bit Behavior and Rationale

For a given RP PIO error, the associated mask bit in the RP PIO Mask register affects its associated status bit setting, error logging, and error signaling in a manner that closely parallels the behavior of mask bits in AER.

SysError generation for a given RP PIO error is primarily controlled by the associated bit in the RP PIO SysError register, but is also contingent upon the associated RP PIO mask bit being Clear. This behavior was chosen for consistency with AER, and also since it is poor practice to generate a SysError without logging the reason.

Exception generation for a given RP PIO error is independent of the associated RP PIO mask bit value. Usage Models are envisioned where an RP PIO error needs to generate an Exception without logging an RP PIO error or triggering DPC.

If a Root Port has outstanding tracked Non-Posted Requests when DPC is triggered, the Root Port is required to synthesize Completions for those tracked Requests, although this may be a logical synthesis, and not the actual creation of Completion TLPs. See Section 2.9.3. These synthesized Completions shall have a UR or CA Completion Status, as determined by the value of the DPC Completion Control bit. The UR/CA Completion Status, combined with the associated Request's targeted Space (Configuration, I/O, or Memory), then determines which RP PIO error control bits are used to govern the remainder of the error handling for each outstanding tracked Request. For example, a Root Port handling the case of a Configuration Read receiving a synthesized UR Completion would use the Cfg UR Cpl bit (bit 0) in the RP PIO SysError and RP PIO Exception registers to determine whether to generate a System Error, generate a processor instruction exception, or both.

Root Port error handling for tracked Non-Posted Requests with errors other than receiving UR and CA Completions is governed by a combination of AER and RP PIO error controls. Examples are CTO[96], Poisoned TLP Received, and Malformed TLP. For a given error managed by AER, the

[96] CTO errors have status and mask bits in both AER and RP PIO, though RP PIO has independent sets of bits for each of the 3 spaces. Other errors in AER have no equivalent errors in RP PIO.

associated AER Mask and Severity bits partially or completely determine if the error must be handled as an uncorrectable error, handled as an Advisory Non-Fatal Error, or handled as a masked error.

- If the AER-managed error is to be handled as an uncorrectable error (see Section 6.2.2.2), DPC is triggered. As described earlier, Completions are synthesized for any outstanding tracked Non-Posted Requests, and for each Request, the RP PIO SysError and RP PIO Exception bits associated with the Request type and Completion Status apply.
- If the AER-managed error is to be handled as an Advisory Non-Fatal Error (see Section 6.2.3.2.4) or a masked error (see Section 6.2.3.2.2), DPC is not triggered, so the RP PIO SysError and RP PIO Exception bits do not apply.
- One notable example is the case of a Root Port with an outstanding PIO read receiving an associated Completion that is poisoned. If the AER Severity bit is Set, forcing the error to be handled as an uncorrectable error, the RP PIO Exception bit determines if all 1's is returned versus generating an exception. If the error is to be handled as an Advisory Non-Fatal Error as described in Section 6.2.3.2.4.3, the poisoned data is propagated internally by the Root Port, and from there is handled in a platform specific manner. If the error is to be handled as a masked error, the poisoned data is handled in a platform specific manner.

6.2.10.4 *Software Triggering of DPC*

If the DPC Software Triggering Supported bit in the DPC Capability register is Set, then software can trigger DPC by writing a 1b to the DPC Software Trigger bit in the DPC Control register, assuming that DPC is enabled and the Port isn't currently in DPC. This mechanism is envisioned to be useful for software and/or firmware development and testing. It also supports usage models where software or firmware examines RP PIO Exceptions or RP PIO Advisory Non-Fatal Errors, and decides to trigger DPC based upon the situation.

When this mechanism triggers DPC, the DPC Trigger Reason and DPC Trigger Reason Extension fields in the DPC Status register will indicate this as the reason.

If a Port is already in DPC when a 1b is written to the DPC Software Trigger bit, the Port remains in DPC, and the DPC Trigger Reason and DPC Trigger Reason Extension fields are not modified.

IMPLEMENTATION NOTE

Avoid Disable Link and Hot-Plug Surprise Use With DPC

It is recommended that software not Set the Link Disable bit in the Link Control register while DPC is enabled but not triggered. Setting the Link Disable bit will cause the Link to be directed to DL_Down, invoking some semantics similar to those in DPC, but lacking others. The subsequent arrival of Posted Requests will likely trigger DPC soon, anyway.

Similarly, it is recommended that a Port that supports DPC not Set the Hot-Plug Surprise bit in the Slot Capabilities register. Having this bit Set blocks the reporting of Surprise Down errors, preventing DPC from being triggered by this important error, greatly reducing the benefit of DPC.

6.2.10.5 *DL_Active ERR_COR Signaling*

Support for this feature is indicated by the DL_Active ERR_COR Signaling Supported bit in the DPC Capability register. The feature is enabled by the DL_ACTIVE ERR_COR Enable bit in the DPC Control register. The DL_ACTIVE state is indicated by the Data Link Layer Link Active bit in the Link Status register. DL_ACTIVE ERR_COR signaling is managed independently of Data Link Layer State Changed interrupts, and it is permitted to use both mechanisms concurrently.

If the DL_ACTIVE ERR_COR Enable bit is Set, and the Correctable Error Reporting Enable bit in the Device Control register is Set, the Port must send an ERR_COR Message each time the Link transitions into the DL_ACTIVE state. DL_ACTIVE ERR_COR signaling must not Set the Correctable Error Detected bit in the Device Status register, since this event is not handled as an error. In contrast to Data Link Layer State Changed interrupts, DL_Active ERR_COR signaling only indicates the Link enters the DL_Active state, not when the Link exits the DL_Active state.

For a given DL_ACTIVE event, if a Port is going to send both an ERR_COR Message and an MSI/MSI-X transaction, then the Port must send the ERR_COR Message prior to sending the MSI/MSI-X transaction. There is no corresponding requirement if the INTx mechanism is being used to signal DL_ACTIVE interrupts, since INTx Messages won't necessarily remain ordered with respect to ERR_COR Messages when passing through routing elements.

IMPLEMENTATION NOTE

Use of DL_ACTIVE ERR_COR Signaling

It is recommended that operating systems use Data Link Layer State Changed interrupts for signaling when DL_ACTIVE changes state. While DL_ACTIVE ERR_COR signaling indicates a subset of the same events, DL_ACTIVE ERR_COR signaling is primarily intended for use by platform firmware, when it needs to be notified in order to do Downstream Port configuration or provide "firmware first" services.

6.3 Virtual Channel Support

6.3.1 Introduction and Scope

The Virtual Channel mechanism provides a foundation for supporting differentiated services within the PCI Express fabric. It enables deployment of independent physical resources that together with traffic labeling are required for optimized handling of differentiated traffic. Traffic labeling is supported using Traffic Class TLP-level labels. The policy for traffic differentiation is determined by the TC/VC mapping and by the VC-based, Port-based, and Function-based arbitration mechanisms. The TC/VC mapping depends on the platform application requirements. These requirements drive the choice of the arbitration algorithms and configurability/programmability of arbiters allows detailed tuning of the traffic servicing policy.

The definition of the Virtual Channel and associated Traffic Class mechanisms is covered in Chapter 2. The VC configuration/programming model is defined in Sections 7.9.1 and 7.9.2.

This section covers VC mechanisms from the system perspective. It addresses the next level of details on:

- Supported TC/VC configurations
- VC-based arbitration – algorithms and rules
- Traffic ordering considerations
- Isochronous support as a specific usage model

6.3.2 TC/VC Mapping and Example Usage

A Virtual Channel is established when one or more TCs are associated with a physical resource designated by a VC ID. Every Traffic Class that is supported on a given path within the fabric must be mapped to one of the enabled Virtual Channels. Every Port must support the default TC0/VC0 pair – this is "hardwired". Any additional TC mapping or additional VC resource enablement is optional and is controlled by system software using the programming model described in Sections 7.9.1 and 7.9.2.

The number of VC resources provisioned within a component or enabled within a given fabric may vary due to implementation and usage model requirements, due to Hot-Plug of disparate components with varying resource capabilities, or due to system software restricting what resources may be enabled on a given path within the fabric.

Some examples to illustrate:

- A set of components (Root Complex, Endpoints, Switches) may only support the mandatory VC0 resource that must have TC0 mapped to VC0. System software may, based on application usage requirements, map one or all non-zero TCs to VC0 as well on any or all paths within the fabric.
- A set of components may support two VC resources, e.g., VC0 and VC1. System software must map TC0/VC0 and in addition, may map one or all non-zero TC labels to either VC0 or VC1. As above, these mappings may be enabled on any or all paths within the fabric. Refer to the examples below for additional information.
- A Switch may be implemented with eight Ports – seven x1 Links with two VC resources and one x16 Link with one VC resource. System software may enable both VC resources on the x1 Links and assign one or more additional TCs to either VC thus allowing the Switch to differentiate traffic flowing between any Ports. The x16 Link must also be configured to map any non-TC0 traffic to VC0 if such traffic is to flow on this Link. Note: multi-Port components (Switches and Root Complex) are required to support independent TC/VC mapping per Port.

In any of the above examples, system software has the ability to map one, all, or a subset of the TCs to a given VC. Should system software wish to restrict the number of traffic classes that may flow through a given Link, it may configure only a subset of the TCs to the enabled VC resources. Any TLP indicating a TC that has not been mapped to an enabled VC resource must be treated as a Malformed TLP. This is referred to as TC Filtering. Flow Control credits for this TLP will be lost, and an uncorrectable error will be generated, so software intervention will usually be required to restore proper operation after a TC Filtering event occurs.

A graphical example of TC filtering is illustrated in Figure 6-4, where TCs (2:6) are not mapped to the Link that connects Endpoint A and the Switch. This means that the TLPs with TCs (2:6) are not allowed between the Switch and Endpoint A.

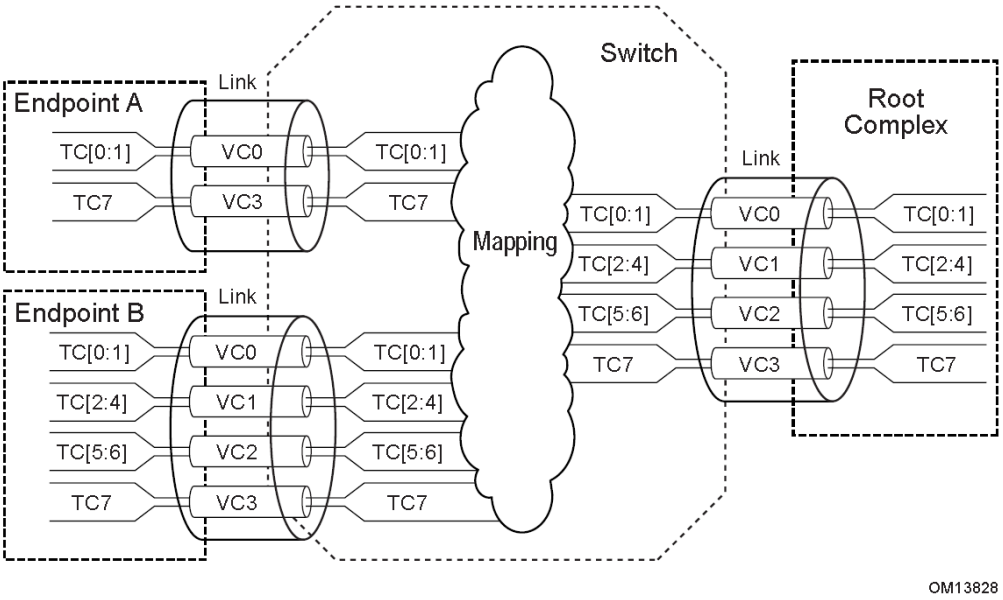


Figure 6-4: TC Filtering Example

Figure 6-5 shows an example of TC to VC mapping. A simple Switch with one Downstream Port and one Upstream Port connects an Endpoint to a Root Complex. At the Upstream Port, two VCs (VC0 and VC1) are enabled with the following mapping: TC(0-6)/VC0, TC7/VC1. At the Downstream Port, only VC0 is enabled and all TCs are mapped to VC0. In this example while TC7 is mapped to VC0 at the Downstream Port, it is re-mapped to VC1 at the Upstream Port. Although the Endpoint only supports VC0, when it labels transactions with different TCs, transactions associated with TC7 from/to the Endpoint can take advantage of the second Virtual Channel enabled between the Switch and the Root Complex.

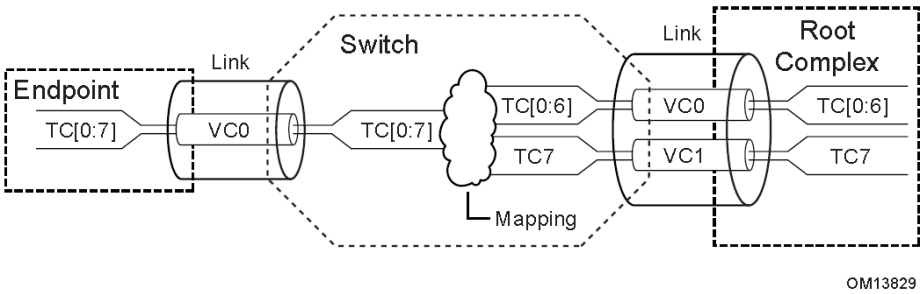


Figure 6-5: TC to VC Mapping Example

IMPLEMENTATION NOTE

Multiple TCs Over a Single VC

A single VC implementation may benefit from using multiple TCs. TCs provide ordering domains that may be used to differentiate traffic within the Endpoint or the Root Complex independent of the number of VCs supported.

In a simple configuration, where only VC0 is supported, traffic differentiation may not be accomplished in an optimum manner since the different TCs cannot be physically segregated. However, the benefits of carrying multiple TCs can still be exploited particularly in the small and "shallow" topologies where Endpoints are connected directly to Root Complex rather than through cascaded Switches. In these topologies traffic that is targeting Root Complex only needs to traverse a single Link, and an optimized scheduling of packets on both sides (Endpoint and Root Complex) based on TCs may accomplish significant improvement over the case when a single TC is used. Still, the inability to route differentiated traffic through separate resources with fully independent flow control and independent ordering exposes all of the traffic to the potential head-of-line blocking conditions. Optimizing Endpoint internal architecture to minimize the exposure to the blocking conditions can reduce those risks.

6.3.3 VC Arbitration

Arbitration is one of the key aspects of the Virtual Channel mechanism and is defined in a manner that fully enables configurability to the specific application. In general, the definition of the VC-based arbitration mechanism is driven by the following objectives:

- To prevent false transaction timeouts and to guarantee data flow forward progress
- To provide differentiated services between data flows within the fabric
- To provide guaranteed bandwidth with deterministic (and reasonably small) end-to-end latency between components

Links are bidirectional, i.e., each Port can be an Ingress or an Egress Port depending on the direction of traffic flow. This is illustrated by the example of a 3-Port Switch in Figure 6-6, where the paths for traffic flowing between Switch Ports are highlighted with different types of lines. In the following sections, VC Arbitration is defined using a Switch arbitration model since the Switch represents a functional superset from the arbitration perspective.

In addition, one-directional data flow is used in the description.

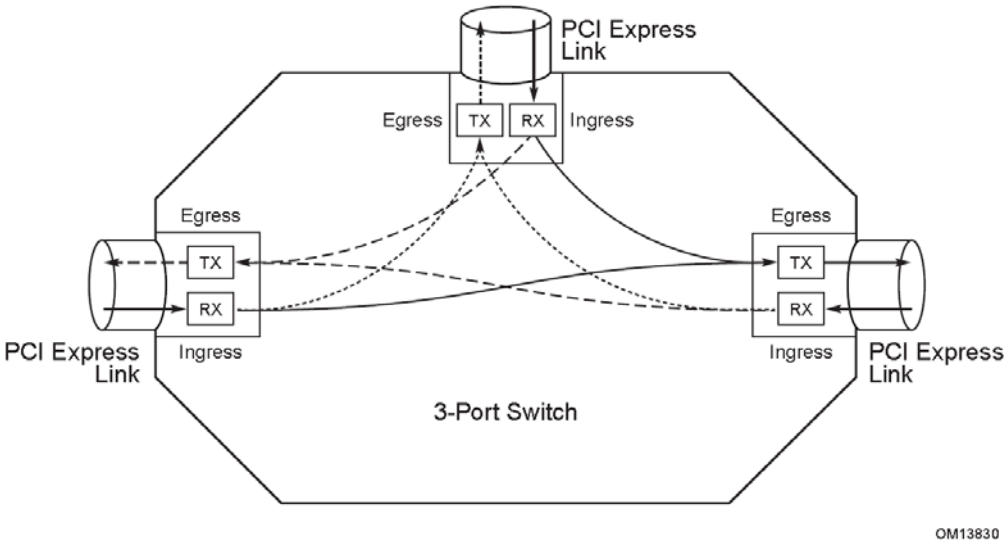


Figure 6-6: An Example of Traffic Flow Illustrating Ingress and Egress

6.3.3.1 *Traffic Flow and Switch Arbitration Model*

The following set of figures (Figure 6-7 and Figure 6-8) illustrates traffic flow through the Switch and summarizes the key aspects of the arbitration.

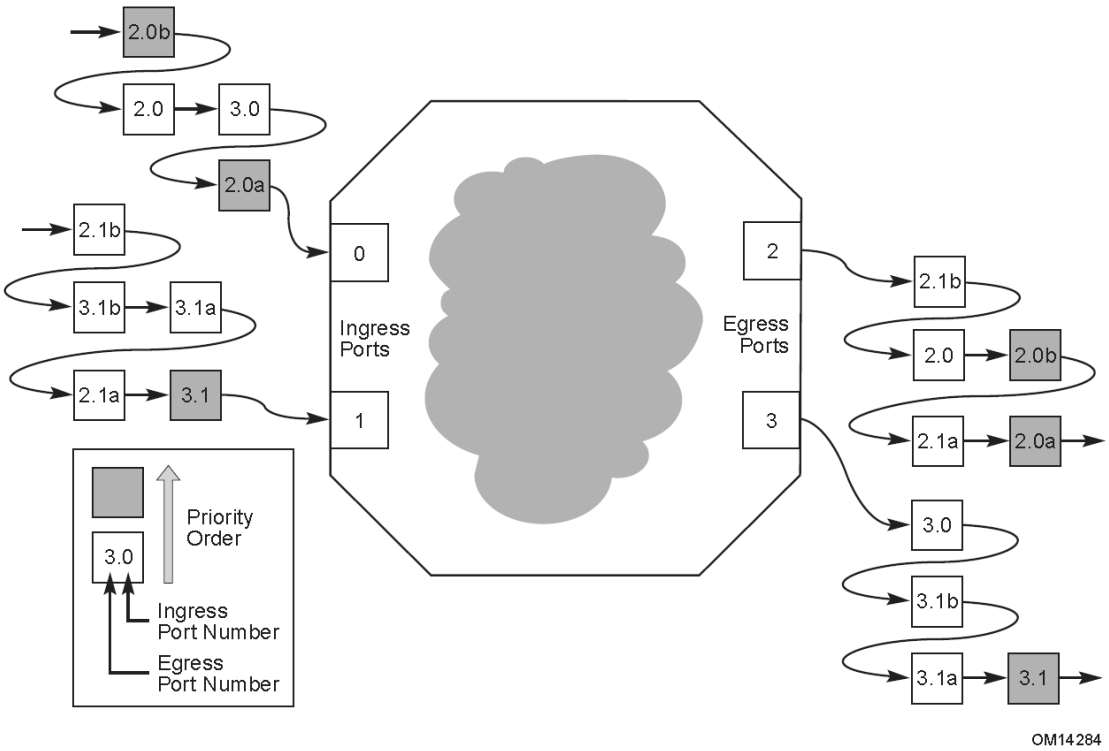


Figure 6-7: An Example of Differentiated Traffic Flow Through a Switch

At each Ingress Port an incoming traffic stream is represented in Figure 6-7 by small boxes. These boxes represent packets that are carried within different VCs that are distinguished using different levels of gray. Each of the boxes that represents a packet belonging to different VC includes designation of Ingress and Egress Ports to indicate where the packet is coming from and where it is going to. For example, designation "3.0" means that this packet is arriving at Port #0 (Ingress) and is destined to Port #3 (Egress). Within the Switch, packets are routed and serviced based on Switch internal arbitration mechanisms.

Switch arbitration model defines a required arbitration infrastructure and functionality within a Switch. This functionality is needed to support a set of arbitration policies that control traffic contention for an Egress Port from multiple Ingress Ports.

Figure 6-8 shows a conceptual model of a Switch highlighting resources and associated functionality in ingress to egress direction. Note that each Port in the Switch can have the role of an Ingress or Egress Port. Therefore, this figure only shows one particular scenario where the 4-Port Switch in this example has ingress traffic on Port #0 and Port #1, that targets Port #2 as an Egress Port. A different example may show different flow of traffic implying different roles for Ports on the Switch. The PCI Express architecture enables peer-to-peer communication through the Switch and, therefore, possible scenarios using the same example may include multiple separate and simultaneous ingress to egress flows (e.g., Port 0 to Port 2 and Port 1 to Port 3).

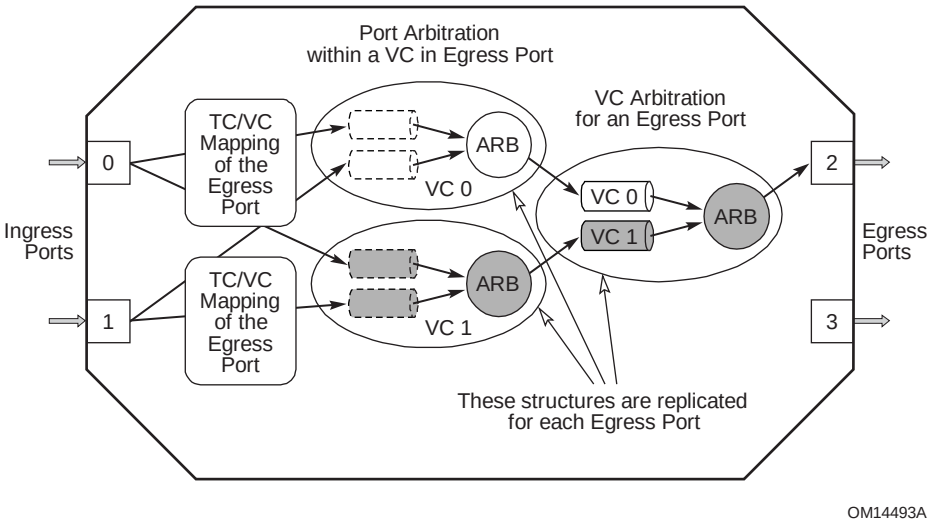


Figure 6-8: Switch Arbitration Structure

The following two steps conceptually describe routing of traffic received by the Switch on Port 0 and Port 1 and destined to Port 2. First, the target Egress Port is determined based on address/routing information in the TLP header. Secondly, the target VC of the Egress Port is determined based on the TC/VC map of the Egress Port. Transactions that target the same VC in the Egress Port but are from different Ingress Ports must be arbitrated before they can be forwarded to the corresponding resource in the Egress Port. This arbitration is referred to as the Port Arbitration.

Once the traffic reaches the destination VC resource in the Egress Port, it is subject to arbitration for the shared Link. From the Egress Port point of view this arbitration can be conceptually defined as a simple form of multiplexing where the multiplexing control is based on arbitration policies that

are either fixed or configurable/programmable. This stage of arbitration between different VCs at an Egress Port is called the VC Arbitration of the Egress Port.

Independent of VC arbitration policy, a management/control logic associated with each VC must observe transaction ordering and flow control rules before it can make pending traffic visible to the arbitration mechanism.

IMPLEMENTATION NOTE

VC Control Logic at the Egress Port

VC control logic at every Egress Port includes:

- VC Flow Control logic
- VC Ordering Control logic

Flow control credits are exchanged between two Ports connected to the same Link. Availability of flow control credits is one of the qualifiers that VC control logic must use to decide when a VC is allowed to compete for the shared Link resource (i.e., Data Link Layer transmit/retry buffer). If a candidate packet cannot be submitted due to the lack of an adequate number of flow control credits, VC control logic must mask the presence of pending packet to prevent blockage of traffic from other VCs. Note that since each VC includes buffering resources for Posted Requests, Non-Posted Requests, and Completion packets, the VC control logic must also take into account availability of flow control credits for the particular candidate packet. In addition, VC control logic must observe ordering rules (see Section 2.4 for more details) for Posted/Non-Posted/Completion transactions to prevent deadlocks and violation of producer/consumer ordering model.

6.3.3.2 *VC Arbitration – Arbitration Between VCs*

This specification defines a default VC prioritization via the VC Identification (VC ID) assignment, i.e., the VC IDs are arranged in ascending order of relative priority in the Virtual Channel Capability structure or Multi-Function Virtual Channel Capability structure. The example in Figure 6-9 illustrates a Port that supports eight VCs with VC0 treated as the lowest priority and VC7 as the highest priority.

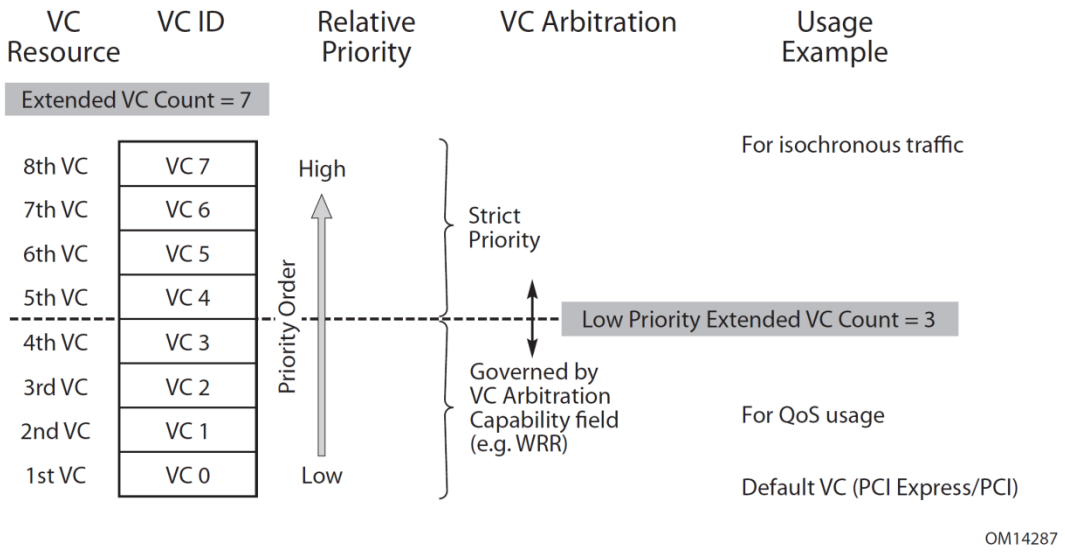


Figure 6-9: VC ID and Priority Order – An Example

The availability of default prioritization does not restrict the type of algorithms that may be implemented to support VC arbitration – either implementation-specific or one of the architecture-defined methods:

- Strict Priority – Based on inherent prioritization, i.e., VC0 = lowest, VC7 = highest
- Round Robin (RR) – Simplest form of arbitration where all VCs have equal priority
- Weighted RR – Programmable weight factor determines the level of service

If strict priority arbitration is supported by the hardware for a subset of the VC resources, software can configure the VCs into two priority groups – a lower and an upper group. The upper group is treated as a strict priority arbitration group while the lower group is arbitrated to only when there are no packets to process in the upper group. Figure 6-9 illustrates an example configuration that supports eight VCs separated into two groups – the lower group consisting of VC0-VC3 and the upper group consisting of VC4-VC7. The arbitration within the lower group can be configured to one of the supported arbitration methods. The Low Priority Extended VC Count field in the Port VC Capability register 1 indicates the size of this group. The arbitration methods are listed in the VC Arbitration Capability field in the Port VC Capability register 2. Refer to Sections 7.9.1 and 7.9.2 for details. When the Low Priority Extended VC Count field is set to zero, all VCs are governed by the strict-priority VC arbitration; when the field is equal to the Extended VC Count, all VCs are governed by the VC arbitration indicated by the VC Arbitration Capability field.

6.3.3.2.1 Strict Priority Arbitration Model

Strict priority arbitration enables minimal latency for high-priority transactions. However, there is potential danger of bandwidth starvation should it not be applied correctly. Using strict priority requires all high-priority traffic to be regulated in terms of maximum peak bandwidth and Link usage duration. Regulation must be applied either at the transaction injection Port/Function or within subsequent Egress Ports where data flows contend for a common Link. System software must configure traffic such that lower priority transactions will be serviced at a sufficient rate to avoid transaction timeouts.

6.3.3.2.2 Round Robin Arbitration Model

Round Robin arbitration is used to provide, at the transaction level, equal[97] opportunities to all traffic. Note that this scheme is used where different unordered streams need to be serviced with the same priority.

In the case where differentiation is required, a Weighted Round Robin scheme can be used. The WRR scheme is commonly used in the case where bandwidth regulation is not enforced by the sources of traffic and therefore it is not possible to use the priority scheme without risking starvation of lower priority traffic. The key is that this scheme provides fairness during traffic contention by allowing at least one arbitration win per arbitration loop. Assigned weights regulate both minimum allowed bandwidth and maximum burstiness for each VC during the contention. This means that it bounds the arbitration latency for traffic from different VCs. Note that latencies are also dependent on the maximum packet sizes allowed for traffic that is mapped onto those VCs.

One of the key usage models of the WRR scheme is support for QoS policy where different QoS levels can be provided using different weights.

Although weights can be fixed (by hardware implementation) for certain applications, to provide more generic support for different applications, components that support the WRR scheme are recommended to implement programmable WRR. Programming of WRR is controlled using the software interface defined in Sections 7.9.1 and 7.9.2.

6.3.3.3 *Port Arbitration – Arbitration Within VC*

For Switches, Port Arbitration refers to the arbitration at an Egress Port between traffic coming from other Ingress Ports that is mapped to the same VC. For Root Ports, Port Arbitration refers to the arbitration at a Root Egress Port between peer-to-peer traffic coming from other Root Ingress Ports that is mapped to the same VC. For RCRBs, Port Arbitration refers to the arbitration at the RCRB (e.g., for host memory) between traffic coming from Root Ports that is mapped to the same VC. An inherent prioritization scheme for arbitration among VCs in this context is not applicable since it would imply strict arbitration priority for different Ports. Traffic from different Ports can be arbitrated using the following supported schemes:

- Hardware-fixed arbitration scheme, e.g., Round Robin
- Programmable WRR arbitration scheme
- Programmable Time-based WRR arbitration scheme

---

[97] Note that this does not imply equivalence and fairness in the terms of bandwidth usage.

Hardware-fixed RR or RR-like scheme is the simplest to implement since it does not require any programmability. It makes all Ports equal priority, which is acceptable for applications where no software-managed differentiation or per-Port-based bandwidth budgeting is required.

Programmable WRR allows flexibility since it can operate as flat RR or if differentiation is required, different weights can be applied to traffic coming from different Ports in the similar manner as described in Section 6.3.3.2. This scheme is used where different allocation of bandwidth needs to be provided for different Ports.

A Time-based WRR is used for applications where not only different allocation of bandwidth is required but also a tight control of usage of that bandwidth. This scheme allows control of the amount of traffic that can be injected from different Ports within a certain fixed period of time. This is required for certain applications such as isochronous services, where traffic needs to meet a strict deadline requirement. Section 6.3.4 provides basic rules to support isochronous applications. For more details on time-based arbitration and on the isochronous service as a usage model for this arbitration scheme refer to Appendix A.

6.3.3.4 *Multi-Function Devices and Function Arbitration*

The multi-Function arbitration model defines an optional arbitration infrastructure and functionality within a Multi-Function Device. This functionality is needed to support a set of arbitration policies that control traffic contention for the device's Upstream Egress Port from its multiple Functions.

Figure 6-10 shows a conceptual model of a Multi-Function Device highlighting resources and associated functionality. Note that each Function optionally contains a VC Capability structure, which if present manages TC/VC mapping, optional Port Arbitration, and optional VC Arbitration, all within the Function. The MFVC Capability structure manages TC/VC mapping, optional Function Arbitration, and optional VC Arbitration for the device's Upstream Egress Port. Together these resources enable enhanced QoS management for Upstream requests. However, unlike a complete Switch with devices on its Downstream Ports, the Multi-Function Device model does not support full QoS management for peer-to-peer requests between Functions or for Downstream requests.

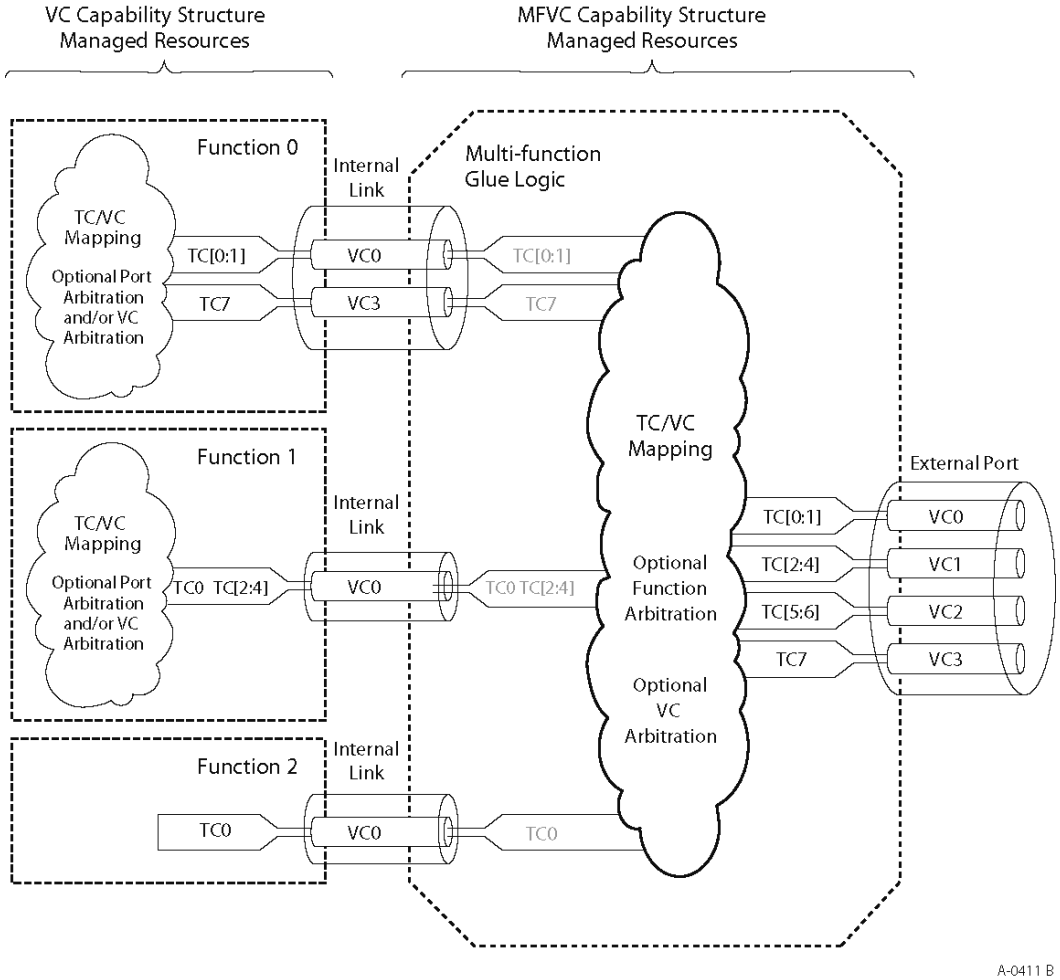


Figure 6-10: Multi-Function Arbitration Model

QoS for an Upstream request originating at a Function is managed as follows. First, a Function-specific mechanism applies a TC to the request. For example, a device driver might configure a Function to tag all its requests with TC7.

Next, if the Function contains a VC Capability structure, it specifies the TC/VC mapping to one of the Function's VC resources (perhaps the Function's single VC resource). In addition, the VC Capability structure supports the enablement and configuration of the Function's VC resources.

If the Function is a Switch and the target VC resource supports Port Arbitration, this mechanism governs how the Switch's multiple Downstream Ingress Ports arbitrate for that VC resource. If the Port Arbitration mechanism supports time-based WRR, this also governs the injection rate of requests from each Downstream Ingress Port.

If the Function supports VC arbitration, this mechanism manages how the Function's multiple VC resources arbitrate for the conceptual internal link to the MFVC resources.

Once a request packet conceptually arrives at MFVC resources, address/routing information in the TLP header determines whether the request goes Upstream or peer-to-peer to another Function. For the case of peer-to-peer, QoS management is left to unarchitected device-specific mechanisms. For the case of Upstream, TC/VC mapping in the MFVC Capability structure determines which VC resource the request will target. The MFVC Capability structure also supports enablement and configuration of the VC resources in the multi-Function glue logic. If the target VC resource supports Function Arbitration, this mechanism governs how the multiple Functions arbitrate for this VC resource. If the Function Arbitration mechanism supports time-based WRR, this governs the injection rate of requests for each Function into this VC resource.

Finally, if the MFVC Capability structure supports VC Arbitration, this mechanism governs how the MFVC's multiple VCs compete for the device's Upstream Egress Port. Independent of VC arbitration policy, management/control logic associated with each VC must observe transaction ordering and flow control rules before it can make pending traffic visible to the arbitration mechanism.

IMPLEMENTATION NOTE

Multi-Function Devices without the MFVC Capability Structure

If a Multi-Function Device lacks an MFVC Capability structure, the arbitration of data flows from different Functions of a Multi-Function Device is beyond the scope of this specification However, if a Multi-Function Device supports TCs other than TC0 and does not implement an MFVC Capability structure, it must implement a single VC Capability structure in Function 0 to provide architected TC/VC mappings for the Link.

6.3.4 Isochronous Support

Servicing isochronous data transfer requires a system to provide not only guaranteed data bandwidth but also deterministic service latency. The isochronous support mechanisms are defined to ensure that isochronous traffic receives its allocated bandwidth over a relevant period of time while also preventing starvation of the other traffic in the system. Isochronous support mechanisms apply to communication between Endpoint and Root Complex as well as to peer-to-peer communication.

Isochronous service is realized through proper use of mechanisms such as TC transaction labeling, VC data-transfer protocol, and TC-to-VC mapping. End-to-end isochronous service requires software to set up proper configuration along the path between the Requester and the Completer. This section describes the rules for software configuration and the rules hardware components must follow to provide end-to-end isochronous services. More information and background material regarding isochronous applications and isochronous service design guidelines can be found in Appendix A.

6.3.4.1 *Rules for Software Configuration*

System software must obey the following rules to configure PCI Express fabric for isochronous traffic:

- Software must designate one or more TCs for isochronous transactions.
- Software must ensure that the Attribute fields of all isochronous requests targeting the same Completer are fixed and identical.
- Software must configure all VC resources used to support isochronous traffic to be serviced (arbitrated) at the requisite bandwidth and latency to meet the application objectives. This may be accomplished using strict priority, WRR, or hardware-fixed arbitration.
- Software should not intermix isochronous traffic with non-isochronous traffic on a given VC.
- Software must observe the Maximum Time Slots capability reported by the Port or RCRB.
- Software must not assign all Link capacity to isochronous traffic. This is required to ensure the requisite forward progress of other non-isochronous transactions to avoid false transaction timeouts.
- Software must limit the Max_Payload_Size for each path that supports isochronous to meet the isochronous latency. For example, all traffic flowing on a path from an isochronous capable device to the Root Complex should be limited to packets that do not exceed the Max_Payload_Size required to meet the isochronous latency requirements.
- Software must set Max_Read_Request_Size of an isochronous-configured device with a value that does not exceed the Max_Payload_Size set for the device.

6.3.4.2 *Rules for Requesters*

A Requester requiring isochronous services must obey the following rules:

- The value in the Length field of read requests must never exceed Max_Payload_Size.
- If isochronous traffic targets the Root Complex and the RCRB indicates it cannot meet the isochronous bandwidth and latency requirements without requiring all transactions to set the No Snoop attribute bit, indicated by setting the Reject Snoop Transactions bit, then this bit must be set within the TLP header else the transaction will be rejected.

6.3.4.3 *Rules for Completers*

A Completer providing isochronous services must obey the following rules:

- A Completer should not apply flow control induced backpressure to uniformly injected isochronous requests under normal operating conditions.
- A Completer must report its isochronous bandwidth capability in the Maximum Time Slots field in the VC Resource Capability register. Note that a Completer must account for partial writes.
- A Completer must observe the maximum isochronous transaction latency.
- A Root Complex as a Completer must implement at least one RCRB and support time-based Port Arbitration for the associated VCs. Note that time-based Port Arbitration only applies to request transactions.

6.3.4.4 *Rules for Switches and Root Complexes*

A Switch providing isochronous services must obey the following rules. The same rules apply to Root Complexes that support isochronous data flows peer-to-peer between Root Ports, abbreviated in this section as "P2P-RC".

- An isochronous-configured Switch or P2P-RC Port should not apply flow control induced backpressure to uniformly injected isochronous requests under normal operating conditions.
- An isochronous-configured Switch or P2P-RC Port must observe the maximum isochronous transaction latency.
- A Switch or P2P-RC component must support time-based Port Arbitration for each Port that supports one or more VCs capable of supporting isochronous traffic. Note that time-based Port Arbitration applies to request transactions but not to completion transactions.

6.3.4.5 *Rules for Multi-Function Devices*

A Multi-Function Device that includes an MFVC Capability structure providing isochronous services must obey the following rules:

- MFVC glue logic configured for isochronous operation should not apply backpressure to uniformly injected isochronous requests from its Functions under normal operating conditions.
- The MFVC Capability structure must support time-based Function Arbitration for each VC capable of supporting isochronous traffic. Note that time-based Function Arbitration applies only to Upstream request transactions; it does not apply to any Downstream or peer-to-peer request transactions, nor to any completion transactions.

A Multi-Function Device that lacks an MFVC Capability structure has no architected mechanism to provide isochronous services for its multiple Functions concurrently.

6.4 Device Synchronization

System software requires a "stop" mechanism for ensuring that there are no outstanding transactions for a particular device in a system. For example, without such a mechanism renumbering Bus Numbers during system operation may cause the Requester ID (which includes the Bus Number) for a given device to change while Requests or Completions for that device are still in flight, and may thus be rendered invalid due to the change in the Requester ID. It is also desirable to be able to ensure that there are no outstanding transactions during a Hot-Plug orderly removal.

The details of stop mechanism implementation depend on the device hardware, device driver software, and system software. However, the fundamental requirements which must be supported to allow system software management of the fabric include the abilities to:

- Block the device from generating new Requests
- Block the generation of Requests issued to the device
- Determine that all Requests being serviced by the device have been completed
- Determine that all non-posted Requests initiated by the device have completed
- Determine that all posted Requests initiated by the device have reached their destination

The ability of the driver and/or system software to block new Requests from the device is supported by the Bus Master Enable, SERR# Enable, and Interrupt Disable bits in the Command register (Section7.5.1.1.3) of each device Function, and other such control bits.

Requests issued to the device are generally under the direct control of the driver, so system software can block these Requests by directing the driver to stop generating them (the details of this communication are system software specific). Similarly, Requests serviced by the device are normally under the device driver's control, so determining the completion of such requests is usually trivial.

The Transactions Pending bit provides a consistent way on a per-Function basis for software to determine that all non-posted Requests issued by the device have been completed (see Section 7.5.3.5).

Determining that posted Requests have reached their destination is handled by generating a transaction to "flush" any outstanding Requests. Writes to system memory using TC0 will be flushed by host reads of the device, and so require no explicit flush protocol. Writes using TCs other than TC0 require some type of flush synchronization mechanism. The mechanism itself is implementation specific to the device and its driver software. However, in all cases the device hardware and software implementers should thoroughly understand the ordering rules described in Section 2.4. This is especially true if the Relaxed Ordering or ID-Based Ordering attributes are set for any Requests initiated by the device.

IMPLEMENTATION NOTE

Flush Mechanisms

In a simple case such as that of an Endpoint communicating only with host memory through TC0, "flush" can be implemented simply by reading from the Endpoint. If the Endpoint issues writes to main memory using TCs other than TC0, "flush" can be implemented with a memory read on the corresponding TCs directed to main memory. The memory read needs to be performed on all TCs that the Endpoint is using.

If a memory read is used to "flush" outstanding transactions, but no actual read is required, it may be desirable to use the zero-length read semantic described in Section 2.2.5.

Peer-to-peer interaction between devices requires an explicit synchronization protocol between the involved devices, even if all communication is through TC0. For a given system, the model for managing peer-to-peer interaction must be established. System software, and device hardware and software must then conform to this model. The requirements for blocking Request generation and determining completion of Requests match the requirements for non-peer interaction, however the determination that Posted Requests have reached peer destination device(s) requires an explicit synchronization mechanism. The mechanism itself is implementation specific to the device, its driver software, and the model used for the establishment and disestablishment of peer communications.

6.5 Locked Transactions

6.5.1 Introduction

Locked Transaction support is required to prevent deadlock in systems that use legacy software which causes the accesses to I/O devices. Note that some CPUs may generate locked accesses as a result of executing instructions that implicitly trigger lock. Some legacy software misuses these transactions and generates locked sequences even when exclusive access is not required. Since locked accesses to I/O devices introduce potential deadlocks apart from those mentioned above, as well as serious performance degradation, PCI Express Endpoints are prohibited from supporting locked accesses, and new software must not use instructions which will cause locked accesses to I/O devices. Legacy Endpoints support locked accesses only for compatibility with existing software.

Only the Root Complex is allowed to initiate Locked Requests on PCI Express. Locked Requests initiated by Endpoints and Bridges are not supported. This is consistent with limitations for locked transaction use outlined in the *PCI Local Bus Specification* (Appendix F- Exclusive Accesses).

This section specifies the rules associated with supporting locked accesses from the Host CPU to Legacy Endpoints, including the propagation of those transactions through Switches and PCI Express/PCI Bridges.

6.5.2 Initiation and Propagation of Locked Transactions - Rules

Locked transaction sequences are generated by the Host CPU(s) as one or more reads followed by a number of writes to the same location(s). When a lock is established, all other traffic is blocked from using the path between the Root Complex and the locked Legacy Endpoint or Bridge.

- A locked transaction sequence or attempted locked transaction sequence is initiated on PCI Express using the "lock"–type Read Request/Completion (MRdLk/CplDLk) and terminated with the Unlock Message
  - Locked Requests which are completed with a status other than Successful Completion do not establish lock (explained in detail in the following sections)
  - Regardless of the status of any of the Completions associated with a locked sequence, all locked sequences and attempted locked sequences must be terminated by the transmission of an Unlock Message.
  - MRdLk, CplDLk, and Unlock semantics are allowed only for the default Traffic Class (TC0)
  - Only one locked transaction sequence attempt may be in progress at a given time within a single hierarchy domain
- The Unlock Message is sent from the Root Complex down the locked transaction path to the Completer, and may be broadcast from the Root Complex to all Endpoints and Bridges
  - Any device which is not involved in the locked sequence must ignore this Message
- Any violation of the rules for initiation and propagation of locked transactions can result in undefined device and/or system behavior
  - The initiation and propagation of a locked transaction sequence through PCI Express is performed as follows:
- A locked transaction sequence is started with a MRdLk Request
  - Any successive reads for the locked transaction sequence must also use MRdLk Requests
  - The Completions for any MRdLk Request use the CplDLk Completion type for successful Requests, and the CplLk Completion type for unsuccessful Requests
- If any read associated with a locked sequence is completed unsuccessfully, the Requester must assume that the atomicity of the lock is no longer assured, and that the path between the Requester and Completer is no longer locked
- All writes for the locked sequence use MWr Requests
- The Unlock Message is used to indicate the end of a locked sequence
  - A Switch propagates Unlock Messages to the locked Egress Port

- Upon receiving an Unlock Message, a Legacy Endpoint or Bridge must unlock itself if it is in a locked state
  - If not locked, or if the Receiver is a PCI Express Endpoint or Bridge which does not support lock, the Unlock Message is ignored and discarded

6.5.3 Switches and Lock - Rules

Switches must distinguish transactions associated with locked sequences from other transactions to prevent other transactions from interfering with the lock and potentially causing deadlock. The following rules cover how this is done. Note that locked accesses are limited to TC0, which is always mapped to VC0.

- When a Switch propagates a MRdLk Request from the Ingress Port (closest to the Root Complex) to the Egress Port, it must block all Requests which map to the default Virtual Channel (VC0) from being propagated to the Egress Port
  - If a subsequent MRdLk Request is Received at this Ingress Port addressing a different Egress Port, the behavior of the Switch is undefined

  Note: This sort of split-lock access is not supported by PCI Express and software must not cause such a locked access. System deadlock may result from such accesses.
- When the CplDLk for the first MRdLk Request is returned, if the Completion indicates a Successful Completion status, the Switch must block all Requests from all other Ports from being propagated to either of the Ports involved in the locked access, except for Requests which map to non-VC0 on the Egress Port
- The two Ports involved in the locked sequence must remain blocked as described above until the Switch receives the Unlock Message (at the Ingress Port for the initial MRdLk Request)
  - The Unlock Message must be forwarded to the locked Egress Port
  - The Unlock Message may be broadcast to all other Ports
  - The Ingress Port is unblocked once the Unlock Message arrives, and the Egress Port(s) which were blocked are unblocked following the Transmission of the Unlock Message out of the Egress Ports
    - Ports which were not involved in the locked access are unaffected by the Unlock Message

6.5.4 PCI Express/PCI Bridges and Lock - Rules

The requirements for PCI Express/PCI Bridges are similar to those for Switches, except that, because PCI Express/PCI Bridges use only the default Virtual Channel and Traffic Class, all other traffic is blocked during the locked access. The requirements on the PCI bus side of the PCI Express/PCI Bridge match the requirements for a PCI/PCI Bridge (see the *PCI-to-PCI Bridge Architecture Specification, Revision 1.2* and the *PCI Express to PCI/PCI-X Bridge Specification, Revision 1.0*).

6.5.5 Root Complex and Lock - Rules

A Root Complex is permitted to support locked transactions as a Requester. If locked transactions are supported, a Root Complex must follow the sequence described in Section 6.5.2 to perform a locked access. The mechanisms used by the Root Complex to interface PCI Express to the Host CPU(s) are outside the scope of this document.

6.5.6 Legacy Endpoints

Legacy Endpoints are permitted to support locked accesses, although their use is discouraged. If locked accesses are supported, Legacy Endpoints must handle them as follows:

- The Legacy Endpoint becomes locked when it Transmits the first Completion for the first Read Request of the locked access with a Successful Completion status
  - If the completion status is not Successful Completion, the Legacy Endpoint does not become locked
  - Once locked, the Legacy Endpoint must remain locked until it receives the Unlock Message
- While locked, a Legacy Endpoint must not issue any Requests using TCs which map to the default Virtual Channel (VC0)

  Note that this requirement applies to all possible sources of Requests within the Endpoint, in the case where there is more than one possible source of Requests.
  - Requests may be issued using TCs which map to VCs other than the default Virtual Channel

6.5.7 PCI Express Endpoints

PCI Express Endpoints do not support lock. A PCI Express Endpoint must treat a MRdLk Request as an Unsupported Request (see Chapter 2).

6.6 PCI Express Reset - Rules

This section specifies the PCI Express Reset mechanisms. This section covers the relationship between the architectural mechanisms defined in this document and the reset mechanisms defined in this document. Any relationship between the PCI Express Conventional Reset and component or platform reset is component or platform specific (except as explicitly noted).

6.6.1 Conventional Reset

Conventional Reset includes all reset mechanisms other than Function Level Reset. There are two categories of Conventional Resets: Fundamental Reset and resets that are not Fundamental Reset. This section applies to all types of Conventional Reset.

In all form factors and system hardware configurations, there must, at some level, be a hardware mechanism for setting or returning all Port states to the initial conditions specified in this document

– this mechanism is called "Fundamental Reset". This mechanism can take the form of an auxiliary signal provided by the system to a component or adapter card, in which case the signal must be called PERST#, and must conform to the rules specified in Section 4.2.4.8.1. When PERST# is provided to a component or adapter, this signal must be used by the component or adapter as Fundamental Reset. When PERST# is not provided to a component or adapter, Fundamental Reset is generated autonomously by the component or adapter, and the details of how this is done are outside the scope of this document. If a Fundamental Reset is generated autonomously by the component or adapter, and if power is supplied by the platform to the component/adapter, the component/adapter must generate a Fundamental Reset to itself if the supplied power goes outside of the limits specified for the form factor or system.

- There are three distinct types of Conventional Reset: cold, warm, and hot:
  - A Fundamental Reset must occur following the application of power to the component. This is called a cold reset.
  - In some cases, it may be possible for the Fundamental Reset mechanism to be triggered by hardware without the removal and re-application of power to the component. This is called a warm reset. This document does not specify a means for generating a warm reset.
  - There is an in-band mechanism for propagating Conventional Reset across a Link. This is called a hot reset and is described in Section 4.2.4.8.2.

    There is an in-band mechanism for software to force a Link into Electrical Idle, "disabling" the Link. The Disabled LTSSM state is described in Section 4.2.5.9, the Link Disable control bit is described in Section 7.5.3.7, and the Downstream Port Containment mechanism is described in Section 6.2.10. Disabling a Link causes Downstream components to undergo a hot reset.

    Note also that the Data Link Layer reporting DL_Down status is in some ways identical to a hot reset – see Section 2.9.
- On exit from any type of Conventional Reset (cold, warm, or hot), all Port registers and state machines must be set to their initialization values as specified in this document, except for sticky registers (see Section 7.4).
  - Note that, from a device point of view, any type of Conventional Reset (cold, warm, hot, or DL_Down) has the same effect at the Transaction Layer and above as would RST# assertion and deassertion in conventional PCI.
- On exit from a Fundamental Reset, the Physical Layer will attempt to bring up the Link (see Section 4.2.5). Once both components on a Link have entered the initial Link Training state, they will proceed through Link initialization for the Physical Layer and then through Flow Control initialization for VC0, making the Data Link and Transaction Layers ready to use the Link.
  - Following Flow Control initialization for VC0, it is possible for TLPs and DLLPs to be transferred across the Link.

Following a Conventional Reset, some devices may require additional time before they are able to respond to Requests they receive. Particularly for Configuration Requests it is necessary that components and devices behave in a deterministic way, which the following rules address.

The first set of rules addresses requirements for components and devices:

- A component must enter the LTSSM Detect state within 20 ms of the end of Fundamental Reset (Link Training is described in Section 4.2.4).
  - Note: In some systems, it is possible that the two components on a Link may exit Fundamental Reset at different times. Each component must observe the requirement to enter the initial active Link Training state within 20 ms of the end of Fundamental Reset from its own point of view.
- On the completion of Link Training (entering the DL_Active state, see Section 3.2), a component must be able to receive and process TLPs and DLLPs.

The second set of rules addresses requirements placed on the system:

- To allow components to perform internal initialization, system software must wait a specified minimum period following the end of a Conventional Reset of one or more devices before it is permitted to issue Configuration Requests to those devices, unless Readiness Notifications mechanisms are used (see Section 6.23.).
  - With a Downstream Port that does not support Link speeds greater than 5.0 GT/s, software must wait a minimum of 100 ms before sending a Configuration Request to the device immediately below that Port.
  - With a Downstream Port that supports Link speeds greater than 5.0 GT/s, software must wait a minimum of 100 ms after Link training completes before sending a Configuration Request to the device immediately below that Port. Software can determine when Link training completes by polling the Data Link Layer Link Active bit or by setting up an associated interrupt (see Section 6.7.3.3).
  - A system must guarantee that all components intended to be software visible at boot time are ready to receive Configuration Requests within the applicable minimum period based on the end of Conventional Reset at the Root Complex – how this is done is beyond the scope of this specification.
  - Note: Software should use 100 ms wait periods only if software enables CRS Software Visibility. Otherwise, Completion timeouts, platform timeouts, or lengthy processor instruction stalls may result. See the Configuration Request Retry Status Implementation Note in Section 2.3.1.
- Unless Readiness Notifications mechanisms are used (see Section 6.23), the Root Complex and/or system software must allow at least 1.0 s after a Conventional Reset of a device, before it may determine that a device which fails to return a Successful Completion status for a valid Configuration Request is a broken device. This period is independent of how quickly Link training completes.

  Note: This delay is analogous to the T_{rhfa} parameter specified for PCI/PCI-X, and is intended to allow an adequate amount of time for devices which require self initialization.
- When attempting a Configuration access to devices on a PCI or PCI-X bus segment behind a PCI Express/PCI(-X) Bridge, the timing parameter T_{rhfa} must be respected.

For this second set of rules, if system software does not have direct visibility into the state of Fundamental Reset (e.g., Hot-Plug; see Section 6.7), software must base these timing parameters on an event known to occur after the end of Fundamental Reset.

When a Link is in normal operation, the following rules apply:

- If, for whatever reason, a normally operating Link goes down, the Transaction and Data Link Layers will enter the DL_Inactive state (see Sections 2.9 and 3.2.1).
- For any Root or Switch Downstream Port, setting the Secondary Bus Reset bit of the Bridge Control register associated with the Port must cause a hot reset to be sent (see Section 4.2.4.8.2).
- For a Switch, the following must cause a hot reset to be sent on all Downstream Ports:
  - Setting the Secondary Bus Reset bit of the Bridge Control register associated with the Upstream Port
  - The Data Link Layer of the Upstream Port reporting DL_Down status. In Switches that support Link speeds greater than 5.0 GT/s, the Upstream Port must direct the LTSSM of each Downstream Port to the Hot Reset state, but not hold the LTSSMs in that state. This permits each Downstream Port to begin Link training immediately after its hot reset completes. This behavior is recommended for all Switches.
  - Receiving a hot reset on the Upstream Port

Certain aspects of Fundamental Reset are specified in this document and others are specific to a platform, form factor and/or implementation. Specific platforms, form factors or application spaces may require the additional specification of the timing and/or sequencing relationships between the components of the system for Fundamental Reset. For example, it might be required that all PCI Express components within a chassis observe the assertion and deassertion of Fundamental Reset at the same time (to within some tolerance). In a multi-chassis environment, it might be necessary to specify that the chassis containing the Root Complex be the last to exit Fundamental Reset.

In all cases where power and PERST# are supplied, the following parameters must be defined:

- T_{pvperl} – PERST# must remain active at least this long after power becomes valid
- T_{perst} – When asserted, PERST# must remain asserted at least this long
- T_{fail} – When power becomes invalid, PERST# must be asserted within this time

Additional parameters may be specified.

In all cases where a reference clock is supplied, the following parameter must be defined:

- $T_{perst\text{-}clk}$ – PERST# must remain active at least this long after any supplied reference clock is stable

Additional parameters may be specified.

6.6.2 Function Level Reset (FLR)

The FLR mechanism enables software to quiesce and reset Endpoint hardware with Function-level granularity. Three example usage models illustrate the benefits of this feature:

- In some systems, it is possible that the software entity that controls a Function will cease to operate normally. To prevent data corruption, it is necessary to stop all PCI Express and

external I/O (not PCI Express) operations being performed by the Function. Other defined reset operations do not guarantee that external I/O operations will be stopped.

- In a partitioned environment where hardware is migrated from one partition to another, it is necessary to ensure that no residual "knowledge" of the prior partition be retained by hardware, for example, a user's secret information entrusted to the first partition but not to the second. Further, due to the wide range of Functions, it is necessary that this be done in a Function-independent way.
- When system software is taking down the software stack for a Function and then rebuilding that stack, it is sometimes necessary to return the state to an uninitialized state before rebuilding the Function's software stack.

Implementation of FLR is optional (not required), but is strongly recommended.

FLR applies on a per Function basis. Only the targeted Function is affected by the FLR operation. The Link state must not be affected by an FLR.

FLR modifies the Function state described by this specification as follows:

- Function registers and Function-specific state machines must be set to their initialization values as specified in this document, except for the following:
  - sticky-type registers (ROS, RWS, RW1CS)
  - registers defined as type HwInit
  - these other fields or registers:
    - Captured Slot Power Limit Value in the Device Capabilities register
    - Captured Slot Power Limit Scale in the Device Capabilities register
    - Max_Payload_Size in the Device Control register
    - Active State Power Management (ASPM) Control in the Link Control register
    - Read Completion Boundary (RCB) in the Link Control register
    - Common Clock Configuration in the Link Control register
    - Extended Synch in the Link Control register
    - Enable Clock Power Management in the Link Control register
    - Hardware Autonomous Width Disable in Link Control register
    - Hardware Autonomous Speed Disable in the Link Control 2 register
    - Link Equalization 8.0 GT/s Request in the Link Status 2 register
    - Link Equalization Request 16.0 GT/s in the 16.0 GT/s Status register
    - Lane Equalization Control register in the Secondary PCI Express Extended Capability structure
    - 16.0 GT/s Lane Equalization Control register in the Physical Layer 16.0 GT/s Extended Capability structure
    - All registers in the Virtual Channel Capability structure

- All registers in the Multi-Function Virtual Channel Capability structure
- All registers in the Data Link Feature Extended Capability structure
- All registers in the Physical Layer 16.0 GT/s Extended Capability structure
- All registers in the Lane Margining at the Receiver Extended Capability structure

Note that the controls that enable the Function to initiate requests on PCI Express are cleared, including Bus Master Enable, MSI Enable, and the like, effectively causing the Function to become quiescent on the Link.

Note that Port state machines associated with Link functionality including those in the Physical and Data Link Layers are not reset by FLR, and VC0 remains initialized following an FLR.

- ❑ Any outstanding INTx interrupt asserted by the Function must be deasserted by sending the corresponding Deassert_INTx Message prior to starting the FLR.

Note that when the FLR is initiated to a Function of a Multi-Function Device, if another Function continues to assert a matching INTx, no Deassert_INTx Message will be transmitted.

After an FLR has been initiated by writing a 1b to the Initiate Function Level Reset bit, the Function must complete the FLR within 100 ms. If software initiates an FLR when the Transactions Pending bit is 1b, then software must not initialize the Function until allowing adequate time for any associated Completions to arrive, or to achieve reasonable certainty that any remaining Completions will never arrive. For this purpose, it is recommended that software allow as much time as provided by the pre-FLR value for Completion Timeout on the device. If Completion Timeouts were disabled on the Function when FLR was issued, then the delay is system dependent but must be no less than 100 ms. If Function Readiness Status (FRS – see Section 6.23.2) is implemented, then system software is permitted to issue Configuration Requests to the Function immediately following receipt of an FRS Message indicating Configuration-Ready, however, this does not necessarily indicate that outstanding Requests initiated by the Function have completed.

Note that upon receipt of an FLR, a device Function may either clear all transaction status including Transactions Pending or set the Completion Timeout to its default value so that all pending transactions will time out during FLR execution. Regardless, the Transactions Pending bit must be clear upon completion of the FLR.

Since FLR modifies Function state not described by this specification (in addition to state that is described by this specification), it is necessary to specify the behavior of FLR using a set of criteria that, when applied to the Function, show that the Function has satisfied the requirements of FLR. The following criteria must be applied using Function-specific knowledge to evaluate the Function's behavior in response to an FLR:

- ❑ The Function must not give the appearance of an initialized adapter with an active host on any external interfaces controlled by that Function. The steps needed to terminate activity on external interfaces are outside of the scope of this specification.
    - For example, a network adapter must not respond to queries that would require adapter initialization by the host system or interaction with an active host system, but is permitted to perform actions that it is designed to perform without requiring host initialization or interaction. If the network adapter includes multiple Functions that operate on the same external network interface, this rule affects only those aspects associated with the particular Function reset by FLR.

- The Function must not retain within itself software readable state that potentially includes secret information associated with any preceding use of the Function. Main host memory assigned to the Function must not be modified by the Function.
    - For example, a Function with internal memory readable directly or indirectly by host software must clear or randomize that memory.
- The Function must return to a state such that normal configuration of the Function's PCI Express interface will cause it to be useable by drivers normally associated with the Function

When an FLR is initiated, the targeted Function must behave as follows:

- The Function must return the Completion for the configuration write that initiated the FLR operation and then initiate the FLR.
- While an FLR is in progress:
    - If a Request arrives, the Request is permitted to be silently discarded (following update of flow control credits) without logging or signaling it as an error.
    - If a Completion arrives, 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.
    - While a Function is required to complete the FLR operation within the time limit described above, the subsequent Function-specific initialization sequence may require additional time. If additional time is required, the Function must return a Configuration Request Retry Status (CRS) Completion Status when a Configuration Request is received after the time limit above. After the Function responds to a Configuration Request with a Completion status other than CRS, it is not permitted to return CRS until it is reset again.

IMPLEMENTATION NOTE

Avoiding Data Corruption From Stale Completions

An FLR causes a Function to lose track of any outstanding non-posted Requests. Any corresponding Completions that later arrive are referred to as being "stale". If software issues an FLR while there are outstanding Requests, and then re-enables the Function for operation without waiting for potential stale Completions, any stale Completions that arrive afterwards may cause data corruption by being mistaken by the Function as belonging to Requests issued since the FLR.

Software can avoid data corruption from stale Completions in a variety of ways. Here's a possible algorithm:

1. Software that's performing the FLR synchronizes with other software that might potentially access the Function directly, and ensures such accesses do not occur during this algorithm.
2. Software clears the entire Command register, disabling the Function from issuing any new Requests.
3. Software polls the Transactions Pending bit in the Device Status register either until it is clear or until it has been long enough that software is reasonably certain that Completions associated with any remaining outstanding Transactions will never arrive. On many platforms, the Transactions Pending bit will usually clear within a few milliseconds, so software might choose to poll during this initial period using a tight software loop. On rare cases when the Transactions Pending bit does not clear by this time, software will need to poll for a much longer platform-specific period (potentially seconds), so software might choose to conduct this polling using a timer-based interrupt polling mechanism.
4. Software initiates the FLR.
5. Software waits 100 ms.
6. Software reconfigures the Function and enables it for normal operation.

6.7 PCI Express Hot-Plug Support

The PCI Express architecture is designed to natively support both hot-add and hot-removal ("hot-plug") of cables, add-in cards, and modules. PCI Express hot-plug support provides a "toolbox" of mechanisms that allow different user/operator models to be supported using a self-consistent infrastructure. These mechanisms may be used to implement orderly addition/removal that relies on coordination with the operating system (e.g., traditional PCI hot-plug), as well as async removal which proceeds without lock-step synchronization with the operating system. This section defines the set of hot-plug mechanisms and specifies how the elements of hot-plug, such as indicators and push buttons, must behave if implemented in a system.

6.7.1 Elements of Hot-Plug

Table 6-6 lists the physical elements comprehended in this specification for support of hot-plug models. A form factor specification must define how these elements are used in that form factor. For a given form factor specification, it is possible that only some of the available hot-plug elements

are required, or even that none of these elements are required. In all cases, the form factor specification must define all assumptions and limitations placed on the system or the user by the choice of elements included. Silicon component implementations that are intended to be used only with selected form factors are permitted to support only those elements that are required by the associated form factor(s).

Table 6-6: Elements of Hot-Plug

Element	Purpose
Indicators	Show the power and attention state of the slot
Manually-operated Retention Latch (MRL)	Holds adapter in place
MRL Sensor	Allows the Port and system software to detect the MRL being opened
Electromechanical Interlock	Prevents removal of adapter from slot
Attention Button	Allows user to request hot-plug operations
Software User Interface	Allows user to request hot-plug operations
Slot Numbering	Provides visual identification of slots
Power Controller	Software-controlled electronic component or components that control power to a slot or adapter and monitor that power for fault conditions
Out-of-band Presence Detect	Method of determining physical presence of an adapter in a slot that does not rely on the Physical Layer

6.7.1.1 *Indicators*

Two indicators are defined: the Power Indicator and the Attention Indicator. Each indicator is in one of three states: on, off, or blinking. Hot-plug system software has exclusive control of the indicator states by writing the command registers associated with the indicator (with one exception noted below). The indicator requirements must be included in all form factor specifications. For a given form factor, the indicators may be required or optional or not applicable at all.

The hot-plug capable Port controls blink frequency, duty cycle, and phase of the indicators. Blinking indicators must operate at a frequency of between 1 and 2 Hz, with a 50% (+/- 5%) duty cycle. Blinking indicators are not required to be synchronous or in-phase between Ports.

Indicators may be physically located on the chassis or on the adapter (see the associated form factor specification for Indicator location requirements). Regardless of the physical location, logical control of the indicators is by the Downstream Port of the Upstream component on the Link.

The Downstream Port must not change the state of an indicator unless commanded to do so by software, except for platforms capable of detecting stuck-on power faults (relevant only when a power controller is implemented). In the case of a stuck-on power fault, the platform is permitted to override the Downstream Port and force the Power Indicator to be on (as an indication that the adapter should not be removed). The handling by system software of stuck-on faults is optional and not described in this specification. Therefore, the platform vendor must ensure that this feature, if implemented, is addressed via other software, platform documentation, or by other means.

6.7.1.1.1 Attention Indicator

The Attention Indicator, which must be yellow or amber in color, indicates that an operational problem exists or that the hot-plug slot is being identified so that a human operator can locate it easily.

Table 6-7: Attention Indicator States

Indicator Appearance	Meaning
Off	Normal - Normal operation
On	Attention - Operational problem at this slot
Blinking	Locate - Slot is being identified at the user's request

Attention Indicator Off

The Attention Indicator in the Off state indicates that neither the adapter (if one is present) nor the hot-plug slot requires attention.

Attention Indicator On

The Attention Indicator in the On state indicates that an operational problem exists at the adapter or slot.

An operational problem is a condition that prevents continued operation of an adapter. The operating system or other system software determines whether a specific condition prevents continued operation of an adapter and whether lighting the Attention Indicator is appropriate. Examples of operational problems include problems related to external cabling, adapter, software

drivers, and power faults. In general, the Attention Indicator in the On state indicates that an operation was attempted and failed or that an unexpected event occurred.

The Attention Indicator is not used to report problems detected while validating the request for a hot-plug operation. Validation is a term applied to any check that system software performs to assure that the requested operation is viable, permitted, and will not cause problems. Examples of validation failures include denial of permission to perform a hot-plug operation, insufficient power budget, and other conditions that may be detected before a hot-plug request is accepted.

Attention Indicator Blinking

A blinking Attention Indicator indicates that system software is identifying this slot for a human operator to find. This behavior is controlled by a user (for example, from a software user interface or management tool).

6.7.1.1.2 Power Indicator

The Power Indicator, which must be green in color, indicates the power state of the slot. Table 6-8 lists the Power Indicator states.

Table 6-8: Power Indicator States

Indicator Appearance	Meaning
Off	Power Off - Insertion or removal of the adapter is permitted.
On	Power On - Insertion or removal of the adapter is not permitted.
Blinking	Power Transition - Hot-plug operation is in progress and insertion or removal of the adapter is not permitted.

Power Indicator Off

The Power Indicator in the Off state indicates that insertion or removal of the adapter is permitted. Main power to the slot is off if required by the form factor. Note that, depending on the form factor, other power/signals may remain on, even when main power is off and the Power Indicator is off. In an example using the *PCI Express Card Electromechanical Specification* form factor, if the platform provides Vaux to hot-plug slots and the MRL is closed, any signals switched by the MRL are connected to the slot even when the Power Indicator is off. Signals switched by the MRL are disconnected when the MRL is opened. System software must cause a slot's Power Indicator to be turned off when the slot is not powered and/or it is permissible to insert or remove an adapter. Refer to the appropriate form factor specification for details.

Power Indicator On

The Power Indicator in the On state indicates that the hot-plug operation is complete and that main power to the slot is On and that insertion or removal of the adapter is not permitted.

Power Indicator Blinking

A blinking Power Indicator indicates that the slot is powering up or powering down and that insertion or removal of the adapter is not permitted.

The blinking Power Indicator also provides visual feedback to the operator when the Attention Button is pressed or when hot-plug operation is initiated through the hot-plug software interface.

6.7.1.2 *Manually-operated Retention Latch (MRL)*

An MRL is a manually-operated retention mechanism that holds an adapter in the slot and prevents the user from removing the device. The MRL rigidly holds the adapter in the slot so that cables may be attached without the risk of creating intermittent contact. MRLs that hold down two or more adapters simultaneously are permitted in platforms that do not provide MRL Sensors.

6.7.1.3 *MRL Sensor*

The MRL Sensor is a switch, optical device, or other type of sensor that reports the position of a slot's MRL to the Downstream Port. The MRL Sensor reports closed when the MRL is fully closed and open at all other times (that is, if the MRL fully open or in an intermediate position).

If a power controller is implemented for the slot, the slot main power must be automatically removed from the slot when the MRL Sensor indicates that the MRL is open. If signals such as Vaux and SMBus are switched by the MRL, then these signals must be automatically removed from the slot when the MRL Sensor indicates that the MRL is open and must be restored to the slot when the MRL Sensor indicates that MRL has closed again. Refer to the appropriate form factor specification to identify the signals, if any, switched by the MRL.

Note that the Hot-Plug Controller does not autonomously change the state of either the Power Indicator or the Attention Indicator based on MRL sensor changes.

IMPLEMENTATION NOTE

MRL Sensor Handling

In the absence of an MRL sensor, for some form factors, out-of-band presence detect may be used to handle the switched signals. In this case, when out-of-band presence detect indicates the absence of an adapter in a slot, the switched signals will be automatically removed from the slot.

If an MRL Sensor is implemented without a corresponding MRL Sensor input on the Hot-Plug Controller, it is recommended that the MRL Sensor be routed to power fault input of the Hot-Plug Controller. This allows an active adapter to be powered off when the MRL is opened.

6.7.1.4 *Electromechanical Interlock*

An electromechanical interlock is a mechanism for physically locking the adapter or MRL in place until system software releases it. The state of the electromechanical interlock is set by software and must not change except in response to a subsequent software command. In particular, the state of the electromechanical interlock must be maintained even when power to the hot-plug slot is removed.

The current state of the electromechanical interlock must be reflected at all times in the Electromechanical Interlock Status bit in the Slot Status register, which must be updated within 200 ms of any commanded change. Software must wait at least 1 second after issuing a command to toggle the state of the Electromechanical Interlock before another command to toggle the state can be issued. Systems may optionally expand control of interlocks to provide physical security of the adapter.

6.7.1.5 *Attention Button*

The Attention Button is a momentary-contact push button switch, located adjacent to each hot-plug slot or on the adapter that is pressed by the user to initiate a hot-plug operation at that slot. Regardless of the physical location of the button, the signal is processed and indicated to software by hot-plug hardware associated with the Downstream Port corresponding to the slot.

The Attention Button must allow the user to initiate both hot add and hot remove operations regardless of the physical location of the button.

If present, the Power Indicator provides visual feedback to the human operator (if the system software accepts the request initiated by the Attention Button) by blinking. Once the Power Indicator begins blinking, a 5-second abort interval exists during which a second depression of the Attention Button cancels the operation.

If an operation initiated by an Attention Button fails for any reason, it is recommended that system software present an error message explaining the failure via a software user interface or add the error message to a system log.

6.7.1.6 *Software User Interface*

System software provides a user interface that allows hot insertions and hot removals to be initiated and that allows occupied slots to be monitored. A detailed discussion of hot-plug user interfaces is operating system specific and is therefore beyond the scope of this document.

On systems with multiple hot-plug slots, the system software must allow the user to initiate operations at each slot independent of the states of all other slots. Therefore, the user is permitted to initiate a hot-plug operation on one slot using either the software user interface or the Attention Button while a hot-plug operation on another slot is in process, regardless of which interface was used to start the first operation.

6.7.1.7 *Slot Numbering*

A Physical Slot Identifier (as defined in *PCI Hot-Plug Specification, Revision 1.1*, Section 1.5) consists of an optional chassis number and the physical slot number of the slot. The physical slot number is a

chassis unique identifier for a slot. System software determines the physical slot number from registers in the Port. Chassis number 0 is reserved for the main chassis. The chassis number for other chassis must be a non-zero value obtained from a PCI-to-PCI Bridge's Chassis Number register (see the *PCI-to-PCI Bridge Architecture Specification, Revision 1.2*, Section 13.4).

Regardless of the form factor associated with each slot, each physical slot number must be unique within a chassis.

6.7.1.8 *Power Controller*

The power controller is an element composed of one or more discrete components that acts under control of software to set the power state of the hot-plug slot as appropriate for the specific form factor. The power controller must also monitor the slot for power fault conditions (as defined in the associated form factor specification) that occur on the slot's main power rails and, if supported, auxiliary power rail.

If a power controller is not present, the power state of the hot-plug slot must be set automatically by the hot-plug controller in response to changes in the presence of an adapter in the slot.

The power controller monitors main and auxiliary power faults independently. If a power controller detects a main power fault on the hot-plug slot, it must automatically set its internal main power fault latch and remove main power from the hot-plug slot (without affecting auxiliary power). Similarly, if a power controller detects an auxiliary power fault on the hot-plug slot, it must automatically set its internal auxiliary power fault latch and remove auxiliary power from the hot-plug slot (without affecting main power). Power must remain off to the slot as long as the power fault condition remains latched, regardless of any writes by software to turn on power to the hot-plug slot. The main power fault latch is cleared when software turns off power to the hot-plug slot. The mechanism by which the auxiliary power fault latch is cleared is form factor specific but generally requires auxiliary power to be removed from the hot-plug slot. For example, one form factor may remove auxiliary power when the MRL for the slot is opened while another may require the adapter to be physically removed from the slot. Refer to the associated form factor specifications for specific requirements.

Since the Power Controller Control bit in the Slot Control register reflects the last value written and not the actual state of the power controller, this means there may be an inconsistency between the value of the Power Controller Control bit and the state of the power to the slot in a power fault condition. To determine whether slot is off due to a power fault, software must use the power fault software notification to detect power faults. To determine that a requested power-up operation has otherwise failed, software must use the hot-plug slot power-up time out mechanism described in Section 6.7.3.3.

Software must not assume that writing to the Slot Control register to change the power state of a hot-plug slot causes an immediate power state transition. After turning power on, software must wait for a Data Link Layer State Changed event, as described in Section 6.7.3.3. After turning power off, software must wait for at least 1 second before taking any action that relies on power having been removed from the hot-plug slot. For example, software is not permitted to turn off the power indicator (if present) or attempt to turn on the power controller before completing the 1 second wait period.

6.7.2 Registers Grouped by Hot-Plug Element Association

The registers described in this section are grouped by hot-plug element to convey all registers associated with implementing each element. Register fields associated with each Downstream Port implementing a hot-plug capable slot are located in the PCI Express Capabilities, Slot Capabilities, Slot Control, and Slot Status registers in the PCI Express Capability structure (see Section 7.5.2). Registers reporting the presence of hot-plug elements associated with the device Function on an adapter are located in the Device Capabilities register (also in the PCI Express Capability structure).

6.7.2.1 *Attention Button Registers*

Attention Button Present (Slot Capabilities and Device Capabilities) – This bit indicates if an Attention Button is electrically controlled by the chassis (Slot Capabilities) or by the adapter (Device Capabilities).

Attention Button Pressed (Slot Status) – This bit is set when an Attention Button electrically controlled by the chassis is pressed.

Attention Button Pressed Enable (Slot Control) – When Set, this bit enables software notification on an Attention Button Pressed event (see Section 6.7.3.4).

6.7.2.2 *Attention Indicator Registers*

Attention Indicator Present (Slot Capabilities and Device Capabilities) – This bit indicates if an Attention Indicator is electrically controlled by the chassis (Slot Capabilities) or by the adapter (Device Capabilities).

Attention Indicator Control (Slot Control) – When written, sets an Attention Indicator electrically controlled by the chassis to the written state.

6.7.2.3 *Power Indicator Registers*

Power Indicator Present (Slot Capabilities and Device Capabilities) – This bit indicates if a Power Indicator is electrically controlled by the chassis (Slot Capabilities) or by the adapter (Device Capabilities).

Power Indicator Control (Slot Control) – When written, sets a Power Indicator electrically controlled by the chassis to the written state.

6.7.2.4 *Power Controller Registers*

Power Controller Present (Slot Capabilities) – This bit indicates if a Power Controller is implemented.

Power Controller Control (Slot Control) – Turns the Power Controller on or off according to the value written.

Power Fault Detected (Slot Status) – This bit is set when a power fault is detected at the slot or the adapter.

Power Fault Detected Enable (Slot Control) – When Set, this bit enables software notification on a power fault event (see Section 6.7.3.4).

6.7.2.5 *Presence Detect Registers*

Presence Detect State (Slot Status) – This bit indicates the presence of an adapter in the slot.

Presence Detect Changed (Slot Status) – This bit is set when a presence detect state change is detected.

Presence Detect Changed Enable (Slot Control) – When Set, this bit enables software notification on a presence detect changed event (see Section 6.7.3.4).

6.7.2.6 *MRL Sensor Registers*

MRL Sensor Present (Slot Capabilities) – This bit indicates if an MRL Sensor is implemented.

MRL Sensor Changed (Slot Status) – This bit is set when the value of the MRL Sensor state changes.

MRL Sensor Changed Enable (Slot Control) – When Set, this bit enables software notification on a MRL Sensor changed event (see Section 6.7.3.4).

MRL Sensor State (Slot Status) – This register reports the status of the MRL Sensor if one is implemented.

6.7.2.7 *Electromechanical Interlock Registers*

Electromechanical Interlock Present (Slot Capabilities) – This bit indicates if an Electromechanical Interlock is implemented.

Electromechanical Interlock Status (Slot Status) – This bit reflects the current state of the Electromechanical Interlock.

Electromechanical Interlock Control (Slot Control) – This bit when set to 1b toggles the state of the Electromechanical Interlock.

6.7.2.8 *Command Completed Registers*

No Command Completed Support (Slot Capabilities) – This bit when set to 1b indicates that this slot does not generate software notification when an issued command is completed by the Hot-Plug Controller.

Command Completed (Slot Status) – This bit is set when the Hot-Plug Controller completes an issued command and is ready to accept the next command.

Command Completed Interrupt Enable (Slot Control) – When Set, this bit enables software notification (see Section 6.7.3.4) when a command is completed by the hot-plug control logic.

6.7.2.9 *Port Capabilities and Slot Information Registers*

Slot Implemented (PCI Express Capabilities) – When Set, this bit indicates that the Link associated with this Downstream Port is connected to a slot.

Physical Slot Number (Slot Capabilities) – This hardware initialized field indicates the physical slot number attached to the Port.

Hot-Plug Capable (Slot Capabilities) – When Set, this bit indicates this slot is capable of supporting hot-plug.

Hot-Plug Surprise (Slot Capabilities) – When Set, this bit indicates that adapter removal from the system without any prior notification is permitted for the associated form factor.

6.7.2.10 *Hot-Plug Interrupt Control Register*

Hot-Plug Interrupt Enable (Slot Control) – When Set, this bit enables generation of the hot-plug interrupt on enabled hot-plug events.

6.7.3 PCI Express Hot-Plug Events

A Downstream Port with hot-plug capabilities supports the following hot-plug events:

- Slot Events:
  - Attention Button Pressed
  - Power Fault Detected
  - MRL Sensor Changed
  - Presence Detect Changed
- Command Completed Events
- Data Link Layer State Changed Events

Each of these events has a status field, which indicates that an event has occurred but has not yet been processed by software, and an enable field, which indicates whether the event is enabled for software notification. Some events also have a capability field, which indicates whether the event type is supported on the Port. The grouping of these fields by event type is listed in Section 6.7.2, and each individual field is described in Section 7.5.2.

6.7.3.1 *Slot Events*

A Downstream Port with hot-plug capabilities monitors the slot it controls for the slot events listed above. When one of these slot events is detected, the Port indicates that the event has occurred by setting the status field associated with the event. At that point, the event is pending until software clears the status field.

Once a slot event is pending on a particular slot, all subsequent events of that type are ignored on that slot until the event is cleared. The Port must continue to monitor the slot for all other slot event types and report them as they occur.

If enabled through the associated enable field, slot events must generate a software notification. If the event is not supported on the Port as indicated by the associated capability field, software must not enable software notification for the event. The mechanism by which this notification is reported to software is described in Section 6.7.3.4.

6.7.3.2 *Command Completed Events*

Since changing the state of some hot-plug elements may not happen instantaneously, PCI Express supports hot-plug commands and command completed events. All hot-plug capable Ports are required to support hot-plug commands and, if the capability is reported, command completed events.

Software issues a command to a hot-plug capable Downstream Port by issuing a write transaction that targets any portion of the Port's Slot Control register. A single write to the Slot Control register is considered to be a single command, even if the write affects more than one field in the Slot Control register. In response to this transaction, the Port must carry out the requested actions and then set the associated status field for the command completed event. The Port must process the command normally even if the status field is already set when the command is issued. If a single command results in more than one action being initiated, the order in which the actions are executed is unspecified. All actions associated with a single command execution must not take longer than 1 second.

If command completed events are not supported as indicated by a value of 1b in the No Command Completed Support field of the Slot Capabilities register, a hot-plug capable Port must process a write transaction that targets any portion of the Port's Slot Control register without any dependency on previous Slot Control writes. Software is permitted to issue multiple Slot Control writes in sequence without any delay between the writes.

If command completed events are supported, then software must wait for a command to complete before issuing the next command. However, if the status field is not set after the 1 second limit on command execution, software is permitted to repeat the command or to issue the next command. If software issues a write before the Port has completed processing of the previous command and before the 1 second time limit has expired, the Port is permitted to either accept or discard the write. Such a write is considered a programming error, and could result in a discrepancy between the Slot Control register and the hot plug element state. To recover from such a programming error and return the controller to a consistent state, software must issue a write to the Slot Control register which conforms to the command completion rules.

If enabled through the associated enable field, the completion of a commands must generate a software notification. The exception to this rule is a command that occurs as a result of a write to the Slot Control register that disables software notification of command completed events. Such a command must be processed as described above, but must not generate a software notification.

6.7.3.3 *Data Link Layer State Changed Events*

The Data Link Layer State Changed event provides an indication that the state of the Data Link Layer Link Active bit in the Link Status register has changed. Support for Data Link Layer State Changed events and software notification of these events are required for hot-plug capable Downstream Ports. If this event is supported, the Port sets the status field associated with the event when the value in the Data Link Layer Link Active bit changes.

This event allows software to indirectly determine when power has been applied to a newly hot-plugged adapter. Software must wait for 100 ms after the Data Link Layer Link Active bit reads 1b before initiating a configuration access to the hot added device (see Section 6.6). Software must allow 1 second after the Data Link Layer Link Active bit reads 1b before it is permitted to determine that a hot plugged device which fails to return a Successful Completion for a Valid Configuration Request is a broken device (see Section 6.6).

The Data Link Layer State Changed event must occur within 1 second of the event that initiates the hot-insertion. If a power controller is supported, the time out interval is measured from when software initiated a write to the Slot Control register to turn on the power. If a power controller is not supported, the time out interval is measured from presence detect slot event. Software is allowed to time out on a hot add operation if the Data Link Layer State Changed event does not occur within 1 second. The action taken by software after such a timeout is implementation specific.

6.7.3.4 *Software Notification of Hot-Plug Events*

A hot-plug capable Downstream Port must support generation of an interrupt on a hot-plug event. As described in Sections 6.7.3.1 and 6.7.3.2, each hot-plug event has both an enable bit for interrupt generation and a status bit that indicates when an event has occurred but has not yet been processed by software. There is also a Hot-Plug Interrupt Enable bit in the Slot Control register that serves as a master enable/disable bit for all hot-plug events.

If the Port is enabled for level-triggered interrupt signaling using the INTx messages, the virtual INTx wire must be asserted whenever and as long as the following conditions are satisfied:

- The Interrupt Disable bit in the Command register is set to 0b.
- The Hot-Plug Interrupt Enable bit in the Slot Control register is set to 1b.
- At least one hot-plug event status bit in the Slot Status register and its associated enable bit in the Slot Control register are both set to 1b.

Note that all other interrupt sources within the same Function will assert the same virtual INTx wire when requesting service.

If the Port is enabled for edge-triggered interrupt signaling using MSI or MSI-X, an interrupt message must be sent every time the logical AND of the following conditions transitions from FALSE to TRUE:

- The associated vector is unmasked (not applicable if MSI does not support PVM).
- The Hot-Plug Interrupt Enable bit in the Slot Control register is set to 1b.
- At least one hot-plug event status bit in the Slot Status register and its associated enable bit in the Slot Control register are both set to 1b.

Note that PME and Hot-Plug Event interrupts (when both are implemented) always share the same MSI or MSI-X vector, as indicated by the Interrupt Message Number field in the PCI Express Capabilities register.

The Port may optionally send an MSI when there are hot-plug events that occur while interrupt generation is disabled, and interrupt generation is subsequently enabled.

If wake generation is required by the associated form factor specification, a hot-plug capable Downstream Port must support generation of a wakeup event (using the PME mechanism) on hot-plug events that occur when the system is in a sleep state or the Port is in device state D1, D2, or $D3_{Hot}$.

Software enables a hot-plug event to generate a wakeup event by enabling software notification of the event as described in Section 6.7.3.1. Note that in order for software to disable interrupt generation while keeping wakeup generation enabled, the Hot-Plug Interrupt Enable bit must be cleared. For form factors that support wake generation, a wakeup event must be generated if all three of the following conditions occur:

- The status register for an enabled event transitions from Clearto Set
- The Port is in device state D1, D2, or $D3_{Hot}$, and
- The PME_En bit in the Port's Power Management Control/Status register is Set

Note that the Hot-Plug Controller generates the wakeup on behalf of the hot-plugged device, and it is not necessary for that device to have auxiliary (or main) power.

6.7.4 Firmware Support for Hot-Plug

Some systems that include hot-plug capable Root Ports and Switches that are released before ACPI-compliant operating systems with native hot-plug support are available, can use ACPI firmware for propagating hot-plug events. Firmware control of the hot-plug registers must be disabled if an operating system with native support is used. Platforms that provide ACPI firmware to propagate hot-plug events must also provide a mechanism to transfer control to the operating system. The details of this method are described in the *PCI Firmware Specification*.

6.7.5 Async Removal

Async removal refers to the removal of an adapter or disabling of a Downstream Port Link due to error containment without prior warning to the operating system. This is contrast to standard PCI hot-plug where removal operations are performed in a lock-step manner with the operating system through a well defined sequence of user actions and system management facilities. For example, the user presses the Attention Button to request permission from the operating system to remove the adapter, but the user doesn't actually remove the adapter from the slot until the operating system has quiesced activity to the adapter and granted permission for removal.

Since async removal proceeds before the rest of the PCI Express hierarchy or operating system necessarily becomes aware of the event, special consideration is required beyond that needed for standard PCI hot-plug. This section outlines PCI Express events that may occur as a side effect of async removal and mechanisms for handling async removal.

Since async removal may be unexpected to both the Physical and Data Link Layers of the Downstream Port associated with the slot, Correctable Errors may be reported as a side effect of the event (i.e. Receiver Error, Bad TLP, and Bad DLLP). If these errors are reported, software should handle them as an expected part of this event.

Requesters may experience Completion Timeouts associated with Requests that were accepted, but will never be completed by removed Completers. Any resulting Completion Timeout errors in this context should be handled as an expected part of this event.

Async removal may result in a transition from DL_Active to DL_Down in the Downstream port. This transition may result in a Surprise Down error. In addition, Requesters in the PCI Express hierarchy domain may not become immediately aware of this transition and continue to issue Requests to removed Completers that must be handled by the Downstream Port associated with the slot.

The Surprise Down error resulting from async removal may trigger Downstream Port Containment (See Section 6.2.10). Downstream Port Containment following an async removal may be utilized to hold the Link of a Downstream Port in the Disabled LTSSM state while host software recovers from the side effects of an async removal.

6.8 Power Budgeting Capability

With the addition of a hot-plug capability for adapters, the need arises for the system to be capable of properly allocating power to any new devices added to the system. This capability is a separate and distinct function from power management and a basic level of support is required to ensure proper operation of the system. The power budgeting concept puts in place the building blocks that allow devices to interact with systems to achieve these goals. There are many ways in which the system can implement the actual power budgeting capabilities, and as such, they are beyond the scope of this specification.

Implementation of the Power Budgeting Capability is optional for devices that are implemented either in a form factor which does not require hot-plug support, or that are integrated on the system board. Form factor specifications may require support for power budgeting. The devices and/or adapters are required to remain under the configuration power limit specified in the corresponding electromechanical specification until they have been configured and enabled by the system. The system should guarantee that power has been properly budgeted prior to enabling an adapter.

6.8.1 System Power Budgeting Process Recommendations

It is recommended that system firmware provide the power budget management agent the following information:

- Total system power budget (power supply information).
- Total power allocated by system firmware (system board devices).

- Total number of slots and the types of slots.

System firmware is responsible for allocating power for all devices on the system board that do not have power budgeting capabilities. The firmware may or may not include devices that are connected to the standard power rails. When the firmware allocates the power for a device that implements the Power Budgeting Capability it must set the System Allocated bit to 1b in the Power Budget Capability register to indicate that it has been properly allocated. The power budget manager is responsible for allocating all PCI Express devices including system board devices that have the Power Budgeting Capability and have the System Allocated bit Clear. The power budget manager is responsible for determining if hot-plugged devices can be budgeted and enabled in the system.

There are alternate methods which may provide the same functionality, and it is not required that the power budgeting process be implemented in this manner.

6.9 Slot Power Limit Control

PCI Express provides a mechanism for software controlled limiting of the maximum power per slot that an adapter (associated with that slot) can consume. If supported, the Emergency Power Reduction State, over-rides the mechanisms listed here (see Section 6.25). The key elements of this mechanism are:

- Slot Power Limit Value and Scale fields of the Slot Capabilities register implemented in the Downstream Ports of a Root Complex or a Switch
- Captured Slot Power Limit Value and Scale fields of the Device Capabilities register implemented in Endpoint, Switch, or PCI Express-PCI Bridge Functions present in an Upstream Port
- Set_Slot_Power_Limit Message that conveys the content of the Slot Power Limit Value and Scale fields of the Slot Capabilities register of the Downstream Port (of a Root Complex or a Switch) to the corresponding Captured Slot Power Limit Value and Scale fields of the Device Capabilities register in the Upstream Port of the component connected to the same Link

Power limits on the platform are typically controlled by the software (for example, platform firmware) that comprehends the specifics of the platform such as:

- Partitioning of the platform, including slots for I/O expansion using adapters
- Power delivery capabilities
- Thermal capabilities

This software is responsible for correctly programming the Slot Power Limit Value and Scale fields of the Slot Capabilities registers of the Downstream Ports connected to slots. After the value has been written into the register within the Downstream Port, it is conveyed to the adapter using the Set_Slot_Power_Limit Message (see Section 2.2.8.5). The recipient of the Message must use the value in the Message data payload to limit usage of the power for the entire adapter, unless the adapter will never exceed the lowest value specified in the corresponding form factor specification. It is required that device driver software associated with the adapter be able (by reading the values of the Captured Slot Power Limit Value and Scale fields of the Device Capabilities register) to configure hardware of the adapter to guarantee that the adapter will not exceed the imposed limit. In the case where the platform imposes a limit that is below the minimum needed for adequate

operation, the device driver will be able to communicate this discrepancy to higher level configuration software. Configuration software is required to set the Slot Power Limit to one of the maximum values specified for the corresponding form factor based on the capability of the platform.

The following rules cover the Slot Power Limit control mechanism:

For Adapters:

- Until and unless a Set_Slot_Power_Limit Message is received indicating a Slot Power Limit value greater than the lowest value specified in the form factor specification for the adapter's form factor, the adapter must not consume more than the lowest value specified.
- An adapter must never consume more power than what was specified in the most recently received Set_Slot_Power_Limit Message or the minimum value specified in the corresponding form factor specification, whichever is higher.
- Components with Endpoint, Switch, or PCI Express-PCI Bridge Functions that are targeted for integration on an adapter where total consumed power is below the lowest limit defined for the targeted form factor are permitted to ignore Set_Slot_Power_Limit Messages, and to return a value of 0 in the Captured Slot Power Limit Value and Scale fields of the Device Capabilities register
    - Such components still must be able to receive the Set_Slot_Power_Limit Message without error but simply discard the Message value

For Root Complex and Switches which source slots:

- Configuration software must not program a Set_Slot_Power_Limit value that indicates a limit that is lower than the lowest value specified in the form factor specification for the slot's form factor.

IMPLEMENTATION NOTE

Example Adapter Behavior Based on the Slot Power Limit Control Capability

The following power limit scenarios are examples of how an adapter must behave based on the Slot Power Limit control capability. The form factor limits are representations, and should not be taken as actual requirements.

Note: Form factor #1 has a maximum power requirement of 40 W and 25 W; form factor #2 has a maximum power requirement of 15 W.

Scenario 1: An Adapter Consuming 12 W

- If the adapter is plugged into a form factor #1 40 W slot, the Slot Power Limit control mechanism is followed, and the adapter operates normally.
- If the adapter is plugged into a form factor #1 25 W slot, the Slot Power Limit control mechanism is followed, and the adapter operates normally.
- If the adapter is plugged into a form factor #2 15 W slot, the Slot Power Limit control mechanism is followed, and the adapter operates normally.

In all cases, since the adapter operates normally within all the form factors, it can ignore any of the slot power limit Messages.

Scenario 2: An Adapter Consuming 18 W

- If the adapter is plugged into a form factor #1 40 W slot, the Slot Power Limit control mechanism is followed, and the adapter operates normally.
- If the adapter is plugged into a form factor #1 25 W slot, the Slot Power Limit control mechanism is followed, and the adapter operates normally.
- If the adapter is plugged into a form factor #2 15 W slot, the Slot Power Limit control mechanism is followed, and the adapter must scale down to 15 W or disable operation. An adapter that does not scale within any of the power limits for a given form factor will always be disabled in that form factor and should not be used.

In this case, if the adapter is only to be used in form factor #1, it can ignore any of the slot power limit Messages. To be useful in form factor #2, the adapter should be capable of scaling to the power limit of form factor #2.

Scenario 3: An Adapter Consuming 30 W

- If the adapter is plugged into a form factor #1 40 W slot, the Slot Power Limit control mechanism is followed, and the device operates normally.
- If the adapter is plugged into a form factor #1 25 W slot, the Slot Power Limit control mechanism is followed, and the device must scale down to 25 W or disable operation.
- If the adapter is plugged into a form factor #2 15 W slot, the Slot Power Limit control mechanism is followed, and the adapter must scale down to 15 W or disable operation. An adapter that does not scale within any of the power limits for a given form factor will always be disabled in that form factor and should not be used.

In this case, since the adapter consumes power above the lowest power limit for a slot, the adapter must be capable of scaling or disabling to prevent system failures. Operation of adapters at power levels that exceed the capabilities of the slots in which they are plugged must be avoided.

IMPLEMENTATION NOTE

Slot Power Limit Control Registers

Typically Slot Power Limit register fields within Downstream Ports of a Root Complex or a Switch will be programmed by platform-specific software. Some implementations may use a hardware method for initializing the values of these registers and, therefore, do not require software support.

Components with Endpoint, Switch, or PCI Express-PCI Bridge Functions that are targeted for integration on the adapter where total consumed power is below the lowest limit defined for that form factor are allowed to ignore Set_Slot_Power_Limit Messages. Note that components that take this implementation approach may not be compatible with potential future defined form factors. Such form factors may impose lower power limits that are below the minimum required by a new adapter based on the existing component.

IMPLEMENTATION NOTE

Auto Slot Power Limit Disable

In some environments host software may wish to directly manage the transmission of a Set_Slot_Power_Limit message by performing a Configuration Write to the Slot Capabilities register rather than have the transmission automatically occur when the Link transitions from a non-DL_Up to a DL_Up status. This allows host software to limit power supply surge current by staggering the transition of Endpoints to a higher power state following a Link Down or when multiple Endpoints are simultaneously hot-added due to cable or adapter insertion.

6.10 Root Complex Topology Discovery

A Root Complex may present one of the following topologies to configuration software:

- A single opaque Root Complex such that software has no visibility with respect to internal operation of the Root Complex. All Root Ports are independent of each other from a software perspective; no mechanism exists to manage any arbitration among the various Root Ports for any differentiated services.
- A single Root Complex Component such that software has visibility and control with respect to internal operation of the Root Complex Component. As shown in Figure 6-11, software views the Root Ports as Ingress Ports for the component. The Root Complex internal Port for traffic aggregation to a system Egress Port or an internal sink unit (such as memory) is represented by an RCRB structure. Controls for differentiated services are provided through a Virtual Channel Capability structure located in the RCRB.

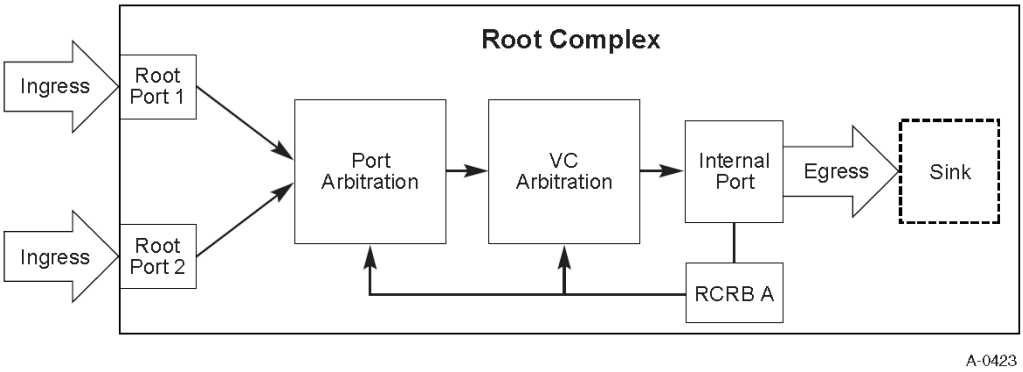


Figure 6-11: Root Complex Represented as a Single Component

- Multiple Root Complex Components such that software not only has visibility and control with respect to internal operation of a given Root Complex Component but also has the ability to discover and control arbitration between different Root Complex Components. As shown in Figure 6-12, software views the Root Ports as Ingress Ports for a given component. An RCRB structure controls egress from the component to other Root Complex Components (RCRB C) or to an internal sink unit such as memory (RCRB A). In addition, an RCRB structure (RCRB B) may also be present in a given component to control traffic from other Root Complex Components. Controls for differentiated services are provided through Virtual Channel Capability structures located appropriately in the RCRBs respectively.

More complex topologies are possible as well.

A Root Complex topology can be represented as a collection of logical Root Complex Components such that each logical component has:

- One or more Ingress Ports.
- An Egress Port.
- Optional associated Virtual Channel capabilities located either in the Configuration Space (for Root Ports) or in an RCRB (for internal Ingress/Egress Ports) if the Root Complex supports Virtual Channels.
- Optional devices/Functions integrated in the Root Complex.

In order for software to correctly program arbitration and other control parameters for PCI Express differentiated services, software must be able to discover a Root Complex's internal topology. Root Complex topology discovery is accomplished by means of the Root Complex Link Declaration Capability as described in Section 7.9.8.

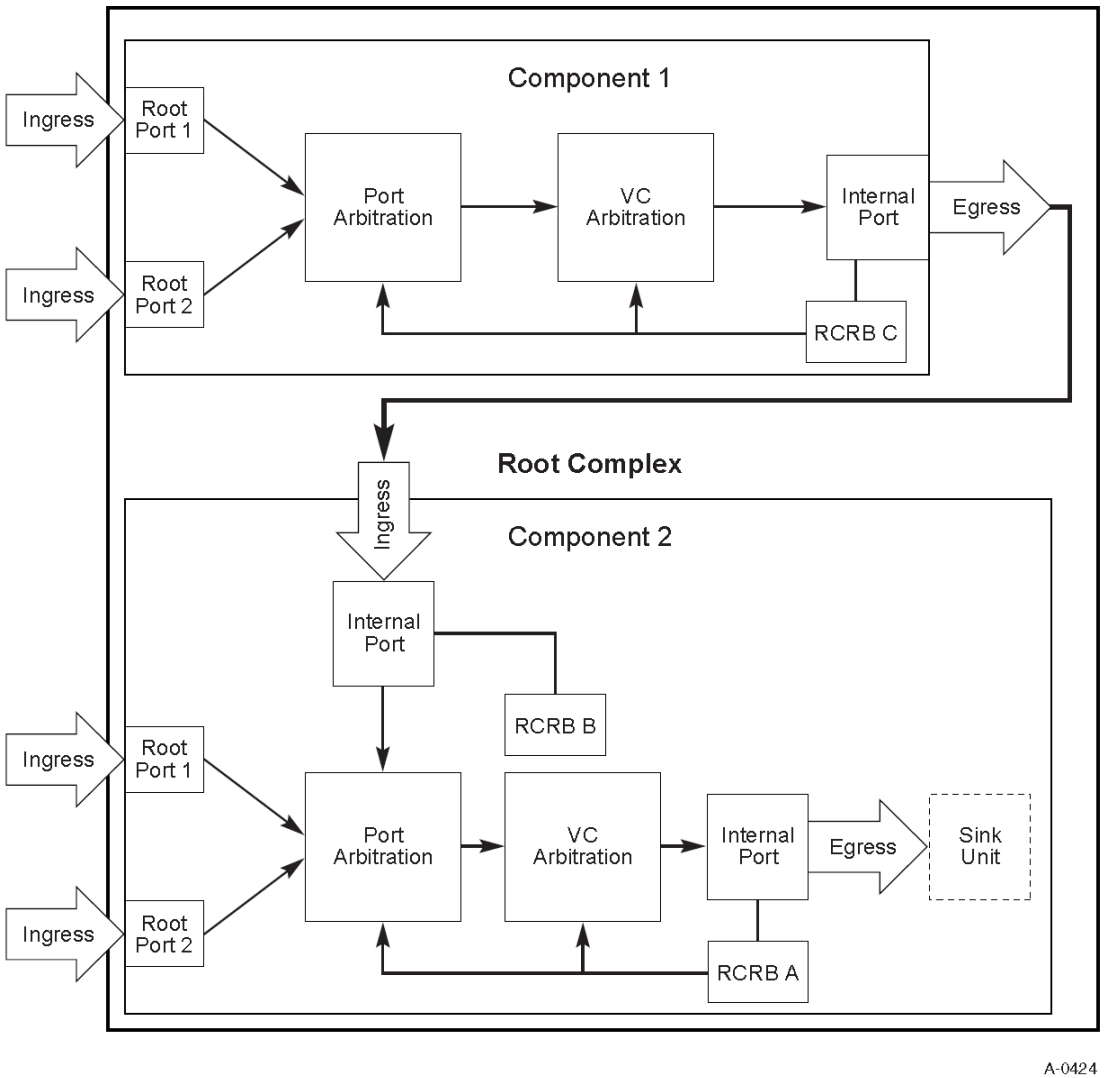


Figure 6-12: Root Complex Represented as Multiple Components

6.11 Link Speed Management

This section describes how Link speed management is coordinated between the LTSSM (Section 4.2.6) and the software Link observation and control mechanisms (Sections 7.5.3.6, 7.5.3.7, 7.5.3.8, 7.5.3.18, 7.5.3.19, and 7.5.3.20).

The Target Link Speed field in the Link Control 2 register in the Downstream Port sets the upper bound for the Link speed. Except as described below, the Upstream component must attempt to maintain the Link at the Target Link Speed, or at the highest speed supported by both components on the Link (as reported by the values in the training sets – see Section 4.2.4.1), whichever is lower.

Any Upstream Port or Downstream Port with the Hardware Autonomous Speed Disable bit in the Link Control 2 register clear is permitted to autonomously change the Link speed using implementation specific criteria.

If the reliability of the Link is unacceptably low, then either component is permitted to lower the Link speed by removing the unreliable Link speed from the list of supported speeds advertised in the training sets the component transmits. The criteria for determination of acceptable Link reliability are implementation specific, and are not dependent on the setting of the Hardware Autonomous Speed Disable bit.

During any given speed negotiation it is possible that one or both components will advertise a subset of all speeds supported, as a means to cap the post-negotiation Link speed. It is permitted for a component to change its set of advertised supported speeds without requesting a Link speed change by driving the Link through Recovery without setting the speed change bit.

When a component's attempt to negotiate to a particular Link speed fails, that component is not permitted to attempt negotiation to that Link speed, or to any higher Link speed, until 200 ms has passed from the return to L0 following the failed attempt, or until the other component on the Link advertises support for the higher Link speed through its transmitted training sets (with or without a request to change the Link speed), whichever comes first.

Software is permitted to restrict the maximum speed of Link operation and set the preferred Link speed by setting the value in the Target Link Speed field in the Upstream component. After modifying the value in the Target Link Speed field, software must trigger Link retraining by writing 1b to the Retrain Link bit. Software is notified of any Link speed changes (as well as any Link width changes) through the Link Bandwidth Notification Mechanism.

Software is permitted to cause a Link to transition to the Polling.Compliance LTSSM state at a particular speed by writing the Link Control 2 register in both components with the same value in the Target Link Speed field and Setting the Enter Compliance bit, and then initiating a Hot Reset on the Link (through the Downstream Port).

Note that this will take the Link to a DL_Down state and therefore cannot be done transparently to other software that is using the Link. The Downstream Port will return to Polling.Active when the Enter Compliance bit is cleared.

6.12 Access Control Services (ACS)

ACS defines a set of control points within a PCI Express topology to determine whether a TLP is to be routed normally, blocked, or redirected. ACS is applicable to RCs, Switches, and Multi-Function Devices.[98] For ACS requirements, single-Function devices that are SR-IOV capable must be handled as if they were Multi-Function Devices, since they essentially behave as Multi-Function Devices after their Virtual Functions (VFs) are enabled.

Implementation of ACS in RCiEPs is permitted but not required. It is explicitly permitted that, within a single Root Complex, some RCiEPs implement ACS and some do not. It is strongly recommended that Root Complex implementations ensure that all accesses originating from RCiEPs (PFs and VFs) without ACS capability are first subjected to processing by the Translation Agent (TA) in the Root Complex before further decoding and processing. The details of such Root Complex handling are outside the scope of this specification.

ACS provides the following types of access control:

[98] Applicable Functions within Multi-Function Devices specifically include PCI Express Endpoints, Switch Upstream Ports, Legacy PCI Express Endpoints, and Root Complex Integrated Endpoints.

- ACS Source Validation (V)
- ACS Translation Blocking (B)
- ACS P2P Request Redirect (R)
- ACS P2P Completion Redirect (C)
- ACS Upstream Forwarding (U)
- ACS P2P Egress Control (E)
- ACS Direct Translated P2P (T)

The specific requirements for each of these are discussed in the following section. The letter in parenthesis following each type is the abbreviation for the associated capability and control bits defined in Section 7.7.7.

ACS hardware functionality is disabled by default, and is enabled only by ACS-aware software. With the exception of ACS Source Validation, ACS access controls are not applicable to Multicast TLPs (see Section 6.14), and have no effect on them.

6.12.1 ACS Component Capability Requirements

ACS functionality is reported and managed via ACS Extended Capability structures. PCI Express components are permitted to implement ACS Extended Capability structures in some, none, or all of their applicable Functions. The extent of what is implemented is communicated through capability bits in each ACS Extended Capability structure. A given Function with an ACS Extended Capability structure may be required or forbidden to implement certain capabilities, depending upon the specific type of the Function and whether it is part of a Multi-Function Device.

ACS is never applicable to a PCI Express to PCI Bridge Function or a Root Complex Event Collector Function, and such Functions must never implement an ACS Extended Capability structure.

6.12.1.1 *ACS Downstream Ports*

This section applies to Root Ports and Downstream Switch Ports that implement an ACS Extended Capability structure. This section applies to Downstream Port Functions both for single-Function devices and Multi-Function Devices.

- ACS Source Validation: must be implemented.

  When enabled, the Downstream Port tests the Bus Number from the Requester ID of each Upstream Request received by the Port to determine if it is associated with the Secondary side of the virtual bridge associated with the Downstream Port, by either or both of:

  - Determining that the Requester ID falls within the Bus Number "aperture" of the Port – the inclusive range specified by the Secondary Bus Number register and the Subordinate Bus Number register.
  - If FPB is implemented and enabled, determining that the Requester ID is associated with the bridge's Secondary Side by the application of the FPB Routing ID mechanism.

  If the Bus Number from the Requester ID of the Request is not within this aperture, this is a reported error (ACS Violation) associated with the Receiving Port (see Section 6.12.4.)

  Completions are never affected by ACS Source Validation.

IMPLEMENTATION NOTE

Upstream Messages and ACS Source Validation

Functions are permitted to transmit Upstream Messages before they have been assigned a Bus Number. Such messages will have a Requester ID with a Bus Number of 00h. If the Downstream Port has ACS Source Validation enabled, these Messages (see Table F-1 and Section 6.23.1) will likely be detected as an ACS Violation error.

- ACS Translation Blocking: must be implemented.

  When enabled, the Downstream Port checks the Address Translation (AT) field of each Upstream Memory Request received by the Port. If the AT field is not the default value, this is a reported error (ACS Violation) associated with the Receiving Port (see Section 6.12.4). This error must take precedence over ACS Upstream Forwarding and any applicable ACS P2P control mechanisms.

  Completions are never affected by ACS Translation Blocking.

- ACS P2P Request Redirect: must be implemented by Root Ports that support peer-to-peer traffic with other Root Ports;[99] must be implemented by Switch Downstream Ports.

  ACS P2P Request Redirect is subject to interaction with the ACS P2P Egress Control and ACS Direct Translated P2P mechanisms (if implemented). Refer to Section 6.12.3 for more information.

---

[99] Root Port indication of ACS P2P Request Redirect or ACS P2P Completion Redirect support does not imply any particular level of peer-to-peer support by the Root Complex, or that peer-to-peer traffic is supported at all

When ACS P2P Request Redirect is enabled in a Switch Downstream Port, peer-to-peer Requests must be redirected Upstream towards the RC.

When ACS P2P Request Redirect is enabled in a Root Port, peer-to-peer Requests must be sent to Redirected Request Validation logic within the RC that determines whether the Request is "reflected" back Downstream towards its original target, or blocked as an ACS Violation error. The algorithms and specific controls for making this determination are not architected by this specification.

Downstream Ports never redirect Requests that are traveling Downstream.

Completions are never affected by ACS P2P Request Redirect.

- ACS P2P Completion Redirect: must be implemented by Root Ports that implement ACS P2P Request Redirect; must be implemented by Switch Downstream Ports.

  The intent of ACS P2P Completion Redirect is to avoid ordering rule violations between Completions and Requests when Requests are redirected. Refer to Section 6.12.5 for more information.

  ACS P2P Completion Redirect does not interact with ACS controls that govern Requests.

  When ACS P2P Completion Redirect is enabled in a Switch Downstream Port, peer-to-peer Completions[100] that do not have the Relaxed Ordering Attribute bit set (1b) must be redirected Upstream towards the RC. Otherwise, peer-to-peer Completions must be routed normally.

  When ACS P2P Completion Redirect is enabled in a Root Port, peer-to-peer Completions that do not have the Relaxed Ordering bit set must be handled such that they do not pass Requests that are sent to Redirected Request Validation logic within the RC. Such Completions must eventually be sent Downstream towards their original peer-to-peer targets, without incurring additional ACS access control checks.

  Downstream Ports never redirect Completions that are traveling Downstream.

  Requests are never affected by ACS P2P Completion Redirect.

- ACS Upstream Forwarding: must be implemented by Root Ports if the RC supports Redirected Request Validation; must be implemented by Switch Downstream Ports.

  When ACS Upstream Forwarding is enabled in a Switch Downstream Port, and its Ingress Port receives an Upstream Request or Completion TLP targeting the Port's own Egress Port, the Port must instead forward the TLP Upstream towards the RC.

  When ACS Upstream Forwarding is enabled in a Root Port, and its Ingress Port receives an Upstream Request or Completion TLP that targets the Port's own Egress Port, the Port must handle the TLP as follows. For a Request, the Root Port must handle it the same as a Request that the Port "redirects" with the ACS P2P Request Redirect mechanism. For a Completion, the Root Port must handle it the same as a Completion that the Port "redirects" with the ACS P2P Completion Redirect mechanism.

  When ACS Upstream Forwarding is not enabled on a Downstream Port, and its Ingress Port receives an Upstream Request or Completion TLP that targets the Port's own Egress Port, the handling of the TLP is undefined.

---

[100] This includes Read Completions, AtomicOp Completions, and other Completions with or without Data.

- ACS P2P Egress Control: implementation is optional.

  ACS P2P Egress Control is subject to interaction with the ACS P2P Request Redirect and ACS Direct Translated P2P mechanisms (if implemented). Refer to Section 6.12.3 for more information.

  A Switch that supports ACS P2P Egress Control can be selectively configured to block peer-to-peer Requests between its Downstream Ports. Software can configure the Switch to allow none or only a subset of its Downstream Ports to send peer-to-peer Requests to other Downstream Ports. This is configured on a per Downstream Port basis.

  An RC that supports ACS P2P Egress Control can be selectively configured to block peer-to-peer Requests between its Root Ports. Software can configure the RC to allow none or only a subset of the Hierarchy Domains to send peer-to-peer Requests to other Hierarchy Domains. This is configured on a per Root Port basis.

  With ACS P2P Egress Control in Downstream Ports, controls in the Ingress Port ("sending" Port) determine if the peer-to-peer Request is blocked, and if so, the Ingress Port handles the ACS Violation error per Section 6.12.4.

  Completions are never affected by ACS P2P Egress Control.

- ACS Direct Translated P2P: must be implemented by Root Ports that support Address Translation Services (ATS) and also support peer-to-peer traffic with other Root Ports;[101] must be implemented by Switch Downstream Ports.

  When ACS Direct Translated P2P is enabled in a Downstream Port, peer-to-peer Memory Requests whose Address Type (AT) field indicates a Translated address must be routed normally ("directly") to the peer Egress Port, regardless of ACS P2P Request Redirect and ACS P2P Egress Control settings. All other peer-to-peer Requests must still be subject to ACS P2P Request Redirect and ACS P2P Egress Control settings.

  Completions are never affected by ACS Direct Translated P2P.

6.12.1.2 *ACS Functions in SR-IOV Capable and Multi-Function Devices*

This section applies to Multi-Function Device ACS Functions, with the exception of Downstream Port Functions, which are covered in the preceding section. For ACS requirements, single-Function devices that are SR-IOV capable must be handled as if they were Multi-Function Devices.

- ACS Source Validation: must not be implemented.
- ACS Translation Blocking: must not be implemented.
- ACS P2P Request Redirect: must be implemented by Functions that support peer-to-peer traffic with other Functions. This includes SR-IOV Virtual Functions (VFs).

---

[101] Root Port indication of ACS Direct Translated P2P support does not imply any particular level of peer-to-peer support by the Root Complex, or that peer-to-peer traffic is supported at all.

ACS P2P Request Redirect is subject to interaction with the ACS P2P Egress Control and ACS Direct Translated P2P mechanisms (if implemented). Refer to Section 6.12.3 for more information.

When ACS P2P Request Redirect is enabled in a multi-Function Device that is not an RCiEP, peer-to-peer Requests (between Functions of the device) must be redirected Upstream towards the RC.

It is permitted but not required to implement ACS P2P Request Redirect in an RCiEP. When ACS P2P Request Redirect is enabled in an RCiEP, peer-to-peer Requests, defined as all Requests that do not target system memory, must be sent to implementation-specific logic within the Root Complex that determines whether the Request is directed towards its original target, or blocked as an ACS Violation error. The algorithms and specific controls for making this determination are not architected by this specification.

Completions are never affected by ACS P2P Request Redirect.

- ACS P2P Completion Redirect: must be implemented by Functions that implement ACS P2P Request Redirect.

  The intent of ACS P2P Completion Redirect is to avoid ordering rule violations between Completions and Requests when Requests are redirected. Refer to Section 6.12.5 for more information.

  ACS P2P Completion Redirect does not interact with ACS controls that govern Requests.

  When ACS P2P Completion Redirect is enabled in a multi-Function Device that is not an RCiEP, peer-to-peer Completions that do not have the Relaxed Ordering bit set must be redirected Upstream towards the RC. Otherwise, peer-to-peer Completions must be routed normally.

  Requests are never affected by ACS P2P Completion Redirect.

- ACS Upstream Forwarding: must not be implemented.

- ACS P2P Egress Control: implementation is optional; is based on Function Numbers or Function Group Numbers; controls peer-to-peer Requests between the different Functions within the multi-Function or SR-IOV capable device.

  ACS P2P Egress Control is subject to interaction with the ACS P2P Request Redirect and ACS Direct Translated P2P mechanisms (if implemented). Refer to Section 6.12.3 for more information.

  Each Function within a multi-Function Device that supports ACS P2P Egress Control can be selectively enabled to block peer-to-peer communication with other Functions or Function Groups[102] within the device. This is configured on a per Function basis.

  With ACS P2P Egress Control in multi-Function or SR-IOV capable devices, controls in the "sending" Function determine if the Request is blocked, and if so, the "sending" Function handles the ACS Violation error per Section 6.12.4.

  When ACS Function Groups are enabled in an ARI Device, ACS P2P Egress Controls are enforced on a per Function Group basis instead of a per Function basis. See Section 6.13.

[102] ACS Function Groups capability is optional for ARI Devices that implement ACS P2P Egress Controls.

Completions are never affected by ACS P2P Egress Control.

- ACS Direct Translated P2P: must be implemented if the Multi-Function Device Function supports Address Translation Services (ATS) and also peer-to-peer traffic with other Functions.

  When ACS Direct Translated P2P is enabled in a multi-Function Device, peer-to-peer Memory Requests whose Address Type (AT) field indicates a Translated address must be routed normally ("directly") to the peer Function, regardless of ACS P2P Request Redirect and ACS P2P Egress Control settings. All other peer-to-peer Requests must still be subject to ACS P2P Request Redirect and ACS P2P Egress Control settings.

  Completions are never affected by ACS Direct Translated P2P.

6.12.1.3 *Functions in Single-Function Devices*

This section applies to single-Function device Functions, with the exception of Downstream Port Functions and SR-IOV capable Functions, which are covered in a preceding section. For ACS requirements, single-Function devices that are SR-IOV capable must be handled as if they were Multi-Function Devices.

No ACS capabilities are applicable, and the Function must not implement an ACS Extended Capability structure.

6.12.2 Interoperability

The following rules govern interoperability between ACS and non-ACS components:

- When ACS P2P Request Redirect and ACS P2P Completion Redirect are not being used, ACS and non-ACS components may be intermixed within a topology and will interoperate fully. ACS can be enabled in a subset of the ACS components without impacting interoperability.
- When ACS P2P Request Redirect, ACS P2P Completion Redirect, or both are being used, certain components in the PCI Express hierarchy must support ACS Upstream Forwarding (of Upstream redirected Requests). Specifically:

The associated Root Port[103] must support ACS Upstream Forwarding. Otherwise, how the Root Port handles Upstream redirected Request or Completion TLPs is undefined. The RC must also implement Redirected Request Validation.

Between each ACS component where P2P TLP redirection is enabled and its associated Root Port, any intermediate Switches must support ACS Upstream Forwarding. Otherwise, how such Switches handle Upstream redirected TLPs is undefined.

6.12.3 ACS Peer-to-Peer Control Interactions

With each peer-to-peer Request, multiple ACS control mechanisms may interact to determine whether the Request is routed directly towards its peer-to-peer target, blocked immediately as an ACS Violation, or redirected Upstream towards the RC for access validation. Peer-to-peer Completion redirection is determined exclusively by the ACS P2P Completion Redirect mechanism.

---

[103] Not applicable for ACS Redirect between Functions of a multi-Function Root Complex Integrated Endpoint.

If ACS Direct Translated P2P is enabled in a Port/Function, peer-to-peer Memory Requests whose Address Translation (AT) field indicates a Translated address must be routed normally ("directly") to the peer Port/Function, regardless of ACS P2P Request Redirect and ACS P2P Egress Control settings. Otherwise such Requests, and unconditionally all other peer-to-peer Requests, must be subject to ACS P2P Request Redirect and ACS P2P Egress Control settings. Specifically, the applicable Egress Control Vector bit, along with the ACS P2P Egress Control Enable bit (E) and the ACS P2P Request Redirect Enable bit (R), determine how the Request is handled. It must be noted that atomicity of accesses cannot be guaranteed if ACS peer-to-peer Request Redirect targets a legacy device location that can be the target of a locked access. Refer to Section 7.7.7 for descriptions of these control bits. Table 6-9 specifies the interactions.

Table 6-9: ACS P2P Request Redirect and ACS P2P Egress Control Interactions

Control Bit E (b)	Control Bit R (b)	Egress Control Vector Bit for the Associated Egress Switch Port, Root Port, Function, or Function Group	Required Handling for Peer-to-Peer Requests
0	0	X – Don't care	Route directly to peer-to-peer target
0	1	X – Don't Care	Redirect Upstream
1	0	1	Handle as an ACS Violation
1	0	0	Route directly to peer-to-peer target
1	1	1	Redirect Upstream
1	1	0	Route directly to peer-to-peer target

6.12.4 ACS Violation Error Handling

ACS Violations may occur due to either hardware or software defects/failures. To assist in fault isolation and root cause analysis, it is recommended that AER be implemented in ACS components. AER prefix/header logging and the Prefix Log/Header Log registers may be used to determine the prefix/header of the offending Request. The ACS Violation Status, Mask, and Severity bits provide positive identification of the error and increased control over error logging and signaling.

When an ACS Violation is detected, the ACS component that operates as the Completer[104] must do the following:

- For Non-Posted Requests, the Completer must generate a Completion with a Completer Abort (CA) Completion Status.
- The Completer must log and signal the ACS Violation as indicated in Figure 6-2. Note the following:

[104] In all cases but one, the ACS component that detects the ACS Violation also operates as the Completer. The exception case is when Root Complex Redirected Request Validation logic disallows a redirected Request. If the redirected Request came through a Root Port, that Root Port must operate as the Completer. If the redirected Request came from a Root Complex Integrated Endpoint, the associated Root Complex Event Collector must operate as the Completer.

- Even though the Completer uses a CA Completion Status when it sends a Completion, the Completer must log an ACS Violation error instead of a Completer Abort error.
- If the severity of the ACS Violation is non-fatal and the Completer sends a Completion with CA Completion Status, this case must be handled as an Advisory Non-Fatal Error as described in Section 6.2.3.2.4.1.

- The Completer[105] must set the Signaled Target Abort bit in either its Status register or Secondary Status register as appropriate.

6.12.5 ACS Redirection Impacts on Ordering Rules

When ACS P2P Request Redirect is enabled, some or all peer-to-peer Requests are redirected, which can cause ordering rule violations in some cases. This section explores those cases, plus a similar case that occurs with RCs that implement "Request Retargeting" as an alternative mechanism for enforcing peer-to-peer access control.

6.12.5.1 *Completions Passing Posted Requests*

When a peer-to-peer Posted Request is redirected, a subsequent peer-to-peer non-RO[106] Completion that is routed directly can effectively pass the redirected Posted Request, violating the ordering rule that non-RO Completions must not pass Posted Requests. Refer to Section 2.4.1 for more information.

ACS P2P Completion Redirect can be used to avoid violating this ordering rule. When ACS P2P Completion Redirect is enabled, all peer-to-peer non-RO Completions will be redirected, thus taking the same path as redirected peer-to-peer Posted Requests. Enabling ACS P2P Completion Redirect when some or all peer-to-peer Requests are routed directly will not cause any ordering rule violations, since it is permitted for a given Completion to be passed by any TLP other than another Completion with the same Transaction ID.

As an alternative mechanism to ACS P2P Request Redirect for enforcing peer-to-peer access control, some RCs implement "Request Retargeting", where the RC supports special address ranges for "peer-to-peer" traffic, and the RC will retarget validated Upstream Requests to peer devices. Upon receiving an Upstream Request targeting a special address range, the RC validates the Request, translates the address to target the appropriate peer device, and sends the Request back Downstream. With retargeted Requests that are Non-posted, if the RC does not modify the Requester ID, the resulting Completions will travel "directly" peer-to-peer back to the original Requester, creating the possibility of non-RO Completions effectively passing retargeted Posted Requests, violating the same ordering rule as when ACS P2P Request Redirect is being used. ACS P2P Completion Redirect can be used to avoid violating this ordering rule here as well.

---

[105] Similarly, if the Request was Non-Posted, when the Requester receives the resulting Completion with CA Completion Status, the Requester must set the Received Target Abort bit in either its Status register or Secondary Status register as appropriate. Note that for the case of a Multi-Function Device incurring an ACS Violation error with a peer-to-peer Request between its Functions, the same Function might serve both as Requester and Completer.

[106] In this section, "non-RO" is an abbreviation characterizing TLPs whose Relaxed Ordering Attribute field is not set.

If ACS P2P Request Redirect and RC P2P Request Retargeting are not being used, there is no envisioned benefit to enabling ACS P2P Completion Redirect, and it is recommended not to do so because of potential performance impacts.

IMPLEMENTATION NOTE

Performance Impacts with ACS P2P Completion Redirect

While the use of ACS P2P Completion Redirect can avoid ordering violations with Completions passing Posted Requests, it also may impact performance. Specifically, all redirected Completions will have to travel up to the RC from the point of redirection and back, introducing extra latency and possibly increasing Link and RC congestion.

Since peer-to-peer Completions with the Relaxed Ordering bit set are never redirected (thus avoiding performance impacts), it is strongly recommended that Requesters be implemented to maximize the proper use of Relaxed Ordering, and that software enable Requesters to utilize Relaxed Ordering by setting the Enable Relaxed Ordering bit in the Device Control register.

If software enables ACS P2P Request Redirect, RC P2P Request Retargeting, or both, and software is certain that proper operation is not compromised by peer-to-peer non-RO Completions passing peer-to-peer[107] Posted Requests, it is recommended that software leave ACS P2P Completion Redirect disabled as a way to avoid its performance impacts.

6.12.5.2 *Requests Passing Posted Requests*

When some peer-to-peer Requests are redirected but other peer-to-peer Requests are routed directly, the possibility exists of violating the ordering rules where Non-posted Requests or non-RO Posted Requests must not pass Posted Requests. Refer to Section 2.4.1 for more information.

These ordering rule violation possibilities exist only when ACS P2P Request Redirect and ACS Direct Translated P2P are both enabled. Software should not enable both these mechanisms unless it is certain either that such ordering rule violations cannot occur, or that proper operation will not be compromised if such ordering rule violations do occur.

IMPLEMENTATION NOTE

Ensuring Proper Operation with ACS Direct Translated P2P

The intent of ACS Direct Translated P2P is to optimize performance in environments where Address Translation Services (ATS) are being used with peer-to-peer communication whose access control is enforced by the RC. Permitting peer-to-peer Requests with Translated addresses to be routed directly avoids possible performance impacts associated with redirection, which introduces extra latency and may increase Link and RC congestion.

[107] These include true peer-to-peer Requests that are redirected by the ACS P2P Request Redirect mechanism, as well as “logically peer-to-peer” Requests routed to the Root Complex that the Root Complex then retargets to the peer device.

For the usage model where peer-to-peer Requests with Translated addresses are permitted, but those with Untranslated addresses are to be blocked as ACS Violations, it is recommended that software enable ACS Direct Translated P2P and ACS P2P Request Redirect, and configure the Redirected Request Validation logic in the RC to block the redirected Requests with Untranslated addresses. This configuration has no ordering rule violations associated with Requests passing Posted Requests.

For the usage model where some Requesters use Translated addresses exclusively with peer-to-peer Requests and some Requesters use Untranslated addresses exclusively with peer-to-peer Requests, and the two classes of Requesters do not communicate peer-to-peer with each other, proper operation is unlikely to be compromised by redirected peer-to-peer Requests (with Untranslated addresses) being passed by direct peer-to-peer Requests (with Translated addresses). It is recommended that software not enable ACS Direct Translated P2P unless software is certain that proper operation is not compromised by the resulting ordering rule violations.

For the usage model where a single Requester uses both Translated and Untranslated addresses with peer-to-peer Requests, again it is recommended that software not enable ACS Direct Translated P2P unless software is certain that proper operation is not compromised by the resulting ordering rule violations. This requires a detailed analysis of the peer-to-peer communications models being used, and is beyond the scope of this specification.

6.13 Alternative Routing-ID Interpretation (ARI)

Routing IDs, Requester IDs, and Completer IDs are 16-bit identifiers traditionally composed of three fields: an 8-bit Bus Number, a 5-bit Device Number, and a 3-bit Function Number. With ARI, the 16-bit field is interpreted as two fields instead of three: an 8-bit Bus Number and an 8-bit Function Number – the Device Number field is eliminated. This new interpretation enables an ARI Device to support up to 256 Functions [0..255] instead of 8 Functions [0..7].

ARI is controlled by a new set of optional capability and control register bits. These provide:

- Software the ability to detect whether a component supports ARI.
- Software the ability to configure an ARI Downstream Port so the logic that determines when to turn a Type 1 Configuration Request into a Type 0 Configuration Request no longer enforces a restriction on the traditional Device Number field being 0.
- Software the ability to configure an ARI Device to assign each Function to a Function Group. Controls based on Function Groups may be preferable when finer granularity controls based on individual Functions are not required.
    - If Multi-Function VC arbitration is supported and enabled, arbitration can optionally be based on Function Groups instead of individual Functions.
    - If ACS P2P Egress Controls are supported and enabled, access control can optionally be based on Function Groups instead of individual Functions.

The following illustrates an example flow for enabling these capabilities and provides additional details on their usage:

1. Software enumerates the PCI Express hierarchy and determines whether the ARI capability is supported.

a. For an ARI Downstream Port, the capability is communicated through the Device Capabilities 2 register.

b. For an ARI Device, the capability is communicated through the ARI Capability structure.

c. ARI has no impact on the base enumeration algorithms used in platforms today.

2. Software enables ARI functionality in each component.

a. In an ARI Downstream Port immediately above an ARI Device, software sets the ARI Forwarding Enable bit in the Device Control 2 register. Setting this bit ensures the logic that determines when to turn a Type 1 Configuration Request into a Type 0 Configuration Request no longer enforces a restriction on the traditional Device Number field being 0.

b. In an ARI Device, Extended Functions are always implicitly enabled. However, once ARI-aware software enables ARI Forwarding in the Downstream Port immediately above the ARI Device, ARI-aware software must discover and configure the Extended Functions.

c. If an ARI Device implements a Multi-Function VC Capability structure with Function arbitration, and also implements MFVC Function Groups, ARI-aware software categorizes Functions into Function Groups.

i. Each Function is assigned to a Function Group represented by a Function Group Number.

ii. A maximum of 8 Function Groups can be configured.

iii. Within the Multi-Function VC Arbitration Table, a Function Group Number is used in place of a Function Number in each arbitration slot.

1. Arbitration occurs on a Function Group basis instead of an individual Function basis.

2. All other aspects of Multi-Function VC arbitration remain unchanged. See Section 7.9.2.10 for additional details.

iv. Function arbitration within each Function Group is implementation-specific.

d. If an ARI Device supports ACS P2P Egress Control, access control can be optionally implemented on a Function Group basis.

e. To improve the enumeration performance and create a more deterministic solution, software can enumerate Functions through a linked list of Function Numbers. The next linked list element is communicated through each Function's ARI Capability register.

i. Function 0 acts as the head of a linked list of Function Numbers. Software detects a non-zero Next Function Number field within the ARI Capability register as the next Function within the linked list. Software issues a configuration probe using the Bus Number captured by the Device and the Function Number derived from the ARI Capability register to locate the next associated Function's configuration space.

ii. Function Numbers may be sparse and non-sequential in their consumption by an ARI Device.

With an ARI Device, the Phantom Functions Supported field within each Function's Device Capabilities register (see Section 7.5.3.3, Table 7-18) must be set to 00b to indicate that Phantom Functions are not supported. The Extended Tag Field Enable bit and the 10-Bit Tag Requester Enable bit can still be used to enable each Function to support higher numbers of outstanding Requests. See Section 2.2.6.2.

Figure 6-13 shows an example system topology with two ARI Devices, one below a Root Port and one below a Switch. For access to Extended Functions in ARI Device X, Root Port A must support ARI Forwarding and have it enabled by software. For access to Extended Functions in ARI Device Y, Switch Downstream Port D must support ARI Forwarding and have it enabled by software. With this configuration, it is recommended that software not enable ARI Forwarding in Root Port B or Switch Downstream Port C.

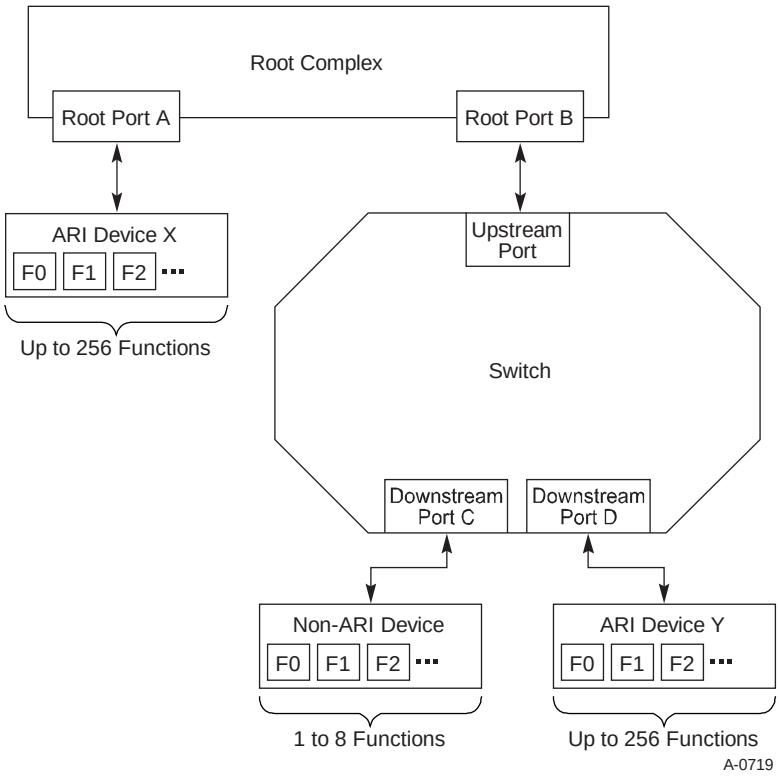


Figure 6-13: Example System Topology with ARI Devices

IMPLEMENTATION NOTE

ARI Forwarding Enable Being Set Inappropriately

It is strongly recommended that software in general Set the ARI Forwarding Enable bit in a Downstream Port only if software is certain that the device immediately below the Downstream Port is an ARI Device. If the bit is Set when a non-ARI Device is present, the non-ARI Device can respond to Configuration Space accesses under what it interprets as being different Device Numbers, and its Functions can be aliased under multiple Device Numbers, generally leading to undesired behavior.

Following a hot-plug event below a Downstream Port, it is strongly recommended that software Clear the ARI Forwarding Enable bit in the Downstream Port until software determines that a newly added component is in fact an ARI Device.

IMPLEMENTATION NOTE

ARI Forwarding Enable Setting at Firmware/Operating System Control Handoff

It is strongly recommended that firmware not have the ARI Forwarding Enable bit Set in a Downstream Port upon control handoff to an operating system unless firmware knows that the operating system is ARI-aware. With this bit Set, a non-ARI-aware operating system might be able to discover and enumerate Extended Functions in an ARI Device below the Downstream Port, but such an operating system would generally not be able to manage Extended Functions successfully, since it would interpret there being multiple Devices below the Downstream Port instead of a single ARI Device. As one example of many envisioned problems, the interrupt binding for INTx virtual wires would not be consistent with what the non-ARI-aware operating system would expect.

6.14 Multicast Operations

The Multicast Capability structure defines a Multicast address range, the segmentation of that range into a number, N, of equal sized Multicast Windows, and the association of each Multicast Window with a Multicast Group, MCG. Each Function that supports Multicast within a component implements a Multicast Capability structure that provides routing directions and permission checking for each MCG for TLPs passing through or to the Function. The Multicast Group is a field of up to 6 bits in width embedded in the address beginning at the MC_Index_Position, as defined in Section 7.9.11.4.

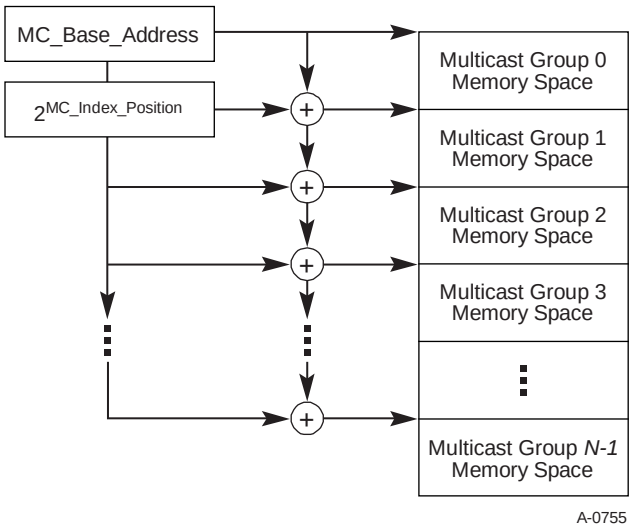


Figure 6-14: Segmentation of the Multicast Address Range

6.14.1 Multicast TLP Processing

A Multicast Hit occurs if all of the following are true:

- MC_Enable is Set
- TLP is a Memory Write or an Address Routed Message, both of which are Posted Requests
- $Address_{TLP}$ >= MC_Base_Address
- $Address_{TLP}$ < (MC_Base_Address+ ($2^{MC_Index_Position} * (MC_Num_Group + 1)$))

In this step, each Switch Ingress Port and other components use values of MC_Enable, MC_Base_Address, MC_Index_Position, and MC_Num_Group from any one of their Functions. Software is required to configure all Functions of a Switch and all Functions of a Multi-Function Upstream Port to have the same values in each of these fields and results are indeterminate if this is not the case.

If the address in a Non-Posted Memory Request hits in a Multicast Window, no Multicast Hit occurs and the TLP is processed normally per the base specification – i.e., as a unicast.

If a Multicast Hit occurs, the only ACS access control that can still apply is ACS Source Validation. In particular, neither ACS redirection nor the ACS Egress Control vector affects operations during a Multicast Hit.

If a Multicast Hit occurs, normal address routing rules do not apply. Instead, the TLP is processed as follows:

The Multicast Group is extracted from the address in the TLP using any Function's values for MC_Base_Address and MC_Index_Position. Specifically:

$$MCG = ((Address_{TLP} - MC_Base_Address) >> MC_Index_Position) \& 3Fh$$

In this process, the component may use any Function's values for MC_Base_Address and MC_Index_Position. Which Function's values are used is device-specific.

Components next check the MC_Block_All and the MC_Block_Untranslated bits corresponding to the extracted MCG. Switches and Root Ports check Multicast TLPs in their Ingress Ports using the MC_Block_All and MC_Block_Untranslated registers associated with the Ingress Port. Endpoint Functions check Multicast TLPs they are preparing to send, using their MC_Block_All and MC_Block_Untranslated registers. If the MC_Block_All bit corresponding to the extracted MCG is set, the TLP is handled as an MC Blocked TLP. If the MC_Block_Untranslated bit corresponding to the extracted MCG is set and the TLP contains an Untranslated Address, the TLP, is also handled as an MC Blocked TLP.

IMPLEMENTATION NOTE

MC_Block_Untranslated and PIO Writes

Programmed I/O (PIO) Writes to Memory Space generally have Untranslated addresses since there is no architected mechanism for software to control the Address Type (AT) field for PIO Requests. Thus, if it's necessary for a given Switch to Multicast any PIO Writes, software should ensure that the appropriate MC_Block_Untranslated bits in the Upstream Port of that Switch are Clear. Otherwise, the Switch Upstream Port may block PIO Writes that legitimately target Multicast Windows. Since it may be necessary for software to clear MC_Block_Untranslated bits in a Switch Upstream Port for the sake of PIO Writes, the following are strongly recommended for a Root Complex capable of Address translation:·

- All Integrated Endpoints each implement a Multicast Capability structure to provide access control for sending Untranslated Multicast TLPs.·
- All peer-to-peer capable Root Ports each implement a Multicast Capability structure to provide access control for Untranslated Multicast TLPs that are forwarded peer-to-peer.

For similar reasons, with Multicast-capable Switch components where the Upstream Port is a Function in a Multi-Function Device, it is strongly recommended that any Endpoints in that Multi-Function Device each implement a Multicast Capability structure.

IMPLEMENTATION NOTE

Multicast Window Size

Each ultimate Receiver of a Multicast TLP may have a different Multicast Window size requirement. At one extreme, a Multicast Window may be required to cover a range of memory implemented within the device. At the other, it may only need to cover a particular offset at which a FIFO register is located. The MC_Window_Size_Requested field within the Multicast Capability register is used by an Endpoint to advertise the size of Multicast Window that it requires.

Unless available address space is limited, resource allocation software may be able to treat each request as a minimum and set the Multicast Window size via MC_Index_Position to accommodate the largest request. In some cases, a request for a larger window size can be satisfied by configuring a smaller window size and assigning the same membership to multiple contiguous MCGs.

IMPLEMENTATION NOTE

Multicast, ATS, and Redirection

The ACS P2P Request Redirection and ACS Direct Translated P2P mechanisms provide a means where P2P Requests with Untranslated Addresses can be redirected to the Root Complex (RC) for access control checking, whereas P2P Requests with Translated Addresses can be routed "directly" to their P2P targets for improved performance. No corresponding redirection mechanism exists for Multicast TLPs.

To achieve similar functionality, an RC might be configured to provide one or more target Memory Space ranges that are not in the Multicast address range, but the RC maps to "protected" Multicast Windows. Multicast TLP senders either with or without ATS capability then target these RC Memory Space ranges in order to access the protected Multicast Windows indirectly. When either type of sender targets these ranges with Memory Writes, each TLP that satisfies the access control checks will be reflected back down by the RC with a Translated Address targeting a protected Multicast Window.[108] ATS-capable senders can request and cache Translated Addresses using the RC Memory Space range, and then later use those Translated Addresses for Memory Writes that target protected Multicast Windows directly and can be Multicast without a taking a trip through the RC.

For hardware enforcement that only Translated Addresses can be used to target the protected Multicast Windows directly, software Sets appropriate MCG bits in the MC_Block_Untranslated register in all applicable Functions throughout the platform. Each MCG whose bit is set will cause its associated Multicast Window to be protected from direct access using Untranslated Addresses.

If the TLP is not blocked in a Switch or Root Complex it is forwarded out all of the Ports, except its Ingress Port, whose MC_Receive bit corresponding to the extracted MCG is set. In an Endpoint, it is consumed by all Functions whose MC_Receive bit corresponding to the extracted MCG is set. If no Ports forward the TLP or no Functions consume it, the TLP is silently dropped.

To prevent loops, it is prohibited for a Root Port or a Switch Port to forward a TLP back out its Ingress Port, even if so specified by the MC_Receive register associated with the Port. An exception is the case described in the preceding Implementation Note, where an RC reflects a unicast TLP that came in on an Ingress Root Port to a Multicast Window. In that case, when specified by the MC_Receive register associated with that Ingress Root Port, the RC is required to send the reflected TLP out the same Root Port that it originally came in.

A Multicast Hit suspends normal address routing, including default Upstream routing in Switches. When a Multicast Hit occurs, the TLP will be forwarded out only those Egress Ports whose MC_Receive bit associated with the MCG extracted from the address in the TLP is set. If the address in the TLP does not decode to any Downstream Port using normal address decode, the TLP will be copied to the Upstream Port only if so specified by the Upstream Port's MC_Receive register.

---

[108] If the original sender belongs to the MCG associated with this Window, the original sender will also receive a copy of the reflected TLP.

6.14.2 Multicast Ordering

No new ordering rules are defined for processing Multicast TLPs. All Multicast TLPs are Posted Requests and follow Posted Request ordering rules. Multicast TLPs are ordered per normal ordering rules relative to other TLPs in a component's ingress stream through the point of replication. Once copied into an egress stream, a Multicast TLP follows the same ordering as other Posted Requests in the stream.

6.14.3 Multicast Capability Structure Field Updates

Some fields of the Multicast Capability structure may be changed at any time. Others cannot be changed with predictable results unless the MC_Enable bit is Clear in every Function of the component. The latter group includes MC_Base_Address and MC_Index_Position.

Fields which software may change at any time include MC_Enable, MC_Num_Group, MC_Receive, MC_Block_All, and MC_Block_Untranslated. Updates to these fields must themselves be ordered. Consider, for example, TLPs A and B arriving in that order at the same Ingress Port and in the same TC. If A uses value X for one of these fields, then B must use the same value or a newer value.

For Multi-Function Upstream Switch Ports Multicast TLPs received by one Switch or transmitted by one Endpoint Function are presented to the other parallel Endpoint Functions and the Downstream Switch Ports of the other parallel Switches (Functions are considered to be parallel if they are in the same Device. A single Multicast TLP is forwarded Upstream when any of the Upstream Switch Functions has the appropriate MC_Receive bit Set.

6.14.4 MC Blocked TLP Processing

When a TLP is blocked by the MC_Block_All or the MC_Block_Untranslated mechanisms, the TLP is dropped. The Function blocking the TLP serves as the Completer. The Completer must log and signal this MC Blocked TLP error as indicated in Figure 6-2. In addition, the Completer must set the Signaled Target Abort bit in either its Status register or Secondary Status register as appropriate. To assist in fault isolation and root cause analysis, it is highly recommended that AER be implemented in Functions with Multicast capability.

In Root Complexes and Switches, if the error occurs with a TLP received by an Ingress Port, the error is reported by that Ingress Port. If the error occurs in an Endpoint Function preparing to send the TLP, the error is reported by that Endpoint Function.

6.14.5 MC_Overlay Mechanism

The MC Overlay mechanism is provided to allow a single BAR in an Endpoint that doesn't contain a Multicast Capability structure to be used for both Multicast and unicast TLP reception. Software can configure the MC_Overlay mechanism to affect this by setting the MC_Overlay_BAR in a Downstream Port so that the Multicast address range, or a portion of it, is remapped (overlaid) onto the Memory Space range accepted by the Endpoint's BAR. At the Upstream Port of a Switch, the mechanism can be used to overlay a portion of the Multicast address range onto a Memory Space range associated with host memory.

A Downstream Port's MC_Overlay mechanism applies to TLPs exiting that Port. An Upstream Port's MC_Overlay mechanism applies to TLPs exiting the Switch heading Upstream. A Port's MC_Overlay mechanism does not apply to TLPs received by the Port, to TLPs targeting memory space within the Port, or to TLPs routed Peer-to-Peer between Functions in a Multi-Function Upstream Port.

When enabled, the overlay operation specifies that bits in the address in the Multicast TLP, whose bit numbers are equal to or higher than the MC_Overlay_Size field, be replaced by the corresponding bits in the MC_Overlay_BAR. In other words:

```
If (MC_Overlay_Size < 6)
Then Egress_TLP_Addr = Ingress_TLP_Addr;
Else Egress_TLP_Addr = {MC_Overlay_BAR[63:MC_Overlay_Size],
                        Ingress_TLP_Addr[MC_Overlay_Size-1:0]};
```

If the TLP with modified address contains the optional ECRC, the unmodified ECRC will almost certainly indicate an error. The action to be taken if a TLP containing an ECRC is Multicast copied to an Egress Port that has MC_Overlay enabled depends upon whether or not optional support for ECRC regeneration is implemented. All of the contingent actions are outlined in Table 6-10. If MC_Overlay is not enabled, the TLP is forwarded unmodified. If MC_Overlay is enabled and the TLP has no ECRC, the modified TLP, with its address replaced as specified in the previous paragraph is forwarded. If the TLP has an ECRC but ECRC regeneration is not supported, then the modified TLP is forwarded with its ECRC dropped and the TD bit in the header cleared to indicate no ECRC attached. If the TLP has an ECRC and ECRC regeneration is supported, then an ECRC check is performed before the TLP is forwarded. If the ECRC check passes, the TLP is forwarded with regenerated ECRC. If the ECRC check fails, the TLP is forwarded with inverted regenerated ECRC.

Table 6-10: ECRC Rules for MC_Overlay

MC_Overlay Enabled	TLP has ECRC	ECRC Regeneration Supported	Action if ECRC Check Passes	Action if ECRC Check Fails
No	x	x	Forward TLP unmodified	
Yes	No	x	Forward modified TLP	
Yes	Yes	No	Forward modified TLP with ECRC dropped and TD bit clear	
Yes	Yes	Yes	Forward modified TLP with regenerated ECRC	Forward modified TLP with inverted regenerated ECRC

IMPLEMENTATION NOTE

MC_Overlay and ECRC Regeneration

Switch and Root Complex Ports have the option to support ECRC regeneration. If ECRC regeneration is supported, then it is highly advised to do so robustly by minimizing the time between checking the ECRC of the original TLP and replacing it with an ECRC computed on the modified TLP. The TLP is unprotected during this time, leaving a data integrity hole if the pre-check and regeneration aren't accomplished in the same pipeline stage.

Stripping the ECRC from Multicast TLPs passing through a Port that has MC_Overlay enabled but doesn't support ECRC regeneration allows the receiving Endpoint to enable ECRC checking. In such a case, the Endpoint will enjoy the benefits of ECRC on non-Multicast TLPs without detecting ECRC on Multicast TLPs modified by the MC_Overlay mechanism.

When Multicast ECRC regeneration is supported, and an ECRC error is detected prior to TLP modification, then inverting the regenerated ECRC ensures that the ECRC error isn't masked by the regeneration process.

IMPLEMENTATION NOTE

Multicast to Endpoints That Don't Have Multicast Capability

An Endpoint Function that doesn't contain a Multicast Capability structure cannot distinguish Multicast TLPs from unicast TLPs. It is possible for a system designer to take advantage of this fact to employ such Endpoints as Multicast targets. The primary requirement for doing so is that the base and limit registers of the virtual PCI to PCI Bridge in the Switch Port above the device be configured to overlap at least part of the Multicast address range or that the MC_Overlay mechanism be employed. Extending this reasoning, it is even possible that a single Multicast target Function could be located on the PCI/PCI-X side of a PCI Express to PCI/PCI-X Bridge.

If an Endpoint without a Multicast Capability structure is being used as a Multicast target and the MC_Overlay mechanism isn't used, then it may be necessary to read from the Endpoint's Memory Space using the same addresses used for Multicast TLPs. Therefore, Memory Reads that hit in a Multicast Window aren't necessarily errors. Memory Reads that hit in a Multicast Window and that don't also hit in the aperture of an RCiEP or the Downstream Port of a Switch will be routed Upstream, per standard address routing rules, and be handled as a UR there.

IMPLEMENTATION NOTE

Multicast in a Root Complex

A Root Complex with multiple Root Ports that supports Multicast may implement as many Multicast Capability structures as its implementation requires. If it implements more than one, software should ensure that certain fields, as specified in Section 6.14.3, are configured identically. To support Multicast to RCiEPs, the implementation needs to expose all TLPs identified as Multicast via the MC_Base_Address register to all potential Multicast target Endpoints integrated within it. Each such Integrated Endpoint then uses the MC_Receive register in its Multicast Capability structure to determine if it should receive the TLP.

IMPLEMENTATION NOTE

Multicast and Multi-Function Devices

All Port Functions and Endpoint Functions that are potential Multicast targets need to implement a Multicast Capability structure so that each has its own MC_Receive vector. Within a single component, software should configure the MC_Enable, MC_Base_Address, MC_Index_Position, and MC_Num_Group fields of these Capability structures identically. That being the case, it is sufficient to implement address decoding logic on only one instance of the Multicast BAR in the component.

IMPLEMENTATION NOTE

Congestion Avoidance

The use of Multicast increases the output link utilization of Switches to a degree proportional to both the size of the Multicast groups used and the fraction of Multicast traffic to total traffic. This results in an increased risk of congestion and congestion spreading when Multicast is used.

To mitigate this risk, components that are intended to serve as Multicast targets should be designed to consume Multicast TLPs at wire speed. Components that are intended to serve as Multicast sources should consider adding a rate limiting mechanism.

In many applications, the application's Multicast data flow will have an inherent rate limit and can be accommodated without causing congestion. Others will require an explicit mechanism to limit the injection rate, selection of a Switch with buffers adequate to hold the requisite bursts of Multicast traffic without asserting flow control, or selection of Multicast target components capable of sinking the Multicast traffic at the required rate. It is the responsibility of the system designer to choose the appropriate mechanisms and components to serve the application.

IMPLEMENTATION NOTE

The Host as a Multicast Recipient

For general-purpose systems, it is anticipated that the Multicast address range will usually not be configured to overlap with Memory Space that's directly mapped to host memory. If host memory is to be included as a Multicast recipient, the Root Complex may need to have some sort of I/O Memory Management Unit (IOMMU) that is capable of remapping portions of Multicast Windows to host memory, perhaps with page-level granularity. Alternatively, the MC_Overlay mechanism in the Upstream Port of a Switch can be used to overlay a portion of the Multicast address range onto host memory.

For embedded systems that lack an IOMMU, it may be feasible to configure Multicast Windows overlapping with Memory Space that's directly mapped to host memory, thus avoiding the need for an IOMMU. Specific details of this approach are beyond the scope of this specification.

6.15 Atomic Operations (AtomicOps)

An Atomic Operation (AtomicOp) is a single PCI Express transaction that targets a location in Memory Space, reads the location's value, potentially writes a new value back to the location, and returns the original value. This "read-modify-write" sequence to the location is performed atomically. AtomicOps include the following:

- FetchAdd (Fetch and Add): Request contains a single operand, the "add" value
  - Read the value of the target location.
  - Add the "add" value to it using two's complement arithmetic ignoring any carry or overflow.

  - Write the sum back to the target location.
  - Return the original value of the target location.
- Swap (Unconditional Swap): Request contains a single operand, the "swap" value
  - Read the value of the target location.
  - Write the "swap" value back to the target location.
  - Return the original value of the target location.
- CAS (Compare and Swap): Request contains two operands, a "compare" value and a "swap" value
  - Read the value of the target location.
  - Compare that value to the "compare" value.
  - If equal, write the "swap" value back to the target location.
  - Return the original value of the target location.

A given AtomicOp transaction has an associated operand size, and the same size is used for the target location accesses and the returned value. FetchAdd and Swap support operand sizes of 32 and 64 bits. CAS supports operand sizes of 32, 64, and 128 bits.

AtomicOp capabilities are optional normative. Endpoints and Root Ports are permitted to implement AtomicOp Requester capabilities. PCI Express Functions with Memory Space BARs as well as all Root Ports are permitted to implement AtomicOp Completer capabilities. Routing elements (Switches, as well as Root Complexes supporting peer-to-peer access between Root Ports) require AtomicOp routing capability in order to route AtomicOp Requests. AtomicOps are architected for device-to-host, device-to-device, and host-to-device transactions. In each case, the Requester, Completer, and all intermediate routing elements must support the associated AtomicOp capabilities.

AtomicOp capabilities are not supported on PCI Express to PCI/PCI-X Bridges. If need be, Locked Transactions can be used for devices below such Bridges. AtomicOps and Locked Transactions can operate concurrently on the same hierarchy.

Software discovers specific AtomicOp Completer capabilities via three new bits in the Device Capabilities 2 register (see Section 7.5.3.15). For increased interoperability, Root Ports are required to implement certain AtomicOp Completer capabilities in sets if at all (see Section 6.15.3.1). Software discovers AtomicOp routing capability via the AtomicOp Routing Supported bit in the Device Capabilities 2 register. Software discovery of AtomicOp Requester capabilities is outside the scope of this specification, but software must set the AtomicOp Requester Enable bit in a Function's Device Control 2 register before the Function can initiate AtomicOp Requests (see Section 7.5.3.16).

With routing elements, software can set an AtomicOp Egress Blocking bit (see Section 7.5.3.16) on a Port-by-Port basis to avoid AtomicOp Requests being forwarded to components that shouldn't receive them, and might handle each as a Malformed TLP, which by default is a Fatal Error. Each blocked Request is handled as an AtomicOp Egress Blocked error, which by default is an Advisory Non-Fatal Error.

AtomicOps are Memory Transactions, so existing standard mechanisms for managing Memory Space access (e.g., Bus Master Enable, Memory Space Enable, and Base Address registers) apply.

6.15.1 AtomicOp Use Models and Benefits

AtomicOps enable advanced synchronization mechanisms that are particularly useful when there are multiple producers and/or multiple consumers that need to be synchronized in a non-blocking fashion. For example, multiple producers can safely enqueue to a common queue without any explicit locking.

AtomicOps also enable lock-free statistics counters, for example where a device can atomically increment a counter, and host software can atomically read and clear the counter.

Direct support for the three chosen AtomicOps over PCI Express enables easier migration of existing high-performance SMP applications to systems that use PCI Express as the interconnect to tightly-coupled accelerators, co-processors, or GP-GPUs. For example, a ported application that uses PCI Express-attached accelerators may be able to use the same synchronization algorithms and data structures as the earlier SMP application.

An AtomicOp to a given target generally incurs latency comparable to a Memory Read to the same target. Within a single hierarchy, multiple AtomicOps can be "in flight" concurrently. AtomicOps generally create negligible disruption to other PCI Express traffic.

Compared to Locked Transactions, AtomicOps provide lower latency, higher scalability, advanced synchronization algorithms, and dramatically less impact to other PCI Express traffic.

6.15.2 AtomicOp Transaction Protocol Summary

Detailed protocol rules and requirements for AtomicOps are distributed throughout the rest of this specification, but here is a brief summary plus some unique requirements.

- AtomicOps are Non-Posted Memory Transactions, supporting 32- and 64-bit address formats.
- FetchAdd, Swap, and CAS each use a distinct type code.
- The Completer infers the operand size from the Length field value and type code in the AtomicOp Request.
- The endian format used by AtomicOp Completers to read and write data at the target location is implementation specific, and permitted to be whatever the Completer determines to be appropriate for the target memory (e.g., little-endian, big-endian, etc.). See Section 2.2.2.
- If an AtomicOp Requester supports Address Translation Services (ATS), the Requester is permitted to use a Translated address in an AtomicOp Request only if the Translated address has appropriate access permissions. Specifically, the Read (R) and Write (W) fields must both be Set, and the Untranslated access only (U) field must be Clear. See Section 2.2.4.1.
- If a component supporting Access Control Services (ACS) supports AtomicOp routing or AtomicOp Requester capability, it handles AtomicOp Requests and Completions the same as with other Memory Requests and Completions with respect to ACS functionality.
- The No Snoop attribute is applicable and permitted to be Set with AtomicOp Requests, but atomicity must be guaranteed regardless of the No Snoop attribute value.

- The Relaxed Ordering attribute is applicable and permitted to be Set with AtomicOp Requests, where it affects the ordering of both the Requests and their associated Completions.
- Ordering requirements for AtomicOp Requests are similar to those for Non-Posted Write Requests. Thus, if a Requester wants to ensure that an AtomicOp Request is observed by the Completer before a subsequent Posted or Non-Posted Request, the Requester must wait for the AtomicOp Completion before issuing the subsequent Request.
- Ordering requirements for AtomicOp Completions are similar to those for Read Completions.
- Unless there's a higher precedence error, an AtomicOp-aware Completer must handle a Poisoned AtomicOp Request as a Poisoned TLP Received error, and must also return a Completion with a Completion Status of Unsupported Request (UR). See Section 2.7.2.2. The value of the target location must remain unchanged.
- If the Completer of an AtomicOp Request encounters an uncorrectable error accessing the target location or carrying out the Atomic operation, the Completer must handle it as a Completer Abort (CA). The subsequent state of the target location is implementation specific.
- AtomicOp-aware Completers are required to handle any properly formed AtomicOp Requests with types or operand sizes they don't support as an Unsupported Request (UR). If the Length field in an AtomicOp Request contains an unarchitected value, the Request must be handled by an AtomicOp-aware Completer as a Malformed TLP. See Section 2.2.7.
- If any Function in a Multi-Function Device supports AtomicOp Completer or AtomicOp routing capability, all Functions with Memory Space BARs in that device must decode properly formed AtomicOp Requests and handle any they don't support as an Unsupported Request (UR). Note that in such devices, Functions lacking AtomicOp Completer capability are forbidden to handle properly formed AtomicOp Requests as Malformed TLPs.
- If an RC has any Root Ports that support AtomicOp routing capability, all RCiEPs in the RC reachable by forwarded AtomicOp Requests must decode properly formed AtomicOp Requests and handle any they don't support as an Unsupported Request (UR).
- With an AtomicOp Request having a supported type and operand size, the AtomicOp-aware Completer is required either to carry out the Request or handle it as Completer Abort (CA) for any location in its target Memory Space. Completers are permitted to support AtomicOp Requests on a subset of their target Memory Space as needed by their programming model (see Section 2.3.1). Memory Space structures defined or inherited by PCI Express (e.g., the MSI-X Table structure) are not required to be supported as AtomicOp targets unless explicitly stated in the description of the structure.
- For a Switch or an RC, when AtomicOp Egress Blocking is enabled in an Egress Port, and an AtomicOp Request targets going out that Egress Port, the Egress Port must handle the Request as an AtomicOp Egress Blocked error[109] (see Figure 6-2) and must also return a Completion with a Completion Status of UR. If the severity of the AtomicOp Egress Blocked error is non-fatal, this case must be handled as an Advisory Non-Fatal Error as described in Section 6.2.3.2.4.1.

---

[109] Though an AtomicOp Egress Blocked error is handled by returning a Completion with UR Status, the error is not otherwise handled as an Unsupported Request. For example, it does not set the Unsupported Request Detected bit in the Device Status register.

6.15.3 Root Complex Support for AtomicOps

RCs have unique requirements and considerations with respect to AtomicOp capabilities.

6.15.3.1 *Root Ports with AtomicOp Completer Capabilities*

AtomicOp Completer capability for a Root Port indicates that the Root Port supports receiving at its Ingress Port AtomicOp Requests that target host memory or Memory Space allocated by a Root Port BAR. This is independent of any RCiEPs that have AtomicOp Completer capabilities.

If a Root Port implements any AtomicOp Completer capability for host memory access, it must implement all 32-bit and 64-bit AtomicOp Completer capabilities. Implementing 128-bit CAS Completer capability is optional.

If an RC has one or more Root Ports that implement AtomicOp Completer capability, the RC must ensure that host memory accesses to a target location on behalf of a given AtomicOp Request are performed atomically with respect to each host processor or device access to that target location range.

If a host processor supports atomic operations via its instruction set architecture, the RC must also ensure that host memory accesses on behalf of a given AtomicOp Request preserve the atomicity of any host processor atomic operations.

6.15.3.2 *Root Ports with AtomicOp Routing Capability*

As with other PCI Express Transactions, the support for peer-to-peer routing of AtomicOp Requests and Completions between Root Ports is optional and implementation dependent. If an RC supports AtomicOp routing capability between two or more Root Ports, it must indicate that capability in each associated Root Port via the AtomicOp Routing Supported bit in the Device Capabilities 2 register.

An RC is not required to support AtomicOp routing between all pairs of Root Ports that have the AtomicOp Routing Supported bit Set. An AtomicOp Request that would require routing between unsupported pairs of Root Ports must be handled as an Unsupported Request (UR), and reported by the "sending" Port.

The AtomicOp Routing Supported bit must be Set for any Root Port that supports forwarding of AtomicOp Requests initiated by host software or RCiEPs. The AtomicOp Routing Supported bit must be Set for any Root Ports that support forwarding of AtomicOp Requests received on their Ingress Port to RCiEPs.

6.15.3.3 *RCs with AtomicOp Requester Capabilities*

An RC is permitted to implement the capability for either host software or RCiEPs to initiate AtomicOp Requests.

Software discovery of AtomicOp Requester capabilities is outside the scope of this specification.

If an RC supports software-initiated AtomicOp Requester capabilities, the specific mechanisms for how software running on a host processor causes the RC to generate AtomicOp Requests is outside the scope of this specification.

IMPLEMENTATION NOTE

Generating AtomicOp Requests via Host Processor Software

If a host processor instruction set architecture (ISA) supports atomic operation instructions that directly correspond to one or more PCI Express AtomicOps, an RC might process the associated internal atomic transaction that targets PCI Express Memory Space much like it processes the internal read transaction resulting from a processor load instruction. However, instead of "exporting" the internal read transaction as a PCI Express Memory Read Request, the RC would export the internal atomic transaction as a PCI Express AtomicOp Request. Even if an RC uses the "export" approach for some AtomicOp types and operand sizes, it would not need to use this approach for all.

For AtomicOp types and operand sizes where the RC does not use the "export" approach, the RC might use an RC register-based mechanism similar to one where some PCI host bridges use CONFIG_ADDRESS and CONFIG_DATA registers to generate Configuration Requests. Refer to the *PCI Local Bus Specification* for details.

The "export" approach may permit a large number of concurrent AtomicOp Requests without becoming RC register limited. It may also be easier to support AtomicOp Request generation from user space software using this approach.

The RC register-based mechanism offers the advantage of working for all AtomicOp types and operand sizes even if the host processor ISA doesn't support the corresponding atomic instructions. It might also support a polling mode for waiting on AtomicOp Completions as opposed to stalling the processor while waiting for a Completion.

6.15.4 Switch Support for AtomicOps

If a Switch supports AtomicOp routing capability for any of its Ports, it must do so for all of them.

6.16 Dynamic Power Allocation (DPA) Capability

A common approach to managing power consumption is through a negotiation between the device driver, operating system, and executing applications. Adding Dynamic Power Allocation for such devices is anticipated to be done as an extension of that negotiation, through software mechanisms that are outside of the scope of this specification. Some devices do not have a device specific driver to manage power efficiently. The DPA Capability provides a mechanism to allocate power dynamically for these types of devices. DPA is optional normative functionality applicable to Endpoint Functions that can benefit from the dynamic allocation of power and do not have an alternative mechanism. If supported, the Emergency Power Reduction State, over-rides the mechanisms listed here (see Section 6.25).

The DPA Capability enables software to actively manage and optimize Function power usage when in the D0 state. DPA is not applicable to power states D1-D3 therefore the DPA Capability is independently managed from the PCI-PM Capability.

DPA defines a set of power substates, each of which with an associated power allocation. Up to 32 substates [0..31] can be defined per Function. Substate 0, the default substate, indicates the maximum power the Function is ever capable of consuming.

Substates must be contiguously numbered from 0 to Substate_Max, as defined in Section 7.9.12.2. Each successive substate has a power allocation lower than or equal to that of the prior substate. For example, a Function with four substates could be defined as follows:

1. Substate 0 (the default) defines a power allocation of 25 Watts.
2. Substate 1 defines a power allocation of 20 Watts.
3. Substate 2 defines a power allocation of 20 Watts.
4. Substate 3 defines a power allocation of 10 Watts.

When the Function is initialized, it will operate within the power allocation associated with substate 0. Software is not required to progress through intermediate substates. Over time, software may dynamically configure the Function to operate at any of the substates in any sequence it chooses. Software is permitted to configure the Function to operate at any of the substates before the Function completes a previously initiated substate transition.

On the completion of the substate transition(s) the Function must compare its substate with the configured substate. If the Function substate does not match the configured substate, then the Function must begin transition to the configured substate. It is permitted for the Function to dynamically alter substate transitions on Configuration Requests instructing the Function to operate in a new substate.

In the prior example, software can configure the Function to transition to substate 4, followed by substate 1, followed by substate 3, and so forth. As a result, the Function must be able to transition between any substates when software configures the associated control field.

The Substate Control Enabled bit provides a mechanism that allows the DPA Capability to be used in conjunction with the software negotiation mechanism mentioned above. When Set, power allocation is controlled by the DPA Capability. When Clear, the DPA Capability is disabled, and the Function is not permitted to directly initiate substate transitions based on configuration of the Substate Control register field. At an appropriate point in time, software participating in the software negotiation mechanism mentioned above clears the bit, effectively taking over control of power allocation for the Function.

It is required that the Function respond to Configuration Space accesses while in any substate.

At any instant, the Function must never draw more power than it indicates through its Substate Status. When the Function is configured to transition from a higher power substate to a lower power substate, the Function's Substate Status must indicate the higher power substate during the transition, and must indicate the lower power substate after completing the transition. When the Function is configured to transition from a lower power substate to a higher power substate, the Function's Substate Status must indicate the higher power substate during the transition, as well as after completing the transition.

Due to the variety of applications and the wide range of maximum power required for a given Function, the transition time required between any substates is implementation specific. To enable software to construct power management policies (outside the scope of this specification), the Function defines two Transition Latency Values. Each of the Function substates associates its

maximum Transition Latency with one of the Transition Latency Values, where the maximum Transition Latency is the time it takes for the Function to enter the configured substate from any other substate. A Function is permitted to complete the substate transition faster than the maximum Transition Latency for the substate.

6.16.1 DPA Capability with Multi-Function Devices

It is permitted for some or all Functions of a Multi-Function Device to implement a DPA Capability. The power allocation for the Multi-Function Device is the sum of power allocations set by the DPA Capability for each Function. It is permitted for the DPA Capability of a Function to include the power allocation for the Function itself as well as account for power allocation for other Functions that do not implement a DPA Capability. The association between multiple Functions for DPA is implementation specific and beyond the scope of this specification.

6.17 TLP Processing Hints (TPH)

TLP Processing Hints is an optional feature that provides hints in Request TLP headers to facilitate optimized processing of Requests that target Memory Space. These Processing Hints enable the system hardware (e.g., the Root Complex and/or Endpoints) to optimize platform resources such as system and memory interconnect on a per TLP basis. The TPH mechanism defines Processing Hints that provide information about the communication models between Endpoints and the Root-complex. Steering Tags are system-specific values used to identify a processing resource that a Requester explicitly targets. System software discovers and identifies TPH capabilities to determine the Steering Tag allocation for each Function that supports TPH.

6.17.1 Processing Hints

The Requester provides hints to the Root Complex or other targets about the intended use of data and data structures by the host and/or device. The hints are provided by the Requester, which has knowledge of upcoming Request patterns, and which the Completer would not be able to deduce autonomously (with good accuracy). Cases of interest to distinguish with such hints include:

DWHR: Device writes then host reads soon

HWDR: Device reads data that the Host is believed to have recently written

D*D*: Device writes/reads, then device reads/writes soon

Includes DWDW, DWDR, DRDW, DRDR

Bi-Directional: Data structure that is shared and has equal read/write access by host and device.

The usage models are mapped to the Processing Hint encodings as described in Table 6-11.

Table 6-11: Processing Hint Mapping

PH[1:0] (b)	Processing Hint	Usage Model
00	Bi-directional data structure	Bi-Directional shared data structure

01	Requester	D*D*
10	Target	DWHR HWDR
11	Target with Priority	Same as target but with temporal re-use priority

6.17.2 Steering Tags

Functions that intend to target a TLP towards a specific processing resource such as a host processor or system cache hierarchy require topological information of the target cache (e.g., which host cache). Steering Tags are system-specific values that provide information about the host or cache structure in the system cache hierarchy. These values are used to associate processing elements within the platform with the processing of Requests.

Software programmable Steering Tag values to be used are stored in an ST Table that is permitted to be located either in the TPH Requester Capability structure (see Section 7.9.13) or combined with the MSI-X Table (see Section 7.7), but not in both locations for a given Function. When the ST Table is combined with the MSI-X Table, the 2 most significant bytes of the Vector Control register of each MSI-X Table entry are used to contain the Steering Tag value.

The choice of ST Table location is implementation specific and is discoverable by software. A Function that implements MSI-X is permitted to locate the ST Table in either location (see Section 7.9.13.2). A Function that implements both MSI and MSI-X is permitted to combine the ST Table with the MSI-X Table and use it, even when MSI-X is disabled (i.e. when MSI is enabled). Each ST Table entry is 2 bytes. The size of the ST Table is indicated in the TPH Requester Capability structure.

For some usage models the Steering Tags are not required or not provided, and in such cases a Function is permitted to use a value of all zeroes in the ST field to indicate no ST preference. The association of each Request with an ST Table entry is device specific and outside the scope of this specification.

6.17.3 ST Modes of Operation

The ST Table Location field in the TPH Requester Capability structure indicates where (if at all) the ST Table is implemented by the Function. If an ST Table is implemented, software can program it with the system-specific Steering Tag values.

Table 6-12: ST Modes of Operation

ST Mode Select [2:0] (b)	ST Mode Name	Description
000	No ST Mode	The Function must use a value of all zeroes for all Steering Tags.
001	Interrupt Vector Mode	Each Steering Tag is selected by an MSI/MSI-X interrupt vector number. The Function is required to use the Steering Tag value from an ST Table entry that can be indexed by a valid MSI/MSI-X interrupt vector number.
010	Device Specific Mode	It is recommended for the Function to use a Steering Tag value from an ST Table entry, but it is not required.
All other encodings	Reserved	Reserved for future use.

In the No ST Mode of operation, the Function must use a value of all zeroes for each Steering Tag, enabling the use of Processing Hints without software-provided Steering Tags.

In the Interrupt Vector Mode of operation, Steering Tags are selected from the ST Table using MSI/MSI-X interrupt vector numbers. For Functions that have MSI enabled, the Function is required to select tags within the range specified by the Multiple Message Enable field in the MSI Capability structure. For Functions that have MSI-X enabled, the Function is required to select tags within the range of the MSI-X Table size. If the ST Table Size is smaller than the enabled range of interrupt vector numbers, the Function is permitted to either not use TPH for certain transactions, to use TPH with a Steering Tag of 0 or to use TPH with an implementation defined mechanism used to select a Steering Tag value from the ST Table. If the ST Table Size is larger than the enabled range of interrupt vector numbers, ST Table Entries corresponding to out of range interrupt vector numbers are ignored by the Function.

In the Device Specific Mode of operation, the assignment of the Steering Tags to Requests is device specific. The number of Steering Tags used by the Function is permitted to be different than the number of interrupt vectors allocated for the Function, irrespective of the ST Table location, and Steering Tag values used in Requests are not required to come from the architected ST Table.

A Function that is capable of generating TPH Requests is required to support the No ST Mode of operation. Support for other ST Modes of operation is optional. Only one ST Mode of operation can be selected at a time by programming ST Mode Select.

IMPLEMENTATION NOTE

ST Table Programming

To ensure that deterministic Steering Tag values are used in Requests, it is recommended that software either quiesce the Function or disable the TPH Requester capability during the process of performing ST Table updates. Failure to do so may result in non-deterministic values of ST values being used during ST Table updates.

6.17.4 TPH Capability

TPH capabilities are optional normative. Each Function capable of generating Request TLPs with TPH is required to implement a TPH Requester Capability structure. Functions that support processing of TLPs with TPH as Completers are required to indicate TPH Completer capability via the Device Capabilities 2 register. TPH is architected to be applied for transactions that target Memory Space, and is applicable for transaction flows between device-to-host, device-to-device and host-to-device. In each case for TPH to be supported, the Requester, Completer, and all intermediate routing elements must support the associated TPH capabilities.

Software discovers the Requester capabilities via the TPH Requester Capability structure and Completer capabilities via the Device Capabilities 2 register (see Section 7.5.3.15). Software must program the TPH Requester Enable field in the TPH Requester Capability structure to enable the Function to initiate Requests with TPH.

TPH only provides additional information to enable optimized processing of Requests that target Memory Space, so existing mechanisms and rules for managing Memory Space access (e.g., Bus Master Enable, Memory Space Enable, and Base Address registers) are not altered.

6.18 Latency Tolerance Reporting (LTR) Mechanism

The Latency Tolerance Reporting (LTR) mechanism enables Endpoints to report their service latency requirements for Memory Reads and Writes to the Root Complex, so that power management policies for central platform resources (such as main memory, RC internal interconnects, and snoop resources) can be implemented to consider Endpoint service requirements. The LTR Mechanism does not directly affect Link power management or Switch internal power management, although it is possible that indirect effects will occur.

The implications of "latency tolerance" will vary significantly between different device types and implementations. When implementing this mechanism, it will generally be desirable to consider if service latencies impact functionality or only performance, if performance impacts are linear, and how much it is possible for the device to use buffering and/or other techniques to compensate for latency sensitivities.

The Root Complex is not required to honor the requested service latencies, but is strongly encouraged to provide a worst case service latency that does not exceed the latencies indicated by the LTR mechanism.

LTR support is discovered and enabled through reporting and control registers described in Chapter 7. Software must not enable LTR in an Endpoint unless the Root Complex and all intermediate Switches indicate support for LTR. Note that it is not required that all Endpoints support LTR to permit enabling LTR in those Endpoints that do support it. When enabling the LTR mechanism in a hierarchy, devices closest to the Root Port must be enabled first.

If an LTR Message is received at a Downstream Port that does not support LTR or if LTR is not enabled, the Message must be treated as an Unsupported Request.

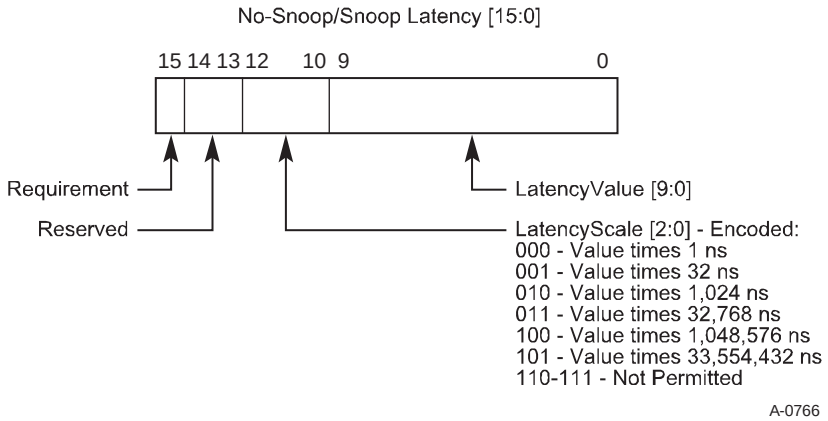


Figure 6-15: Latency Fields Format for LTR Messages

No-Snoop Latency and Snoop Latency: As shown in Figure 6-15, these fields include a Requirement bit that indicates if the device has a latency requirement for the given type of Request. If the Requirement bit is Set, the LatencyValue and LatencyScale fields describe the latency requirement. If the Requirement bit is Clear, there is no latency requirement and the LatencyValue and LatencyScale fields are ignored. With any LTR Message transmission, it is permitted for a device to indicate that a requirement is being reported for only no-snoop Requests, for only snoop Requests, or for both types of Requests. It is also permitted for a device to indicate that it has no requirement for either type of traffic, which it does by clearing the Requirement bit in both fields.

Each field also includes value and scale fields that encode the reported latency. Values are multiplied by the indicated scale to yield an absolute time value, expressible in a range from 1 ns to $2^{25}*(2^{10}-1)$ = 34,326,183,936 ns.

Setting the value and scale fields to all 0's indicates that the device will be impacted by any delay and that the best possible service is requested.

If a device doesn't implement or has no service requirements for a particular type of traffic, then it must have the Requirement bit clear for the associated latency field.

When directed to a non-D0 state by a Write to the PMCSR register, if a device's most recently transmitted LTR message (since the last DL_Down to DL_Up transition) reported one or both latency fields with any Requirement bit set, then it must send a new LTR Message with both of the Requirement bits clear prior to transitioning to the non-D0 state.

When the LTR Mechanism Enable bit is cleared, if a device's most recently sent LTR Message (since the last DL_DOWN to DL_Up transition) reported latency tolerance values with any Requirement bit set, then one of the following applies:

- If the bit was cleared due to a Configuration Write to the Device Control 2 register, the device must send a new LTR Message with all the Requirement bits clear.
- If the bit was cleared due to an FLR, it is strongly recommended that the device send a new LTR Message with all the Requirement bits clear.

When a Downstream Port goes to DL_Down status, any previous latencies recorded for that Port must be treated as invalid.

An LTR Message from a device reflects the tolerable latency from the perspective of the device, for which the platform must consider the service latency itself, plus the delay added by the use of Clock Power Management (CLKREQ#), if applicable. The service latency itself is defined as follows:

- When a device issues a Non-Posted Request, service latency of that Request is the delay from transmission of the last symbol of the Request TLP to the receipt of the first symbol of the first Completion TLP for that Request[110].
- When a device issues one or more Posted Requests such that it cannot issue another Posted Request due to Flow Control backpressure, service latency of the blocked Request is the delay from the transmission of the last symbol of the previous Posted Request to the receipt of the first symbol of the DLLP returning the credit(s) that allows transmission of the blocked Request[110].

If Clock Power Management is used, then the platform implementation-dependent period between when a device asserts CLKREQ# and the device receives a valid clock signal constitutes an additional component of the platform service latency that must be comprehended by the platform when setting platform power management policy.

It is recommended that Endpoints transmit an LTR Message Upstream shortly after LTR is enabled.

It is strongly recommended that Endpoints send no more than two LTR Messages within any 500 μs time period, except where required to by the specification. Downstream Ports must not generate an error if more than two LTR Messages are received within a 500 μs time period, and must properly handle all LTR messages regardless of the time interval between them.

Multi-Function Devices (MFDs) associated with an Upstream Port must transmit a "conglomerated" LTR Message Upstream according to the following rules:

- The acceptable latency values for the Message sent Upstream by the MFD must reflect the lowest values associated with any Function.
    - It is permitted that the snoop and no-snoop latencies reported in the conglomerated Message are associated with different Functions.
    - If none of the Functions report a requirement for a certain type of traffic (snoop/no-snoop), the Message sent by the MFD must not set the Requirement bit corresponding to that type of traffic.

[110] For this definition, all of the symbols of a DLLP or TLP are included. For 8b/10b, the first and last symbols are framing symbols (SDP, STP or END, see Section 4.2.1). For 128b/130b, the first symbol of a packet is the first symbol of a framing token (SDP or STP) and the last symbol of a packet is the last symbol of a CRC or LCRC (see Section 4.2.2).

- The MFD must transmit a new LTR Message Upstream when any Function of the MFD changes the values it has reported internally in such a way as to change the conglomerated value earlier reported by the MFD.

Switches must collect the Messages from Downstream Ports that have the LTR mechanism enabled and transmit a "conglomerated" Message Upstream according to the following rules:

- If a Switch supports the LTR feature, it must support the feature on its Upstream Port and all Downstream Ports.
- A Switch Upstream Port is permitted to transmit LTR Messages only when its LTR Mechanism Enable bit is Set or shortly after software clears its LTR Mechanism Enable bit as described earlier in this section.
- The acceptable latency values for the Message sent Upstream by the Switch must be calculated as follows:
  - If none of the Downstream Ports receive an LTR Message containing a requirement for a certain type of traffic (snoop/no-snoop), then any LTR Message sent by the Switch must not set the Requirement bit corresponding to that type of traffic.
  - Define LTRdnport[N] as the value reported in the LTR Message received at Downstream Port N, with these adjustments if applicable:
    - LTRdnport[N] is effectively infinite if the Requirement bit is clear or if a Not Permitted LatencyScale value is used
    - LTRdnport[N] must be 0 if the Requirement bit is 1 and the LatencyValue field is all 0's regardless of the LatencyScale value
  - Define LTRdnportMin as the minimum value of LTRdnport[N] across all Downstream Ports
  - Define Lswitch as all latency induced by the Switch
    - If Lswitch dynamically changes based on the Switch's operational mode, the Switch must not allow Lswitch to exceed 20% of LTRdnportMin, unless Lswitch for the Switch's lowest latency mode is greater, in which case the lowest latency state must be used
  - Calculate the value to transmit upstream, LTRconglomerated, as LTRdnportMin – Lswitch, unless this value is less than 0 in which case LTRconglomerated is 0
  - If LTRconglomerated is 0, both the LatencyValue and LatencyScale fields must be all 0's in the conglomerated LTR message
- A new LTR message must be transmitted Upstream if the conglomerated latencies are changed as a result of DL_Down invalidating the previous latencies recorded for that Port.
- If a Switch Downstream Port has the LTR Mechanism Enable bit cleared, the Latency Tolerance values recorded for that Port must be treated as invalid, and the latencies to be transmitted Upstream updated and a new conglomerated Message transmitted Upstream if the conglomerated latencies are changed as a result.

- A Switch must transmit an LTR Message Upstream when any Downstream Port/Function changes the latencies it has reported in such a way as to change the conglomerated latency reported by the Switch.
- A Switch must not transmit LTR Messages Upstream unless triggered to do so by one of the events described above.

The RC is permitted to delay processing of device Request TLPs provided it satisfies the device's service requirements.

When the latency requirement is updated during a series of Requests, it is required that the updated latency figure be comprehended by the RC prior to servicing a subsequent Request. In all cases the updated latency value must take effect within a time period equal to or less than the previously reported latency requirement. It is permitted for the RC to comprehend the updated latency figure earlier than this limit.

IMPLEMENTATION NOTE

Optimal Use of LTR

It is recommended that Endpoints transmit an updated LTR Message each time the Endpoint's service requirements change. If the latency tolerance is being reduced, it is recommended to transmit the updated LTR Message ahead of the first anticipated Request with the new requirement, allowing the amount of time indicated in the previously issued LTR Message. If the tolerance is being increased, then the update should immediately follow the final Request with the preceding latency tolerance value.

Typically, the Link will be in ASPM L1, and, if Clock Power Management (Clock PM) is supported, CLKREQ# will be deasserted, at the time an Endpoint reaches an internal trigger that causes the Endpoint to initiate Requests to the RC. The following text shows an example of how LTR is applied in such a case. Key time points are illustrated in Figure 6-16.

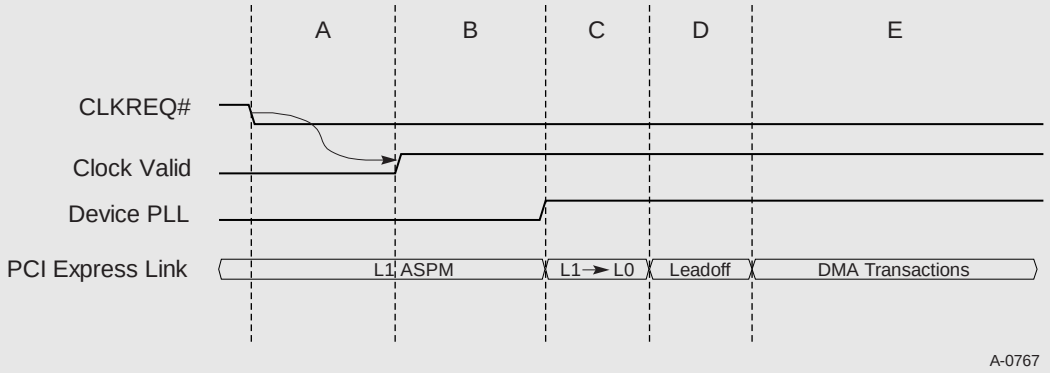


Figure 6-16: CLKREQ# and Clock Power Management

Time A is a platform implementation-dependent period between when a device asserts CLKREQ# and the device receives a valid clock signal. This value will not exceed the latency in effect.

Time B is the device implementation-dependent period between when a device has a valid clock and it can initiate the retraining sequence to transition from L1 ASPM to L0.

Time C is the period during which the transition from L1 ASPM to L0 takes place

Time D for a Read transaction is the time between the transmission of the END symbol in the Request TLP to the receipt of the STP symbol in the Completion TLP for that Request. Time D for a Write transaction is the time between the transmission of the END symbol of the TLP that exhausts the FC credits to the receipt of the SDP symbol in the DLLP returning more credits for that Request type. This value will not exceed the latency in effect.

Time E is the period where the data path from the Endpoint to system memory is open, and data transactions are not subject to the leadoff latency.

The LTR latency semantic reflects the tolerable latency seen by the device as measured by one or both of the following:

Case 1: the device may or may not support Clock PM, but has not deasserted its CLKREQ# signal – The latency observed by the device is represented in Figure 6-16 as the sum of times C and D.

Case 2: the device supports Clock PM and has deasserted CLKREQ#- The latency observed by the device is represented as the sum of times A, C, and D.

To effectively use the LTR mechanism in conjunction with Clock PM, the device will know or be able to measure times B and C, so that it knows when to assert CLKREQ#. The actual values of Time A, Time C, and Time D may vary dynamically, and it is the responsibility of the platform to ensure the sum will not exceed the latency.

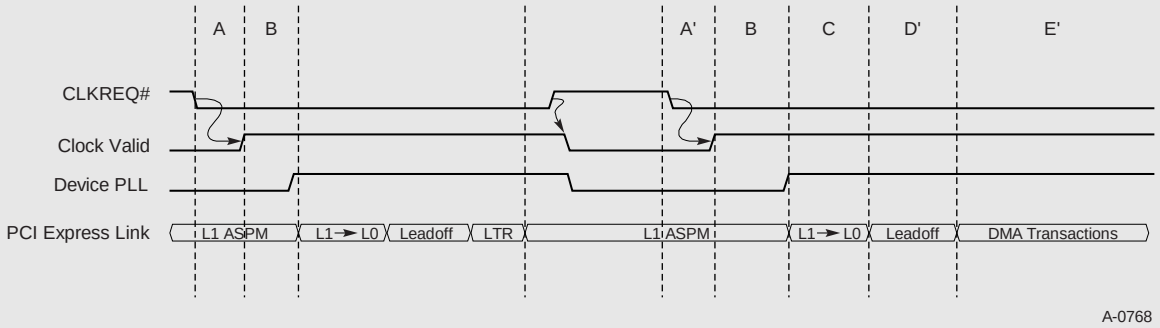


Figure 6-17: Use of LTR and Clock Power Management

In a very simple model, an Endpoint may choose to implement LTR as shown in Figure 6-17. When an Endpoint determines that it is idle, it sends an LTR Message with the software configured maximum latency or the maximum latency the Endpoint can support.

When the Endpoint determines that it has a need to maintain sustained data transfers with the Root Complex, the Endpoint sends a new LTR Message with a shorter latency (at Time E). This LTR Message is sent prior to the next data flush by a time equal to the maximum latency sent before (the time between Time E and Time D'). In between Time E and Time A', the Endpoint can return to a low power state, while the platform transitions to a state where it can provide the shorter latency when the device next needs to transmit data.

Note that the RC may delay processing of device Request TLPs, provided it satisfies the device's service requirements. If, for example, an Endpoint connected to Root Port 1 reports a latency tolerance of 100 µs, and an Endpoint on Root Port 2 report a value of 30 µs, the RC might implement a policy of stalling an initial Request following an idle period from Root Port 1 for 70 µs before servicing the Request with a 30 µs latency, thus providing a perceived service latency to the first Endpoint of 100 µs. This RC behavior provides the RC the ability to batch together Requests for more efficient servicing.

It is possible that, after it is determined that the RC can service snoop and no-snoop Requests from all Endpoints within the maximum snoop and maximum no-snoop time intervals, this information may be communicated to Endpoints by updating the Max Snoop LatencyValue, Max Snoop LatencyScale and Max No-Snoop LatencyValue, Max No-Snoop LatencyScale fields. The intention of this communication would be to prevent Endpoints from sending needless LTR updates.

When an Endpoint's LTR value for snoop Requests changes to become larger (looser) than the value indicated in the Max Snoop LatencyValue/Scale fields, it is recommended that the Endpoint send an LTR message with the snoop LTR value indicated in the Max Snoop LatencyValue/Scale fields. Likewise, when an Endpoint's LTR value for no-snoop Requests changes to become larger (looser) than the value indicated in the Max No-Snoop LatencyValue/Scale fields, it is

recommended that the Endpoint send an LTR message with the no-snoop LTR value indicated in the Max No-Snoop LatencyValue/Scale fields.

It is recommended that Endpoints buffer Requests as much as possible, and then use the full Link bandwidth in bursts that are as long as the Endpoint can practically support, as this will generally lead to the best overall platform power efficiency.

Note that LTR may be enabled in environments where not all Endpoints support LTR, and in such environments, Endpoints that do not support LTR may experience suboptimal service.

6.19 Optimized Buffer Flush/Fill (OBFF) Mechanism

The Optimized Buffer Flush/Fill (OBFF) Mechanism enables a Root Complex to report to Endpoints (throughout a hierarchy) time windows when the incremental platform power cost for Endpoint bus mastering and/or interrupt activity is relatively low. Typically this will correspond to time that the host CPU(s), memory, and other central resources associated with the Root Complex are active to service some other activity, for example the operating system timer tick. The nature and determination of such windows is platform/implementation specific.

An OBFF indication is a hint - Functions are still permitted to initiate bus mastering and/or interrupt traffic whenever enabled to do so, although this will not be optimal for platform power and should be avoided as much as possible.

OBFF is indicated using either of the WAKE# signal or a message (see Section 2.2.8.9). The message is to be used exclusively on interconnects where the WAKE# signal is not available. WAKE# signaling of OBFF or CPU Active must only be initiated by a Root Port when the system is in an operational state, which in an ACPI compliant system corresponds to the S0 state. Functions that are in a non-D0 state must not respond to OBFF or CPU Active signaling.

The OBFF message routing is defined as 100b, for point-to-point, and is only permitted to be transmitted in the Downstream direction. There are multiple OBFF events distinguished. When using the OBFF Message, the OBFF Code field is used to distinguish between different OBFF cases:

- 1111b "CPU Active" – System fully active for all Device actions including bus mastering and interrupts
- 0001b "OBFF" – System memory path available for Device memory read/write bus master activities
- 0000b "Idle" – System in an idle, low power state

All other codes are Reserved.

These codes correspond to various assertion patterns of WAKE# when using WAKE# signaling, as shown in Figure 6-18. There is one negative-going transition when signaling OBFF and two negative going transitions each time CPU Active is signaled. The electrical parameters required when using WAKE# are defined in the WAKE# Signaling section of *PCI Express CEM Specification, Revision 2.0* (or later).

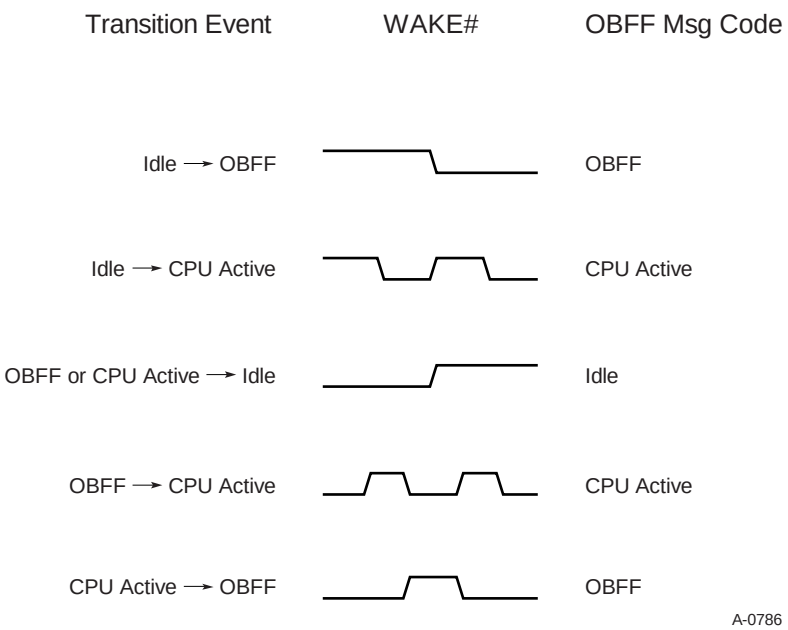


Figure 6-18: Codes and Equivalent WAKE# Patterns

When an OBFF Message is received that indicates a Reserved code, the Receiver, if OBFF is enabled, must treat the indication as a "CPU Active" indication.

An OBFF Message received at a Port that does not implement OBFF or when OBFF is not enabled must be handled as an Unsupported Request (UR). This is a reported error associated with the receiving Port (see Section 6.2). If a Port has OBFF enabled using WAKE# signaling, and that Port receives an OBFF Message, the behavior is undefined.

OBFF indications reflect central resource power management state transitions, and are signaled using WAKE# when this is supported by the platform topology, or using a Message when WAKE# is not available. OBFF support is discovered and enabled through reporting and control registers described in Chapter 7. Software must not enable OBFF in an Endpoint unless the platform supports delivering OBFF indications to that Endpoint.

When the platform indicates the start of a CPU Active or OBFF window, it is recommended that the platform not return to the Idle state in less than 10 µs. It is permitted to indicate a return to Idle in advance of actually entering platform idle, but it is strongly recommended that this only be done to prevent late Endpoint activity from causing an immediate exit from the idle state, and that the advance time be as short as possible.

It is recommended that Endpoints not assume CPU Active or OBFF windows will remain open for any particular length of time.

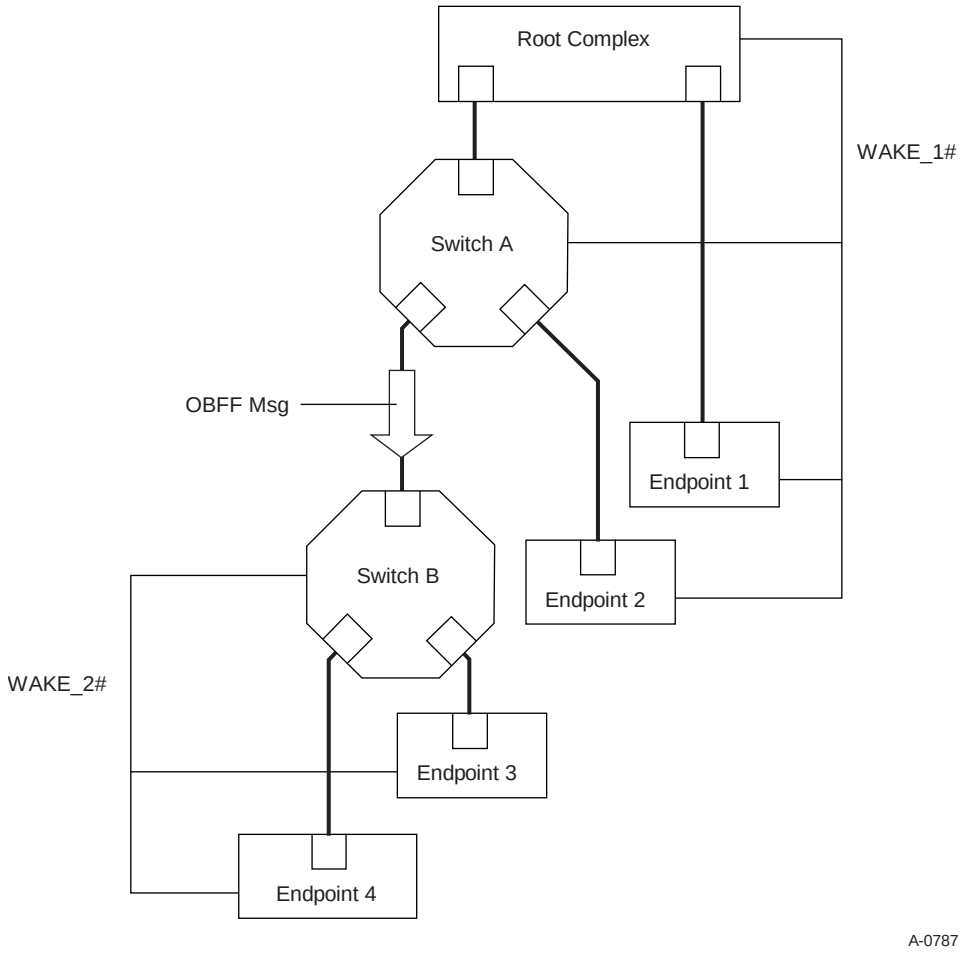


Figure 6-19: Example Platform Topology Showing a Link Where OBFF is Carried by Messages

Figure 6-19 shows an example system where it is necessary for a Switch (A) to translate OBFF indications received using WAKE# into OBFF Messages, which in this case are received by another Switch (B) and translated back to using WAKE# signaling. A HwInit configuration mechanism (set by hardware or firmware) is used to identify cases such as shown in this example (where the link between Switch A and Switch B requires the use of OBFF Messages), and system firmware/software must configure OBFF accordingly.

When a Switch is configured to use OBFF Message signaling at its Upstream Port and WAKE# at one or more Downstream Ports, or vice-versa, when enabled for OBFF, the Switch is required to convert all OBFF indications received at the Upstream Port into the appropriate form at the Downstream Port(s).

When using WAKE#, the enable for any specific Root Port enables the global use of WAKE# unless there are multiple WAKE# signals, in which case only the associated WAKE# signals are affected. When using Message signaling for OBFF, the enable for a particular Root Port enables transmission of OBFF messages from that Root Port only. To ensure OBFF is fully enabled in a platform, all Root Ports indicating OBFF support must be enabled for OBFF. It is permitted for system firmware/software to selectively enable OBFF, but such enabling is beyond the scope of this specification.

To minimize power consumption, system firmware/software is strongly recommended to enable Message signaling of OBFF only when WAKE# signaling is not available for a given link.

OBFF signaling using WAKE# must only be reported as supported by all components connected to a Switch if it is a shared WAKE# signal. In these topologies it is permitted for software to enable OBFF for components connected to the Switch even if the Switch itself does not support OBFF.

It is permitted, although not encouraged, to indicate the same OBFF event more than once in succession.

When a Switch is propagating OBFF indications Downstream, it is strongly encouraged to propagate all OBFF indications. However, especially when using Messages, it may be necessary for the Switch to discard or collapse OBFF indications. It is permitted to discard and replace an earlier indication of a given type when an indication of the same or a different type is received.

Downstream Ports can be configured to transmit OBFF Messages in two ways, which are referred to as Variation A and Variation B. For Variation A, the Port must transmit the OBFF Message if the Link is in the L0 state, but discard the Message when the Link is in the Tx_L0s or L1 state. This variation is preferred when the Downstream Port leads to Devices that are expected to have communication requirements that are not time-critical, and where Devices are expected to signal a non-urgent need for attention by returning the Link state to L0. For Variation B, the Port must transmit the OBFF Message if the Link is in the L0 state, or, if the Link is in the Tx_L0s or L1 state, it must direct the Link to the L0 state and then transmit the OBFF Message. This variation is preferred when the Downstream Port leads to devices that can benefit from timely notification of the platform state.

When initially configured for OBFF operation, the initial assumed indication must be the CPU Active state, regardless of the logical value of the WAKE# signal, until the first transition is observed.

When enabling Ports for OBFF, it is recommended that all Upstream Ports be enabled before Downstream Ports, and Root Ports must be enabled after all other Ports have been enabled. For hot pluggable Ports this sequence will not generally be possible, and it is permissible to enable OBFF using WAKE# to an unconnected hot pluggable Downstream Port. It is recommended that unconnected hot pluggable Downstream Ports not be enabled for OBFF message transmission.

IMPLEMENTATION NOTE

OBFF Considerations for Endpoints

It is possible that during normal circumstances, events could legally occur that could cause an Endpoint to misinterpret transitions from an Idle window to a CPU Active window or OBFF window. For example, a non-OBFF Endpoint could assert WAKE# as a wakeup mechanism, masking the system's transitions of the signal. This could cause the Endpoint to behave in a manner that would be less than optimal for power or performance reasons, but should not be unrecoverable for the Endpoint or the host system.

In order to allow an Endpoint to maintain the most accurate possible view of the host state, it is recommended that the Endpoint place its internal state tracking logic in the CPU Active state when it receives a request that it determines to be host-initiated, and at any point where the Endpoint has a pending interrupt serviced by host software.

6.20 PASID TLP Prefix

The PASID TLP Prefix is an End-End TLP Prefix as defined in Section 2.2.1. Layout of the PASID TLP Prefix is shown in Figure 6-20 and Table 6-13.

When a PASID TLP Prefix is present, the PASID value in the prefix, in conjunction with the requester ID, identifies the Process Address Space ID associated with the Request. Each Function has a distinct set of PASID values. PASID values used by one Function are unrelated to PASID values used by any other Function.

A PASID TLP Prefix is permitted on:

- Memory Requests (including AtomicOp Requests) with Untranslated Addresses (see Section 2.2.4.1).
- Address Translation Requests, ATS Invalidation Messages, Page Request Messages, and PRG Response Messages (see Section 10.1.3).

The PASID TLP Prefix is not permitted on any other TLP.

6.20.1 Managing PASID TLP Prefix Usage

Usage of PASID TLP Prefixes must be specifically enabled. Unless enabled, a component is not permitted to transmit a PASID TLP Prefix.

For Endpoint Functions (including Root Complex Integrated Devices), the following rules apply:

- A Function is not permitted to send and receive TLPs with a PASID TLP Prefix unless PASID Enable is Set (see Section 7.8.8.3):
- A Function must have a mechanism for dynamically associating use of a PASID with a particular Function context. This mechanism is device specific.
- A Function must have a mechanism to request that it gracefully stop using a specific PASID. This mechanism is device specific but must satisfy the following rules:
    - A Function may support a limited number of simultaneous PASID stop requests. Software should defer issuing new stop requests until older stop requests have completed.
    - A stop request in one Function must not affect operation of any other Function.
    - A stop request must not affect operation of any other PASID within the Function.
    - A stop request must not affect operation of transactions that are not associated with a PASID.
    - When the stop request mechanism indicates completion, the Function has:
        - Stopped queuing new Requests for this PASID.
        - Completed all Non-Posted Requests associated with this PASID.
        - Flushed to the host all Posted Requests addressing host memory in all TCs that were used by the PASID. The mechanism used for this is device specific (for example: a non-relaxed Posted Write to host memory or a processor read of the

Function can flush TC0; a zero length read to host memory can flush non-zero TCs).

- Optionally flushed all Peer-to-Peer Posted Requests to their destination(s). The mechanism used for this is device specific.
- Complied with additional rules described in Address Translation Services (Chapter 10) if Address Translations or Page Requests were issued on the behalf of this PASID.

For Root Complexes, the following rules apply:

- ❑ A Root Complex must have a device specific mechanism for indicating support for PASID TLP Prefixes.
- ❑ A Root Complex that supports PASID TLP Prefixes must have a device specific mechanism for enabling them. By default usage of PASID TLP Prefixes is disabled.
- ❑ A Root Complex that supports PASID TLP Prefixes may optionally have a device specific mechanism for enabling them at a finer granularity than the entire Root Complex (e.g., distinct enables for a specific Root Port, Requester ID, Bus Number, Requester ID, or Requester ID/PASID combination).

6.20.2 PASID TLP Layout

A TLP may contain at most one PASID TLP Prefix.

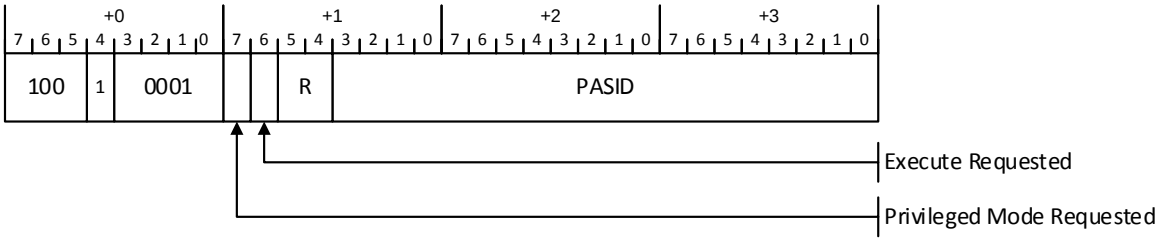


Figure 6-20: PASID TLP Prefix:

Table 6-13: PASID TLP Prefix

Bits	Description
Byte 0 bits 7:5	100b – indicating TLP Prefix
Byte 0 bit 4	1b – indicating End-End TLP Prefix
Byte 0 bits 3:0	0001b – indicating PASID TLP Prefix
Byte 1 bit 7	**Privileged Mode Requested** – If Set indicates a Privileged Mode entity in the Endpoint is issuing the Request, If Clear, indicates a Non-Privileged Mode entity in the Endpoint is issuing the Request. Usage of this bit is specified in Section 6.20.2.3.
Byte 1 bit 6	**Execute Requested** – If Set indicates the Endpoint is requesting Execute Permission. If Clear, indicates the Endpoint is not requesting Execute Permission. Usage of this bit is specified in Section 6.20.2.2.

Bits	Description
Byte 1 bit 3: Byte3 bit 0	**Process Address Space ID (PASID)** – This field contains the PASID value associated with the TLP. Usage of this field is defined in Section 6.20.2.1.

6.20.2.1 *PASID field*

The PASID field identifies the user process associated with a Request. This field is present in all PASID TLP Prefixes.

The PASID field is 20 bits wide. Endpoints and Root Complexes need not support the entire range of the field. For Endpoints, the Max PASID Width field indicates the supported range of PASID values (Section 7.28.2). For Root Complexes, an implementation specific mechanism is used to provide this information.

Endpoints are not permitted to send TLPs with a PASID TLP Prefix unless the PASID Enable bit (Section 7.28.3) is Set. Endpoints that support the PASID TLP Prefix must signal Unsupported Request (UR) when they receive a TLP with a PASID TLP Prefix and the PASID Enable bit is Clear.

Root Complexes may optionally support TLPs with PASID TLP Prefixes. The mechanism used to detect whether a Root Complex supports the PASID TLP Prefix is implementation specific.

For Endpoints, the following rules apply:

- The Endpoint is not permitted to send TLPs with a PASID value greater than or equal to $2^{\text{Max PASID Width}}$.
- The Endpoint is optionally permitted to signal an error when it receives a Request with a PASID value greater than or equal to $2^{\text{Max PASID Width}}$. This is an Unsupported Request error associated with the Receiving Port (see Section 6.2).

For Root Complexes, the following rules apply:

- A Root Complex is not permitted to send a TLP with a PASID value greater than it supports.
- A Root Complex is optionally permitted to signal an error when it receives a Request with a PASID value greater than it supports. This is an Unsupported Request error associated with the Receiving Port (see Section 6.2).

For Completers, the following rules apply:

- For Untranslated Memory Requests, the PASID value and the Untranslated Address are both used in determining the Translated Address used in satisfying the Request.

For address translation related TLPs, usage of this field is defined in Address Translation Services (Chapter 10).

IMPLEMENTATION NOTE

PASID Width Homogeneity

The PASID value is unique per Function and thus the original intent was that the width of the

PASID value supported by that Function could be based on the needs of that Function. However, current system software typically does not follow that model and instead uses the same PASID value in all Functions that access a specific address space. To enable this, system software will typically ensure a common system PASID width for Root Complex and persistent translation agents. Such system software will typically disable ATS on any hot plugged Endpoint Functions or translation agents reporting PASID width support which is less than that of the common system PASID width. The Root Complex, Endpoints, and translation agents, are often implemented independently of system software, therefore it is highly recommended that hardware implement the maximum width of 20 bits to ensure interoperability with system software.
Endpoints may, in an implementation-specific way, be able to map the 20 bit system PASID to an internal representation carrying a smaller width. If this is done, it is critical that the Endpoint do so without impacting system software, which has no mechanism to differentiate such implementation from those that implement the full 20 bit width natively.

6.20.2.2 *Execute Requested*

If the Execute Requested bit is Set, the Endpoint is requesting permission for the Endpoint to Execute instructions in the memory range associated with this request. The meaning of Execute permission is outside the scope of this specification.

Endpoints are not permitted to send TLPs with the Execute Requested bit Set unless the Execute Permission Supported bit (Section 7.8.8.2) and the Execute Permission Enable bit (Section 7.8.8.3) are both Set.

For Root Complexes, the following rules apply:

- Support for Execute Requested by the Root Complex is optional. The mechanism used to determine whether a Root Complex supports Execute Requested is implementation specific.
- A Root Complex that supports the Execute Requested bit should have an implementation specific mechanism to enable it to use the bit.
- A Root Complex that supports the Execute Requested bit may have an implementation specific mechanism to enable use of the bit at a finer granularity (e.g., for a specific Root Port, for a specific Bus Number, for a specific Requester ID, or for a specific Requester ID/PASID combination), and its default value is implementation specific.

For Completers, the following rules apply:

- Completers have a concept of an effective value of the bit. For a given Request, if the Execute Requested bit is supported and it usage is enabled for the Request, the effective value of the bit is the value in the Request; otherwise the effective value of the bit is 0b.
- For Untranslated Memory Read Requests, Completers use the effective value of the bit as part of the protection check. If this protection check fails, Completers treat the Request as if the memory was not mapped.
- For Untranslated Memory Requests, other than an Untranslated Memory Read Request, the bit is Reserved.

For address translation related TLPs, usage of this bit is defined in Address Translation Services (Chapter 10).

6.20.2.3 *Privileged Mode Requested*

If Privileged Mode Requested is Set, the Endpoint is issuing a Request that targets memory associated with Privileged Mode. If Privileged Mode Requested is Clear, the Endpoint is issuing a Request that targets memory associated with Non-Privileged Mode.

The meaning of Privileged Mode and Non-Privileged Mode and what it means for an Endpoint to be operating in Privileged or Non-Privileged Mode depends on the protection model of the system and is outside the scope of this specification.

Endpoints are not permitted to send a TLP with the Privileged Mode Requested bit Set unless both the Privileged Mode Supported bit (Section 7.8.8.2) and the Privileged Mode Enable bit (Section 7.8.8.3) are Set.

For Root Complexes, the following rules apply:

- Support for the Privileged Mode Requested bit by the Root Complex is optional. The mechanism used to determine whether a Root Complex supports the Privileged Mode Requested bit is implementation specific.
- A Root Complex that supports the Privileged Mode Requested bit should have an implementation specific mechanism to enable it to use the bit.
- A Root Complex that supports the Privileged Mode Requested bit may have an implementation specific mechanism to enable use of the bit at a finer granularity (e.g., for a specific Root Port, for a specific Bus Number, for a specific Requester ID, or for a specific Requester ID/PASID combination).

For Completers, the following rules apply:

- Completers have the concept of an effective value of the bit. For a given Request, if the Privileged Mode Requested bit is supported and its usage is enabled for the Request, the effective value of the bit is the value in the Request; otherwise the effective value of the bit is the 0b.
- For Untranslated Memory Requests, Completers use the effective value of the bit as part of its protection check. If this protection check fails, Completers treat the Request as if the memory was not mapped.
- For address translation related TLPs, usage of this bit is defined in Address Translation Services (Chapter 10).

6.21 Lightweight Notification (LN) Protocol

Lightweight Notification (LN) protocol enables Endpoints to register interest in specific cachelines in host memory, and be notified via a hardware mechanism when they are updated.

An LN Requester (LNR) is a client subsystem in an Endpoint that sends LN Read/Write Requests and receives LN Messages. An LN Completer (LNC) is a service subsystem in the host that receives LN Read/Write Requests, and sends LN Messages when registered cachelines are updated.

LN protocol provides a notification service for when cachelines of interest are updated. Most commonly, an LNR sends an LN Read to a Memory Space range that has an associated LNC, requesting a copy of a cacheline (abbreviated "line" in this section). The LNC returns the requested line to the LNR and records that the LNR has requested notification when that line is updated; that is, the LNC "registers" the line. Later, the LNC notifies the LNR via an LN Message when some entity updates the line, so the LNR can take appropriate action. A similar notification service exists for LN Writes, where an LNR writing a line can be notified later when the line is updated.

LN protocol permits multiple LNRs each to register the same line concurrently. An Endpoint with LNR is permitted to write to a line at any time, regardless of whether the LNR has registered the line.

A typical system consists of host processors, host memory, a host internal fabric, Root Ports, Switches, and Endpoints. Figure 6-21 below illustrates a simplified view of the elements and how they are interconnected to provide a context for describing how LN protocol operates.

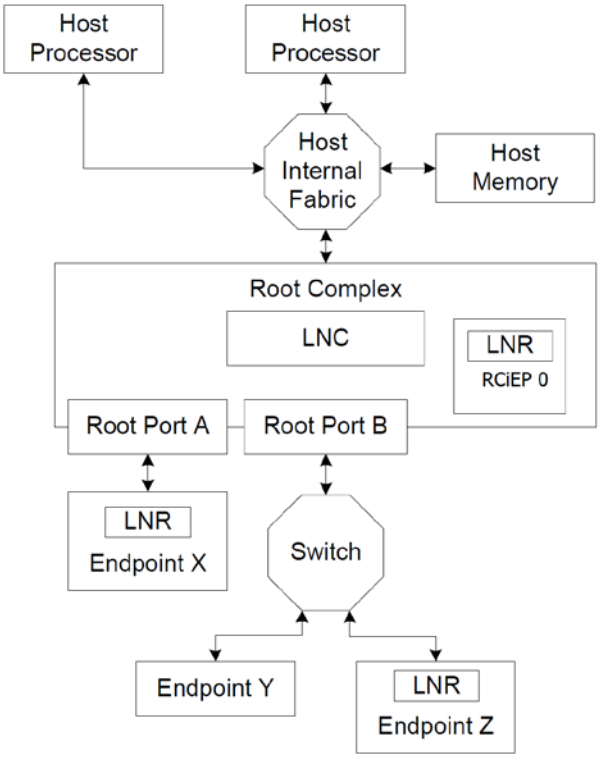


Figure 6-21: Sample System Block Diagram

In the figure above, Endpoint X, Endpoint Z, and RCIEP 0 each contain an LNR. The Root Complex contains an LNC.

6.21.1 LN Protocol Operation

LN is a simple protocol that satisfies several key use models with minimum complexity and cost.

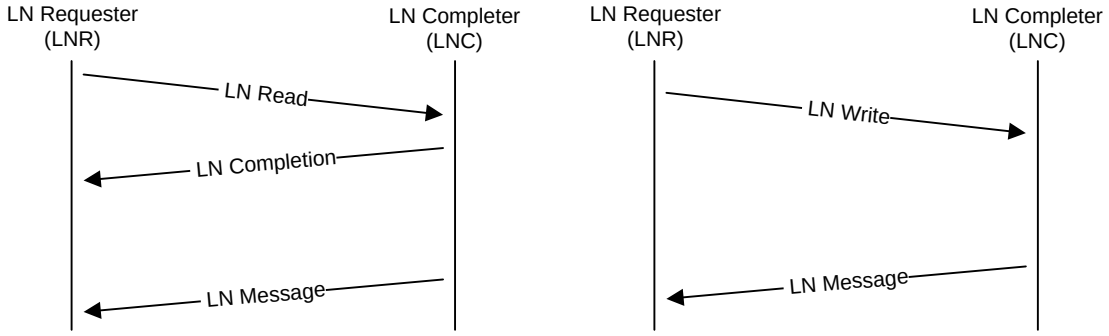


Figure 6-22: LN Protocol Basic Operation

LN protocol operation with a single LNR is shown in the diagram above. For the case of reads, as shown on the left: (1) an LNR requests a copy of a line from host memory using an LN Read; (2) the LNC returns the line in an LN Completion and records that the LNR has registered the line; and (3) the LNC later notifies the LNR using an LN Message when the registered line has been updated. For the case of writes, as shown on the right: (1) an LNR writes to a line to host memory using an LN Write; (2) the LNC records that the LNR has registered the line; and (3) the LNC later notifies the LNR using an LN Message when the registered line has been updated.

An LNC must send an LN Message either if a registered line is updated by some entity (e.g., a CPU or a device) or if the LNC is no longer going to track the state of that line. The latter case is referred to as an eviction, and is indicated by the Notification Reason (NR) field in the LN Message.

If an LN Requester does an LN Read or LN Write to a line that it already has registered, then the LN Requester generally can't determine if a subsequent LN Message it receives for that line was for the most recent LN Read/Write Request or for a previous one.

LN protocol permits multiple LNRs to register the same line concurrently. In this case the LNC notifies the multiple LNRs either by sending a directed LN Message to each LNR, or by sending a broadcast LN Message to each affected Hierarchy Domain.

Once the LNC updates or evicts a given line and sends one or more LN Messages to notify all LNRs with associated registrations, the LNC will not send additional LN Messages for that specific line unless it receives a new LN Read or LN Write to that line.

An LNC is permitted to have an implementation-specific limit on how many LNRs it can independently track for each given line. At or below this limit, the LNC generally uses directed LN Messages for notifications. Above this limit, the LNC uses broadcast LN Messages. An implementation is permitted to have a limit of zero, and use broadcast LN Messages exclusively.

A single LN Message with a specific NR field value can indicate that all lines registered to that LNR have been evicted. Both directed and broadcast versions of this "all evicted" LN Message are permitted.

An LNC is permitted to have an implementation-specific limitations on how many lines or sets of lines it can register concurrently. If the LNC receives a Request to register a new line, but the LNC has insufficient resources to do so, the LNC is permitted to evict one or more old lines in order to free up the necessary resources, or send a LN Message indicating that the new line was evicted.

IMPLEMENTATION NOTE

Excessive Use of Broadcast LN Messages

In order to avoid performance issues, LNCs that use broadcast LN Messages should be implemented to minimize the number of Hierarchy Domains each broadcast LN Message is sent to, and also to keep the rate of broadcast LN Messages within reasonable bounds. Each broadcast LN Message consumes Link bandwidth, and some Endpoints may process broadcast LN Messages at a relatively low rate.

In particular, Endpoints that do not support LN Requester capability may handle received broadcast LN Messages as an exception case using a low performance mechanism, e.g., decoding the Message with device firmware instead of in hardware. Each Message could conceivably take microseconds to process[111], and an excessive rate of them could easily cause back-pressure of Posted Requests within the fabric, causing severe performance problems.

While Endpoints that do not support LN Requester capability may also handle directed LN Messages as an exception case using a low performance mechanism, this should not cause performance problems. With LN protocol, LNCs send directed LN Messages only to Endpoints that support LN Requester capability, and presumably those Endpoints will be able to process both directed and broadcast LN Messages at a rate that avoids performance issues.

6.21.2 LN Registration Management

Since LNCs have limited resources for registering cachelines, an LNR Capability structure in each LNR provides mechanisms for software to limit the LNR's maximum number of outstanding registrations. Software reads the LNR Registration Max field to discover the maximum number an LNR is capable of having outstanding. If necessary, software can set the LNR Registration Limit field to impose a specified limit.

LNC registration resource limitations may be highly implementation specific, perhaps including factors such as set associativity, maximum number of sets, and distinct sets of resources for different regions of host memory. How software discovers these resource limitations is outside the scope of this specification.

To manage its number of outstanding registrations, an LNR can deregister outstanding registrations by sending a zero-length LN Write to each line it wants to deregister.

---

[111] The Posted Request Acceptance Limit permits an Endpoint to take up to 10 µs to process each received Posted Request. See Section 2.3.1.

The LNC is not required to accept registrations for all locations in host memory. However, the granularity for memory regions accepting registrations is required to be no finer than aligned 4KB regions. To determine if a given aligned 4KB region accepts registrations, an LNR can perform an LN Read to any cacheline within the region, and see if an LN Completion is returned. If an LNR wishes to "probe" a region for registration capability without actually creating a registration, the LNR can use a zero-length LN Read, which is required to provide this semantic.

6.21.3 LN Ordering Considerations

LN Reads have the same ordering requirements as other Memory Read Requests, and LN Writes have the same ordering requirements as other Memory Write Requests. LN Completions (for LN Reads) have the same ordering requirements as Completions for other Memory Reads. LN Messages have the same ordering requirements as other Posted Requests.

For a given line, when an LN Completer receives an LN Read followed by an update to that line, the LN Completer is permitted to send the LN Completion and the LN Message in either order.

For a given line, when an LN Completer receives an LN Read that triggers an eviction LN Message, the LN Completer is permitted to send the LN Completion and the LN Message in either order.

For a given line, when an LN Completer receives an LN Write that triggers an eviction LN Message, followed by an update to that line, the LN Completer is permitted to send the two LN Messages in either order.

For different lines, an LN Completer is permitted to send LN Messages in any order.

6.21.4 LN Software Configuration

LN protocol supports 2 cacheline sizes (CLSs) – 64 bytes and 128 bytes. Support for each CLS is optional, both for Root Complexes and Endpoints. The CLS in effect for the system is determined by the host, and indicated by the LN System CLS field in applicable Root Ports and RCRBs. All Root Ports and RCRBs that indicate LN Completer support must indicate the same CLS; otherwise, the results are undefined. The host must not change the system CLS while any operating system is running; otherwise, the results are undefined.

Endpoints supporting LN protocol must support one or both CLSs, and indicate this via the LNR-64 Supported and LNR-128 Supported capability bits. When enabling each LN Requester, software must ensure that the associated LNR CLS control bit is configured to match the system CLS; otherwise, the results are undefined. An LN Requester that supports only one CLS is permitted to hardwire its LNR CLS control bit to the corresponding value.

Software must not change the value of the LNR CLS control bit or the LNR Registration Limit field unless its LNR Enable control bit is already Clear or is being Cleared with the same Configuration Write; otherwise, the results are undefined. If the LNR Enable bit is already Clear, software is permitted to change the values of the LNR CLS bit and LNR Registration Limit field concurrently with Setting the LNR Enable bit, using a single Configuration Write.

Software is permitted to Clear the LNR Enable bit at any time. When the LNR Enable bit is Clear, the LNR must clear all its internal registration state.

6.21.5 LN Protocol Summary

Detailed rules and requirements for LN protocol are distributed throughout the rest of this specification, but here is a general summary plus some unique requirements.

- An LN Read is a Memory Read Request whose TLP Header has the LN bit Set. An LN Completion is a Completion whose TLP Header has the LN bit Set. An LN Write is a Memory Write Request whose TLP Header has the LN bit Set.
- All requirements for Memory Read Requests apply to LN Reads unless explicitly stated otherwise.
  - An LN Read must access no more than a single cacheline, as determined by the system CLS. An LN Completer must handle a violation of this rule as a Completer Abort unless the Completer detects a higher precedence error. Partial cacheline LN Reads, including zero-length LN Reads, are permitted.
  - If an LN Completer handles an LN Read as an Uncorrectable Error or an Advisory Non-Fatal Error, the LN Completer must not register notification service for that Request.
  - An LN Completer must handle a zero-length LN Read as a "probe" of the targeted memory region for registration capability. See Section 6.20.2. If the Completion Status is Successful Completion, the LN bit in the TLP Header must indicate if the region supports registration capability, but the LNC must not create a registration for this case. LN Completers must support registration capability with a granularity no finer than aligned 4KB regions.
  - Ordering and Flow Control rules for LN Reads are identical to those for Memory Read Requests.
- All requirements for Memory Read Completions apply to LN Completions unless explicitly stated otherwise.
  - The Completion for an LN Read must be an LN Completion (i.e., have the LN bit Set in its TLP Header) if the Completer is an LN Completer, the targeted memory region accepts registrations, and the Completion Status is Successful Completion; otherwise the Completion must have the LN bit Clear in its TLP Header. Note that a poisoned Completion will have a Completion Status of Successful Completion. See Section 2.7.2.2.
  - Ordering and Flow Control rules for LN Completions are identical to Completions whose TLP Header has the LN bit Clear.
- All requirements for Memory Write Requests apply to LN Writes unless explicitly stated otherwise.
  - An LN Write must access no more than a single cacheline, as determined by the system CLS. An LN Completer must handle a violation of this rule as a Completer Abort unless the Completer detects a higher precedence error.
  - If an LN Completer handles an LN Write as an Uncorrectable Error, the LN Completer must not register notification service for that Request. Note that depending upon its

configuration, a Completer may handle a poisoned LN Write as an Uncorrectable Error, an Advisory Non-Fatal Error, or a masked error.

- An LN Completer must handle a zero-length LN Write as a request to deregister an existing registration. See Section 6.20.2. If the cacheline targeted by a zero-length LN Write was not previously registered, it must remain unregistered.
- Ordering, Flow Control, and Data Poisoning rules for LN Writes are identical to those for Memory Write Requests.
- A Requester must not generate LN Writes for MSI or MSI-X interrupts. An LN Completer must handle an LN Write targeting an interrupt address range as a Completer Abort unless the Completer detects a higher precedence error.

- For LN Reads and LN Writes, the address must be the proper type, as indicated by the Address Type (AT) field. See Section 2.2.4.1. The proper type depends upon whether a Translation Agent (TA) is being used. See Chapter 10.
  - If a TA is being used, the address must be a Translated address. An LN Requester must support ATS in order to acquire and use Translated addresses.
  - If a TA is not being used, the address must be a Default/Untranslated Address.
  - An LN Completer detecting a violation of the above rules must handle the Request as a Completer Abort (CA), unless a higher precedence error is detected.

6.22 Precision Time Measurement (PTM) Mechanism

6.22.1 Introduction

Precision Time Measurement (PTM) enables precise coordination of events across multiple components with independent local time clocks. Ordinarily, such precise coordination would be difficult given that individual time clocks have differing notions of the value and rate of change of time. To work around this limitation, PTM enables components to calculate the relationship between their local times and a shared PTM Master Time: an independent time domain associated with a PTM Root.

PTM defines the following:

- PTM Requester – A Function capable of using PTM as a consumer associated with an Endpoint or an Upstream Port.
- PTM Responder – A Function capable of using PTM to supply PTM Master Time associated with a Port or an RCRB.
- Time Source – A local clock associated with a PTM Responder.
- PTM Root – The source of PTM Master Time for a PTM Hierarchy. A PTM Root must also be a Time Source and is typically also a PTM Responder.

Each PTM Root supplies PTM Master Time for a PTM Hierarchy: a set of PTM Requesters associated with a single PTM Root.

Figure 6-23. illustrates some example system topologies using PTM. These are only illustrative examples, and are not intended to imply any limits or requirements.

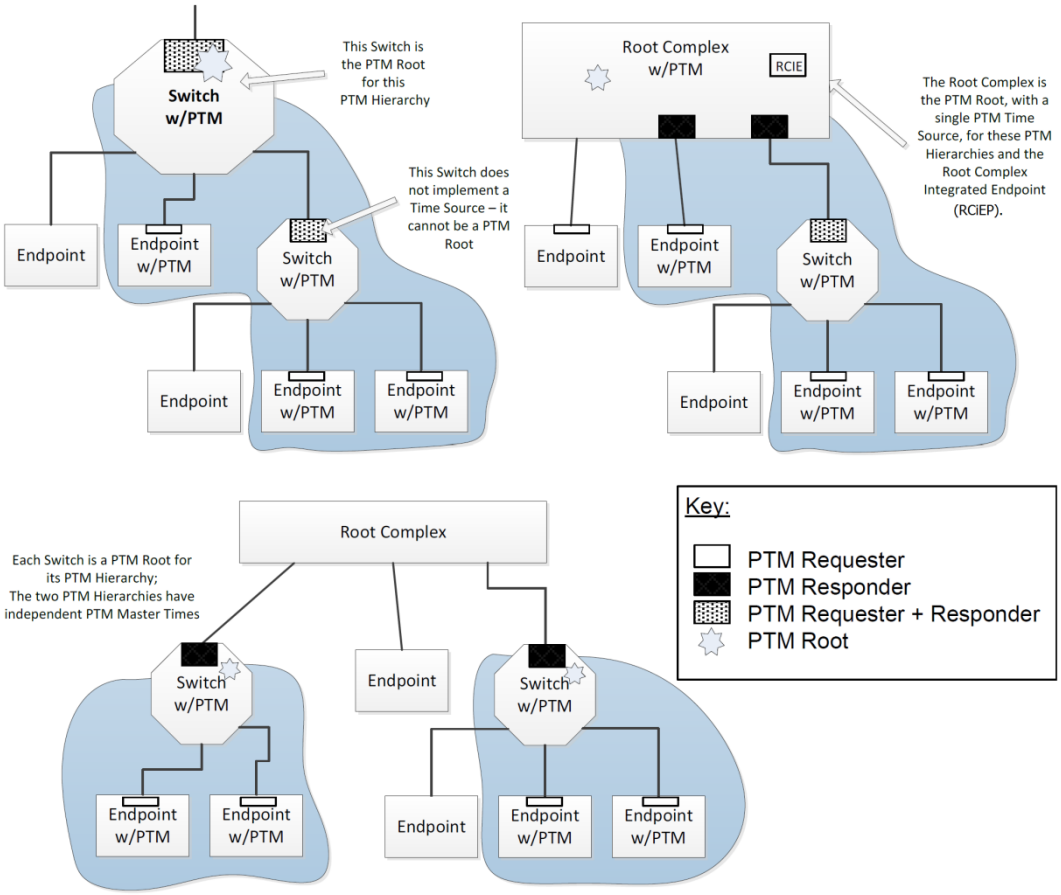


Figure 6-23: Example System Topologies using PTM

6.22.2 PTM Link Protocol

When using PTM between two components on a Link, the Upstream Port, which acts on behalf of the PTM Requester, sends PTM Requests to the Downstream Port on the same Link, which acts on behalf of the PTM Responder.

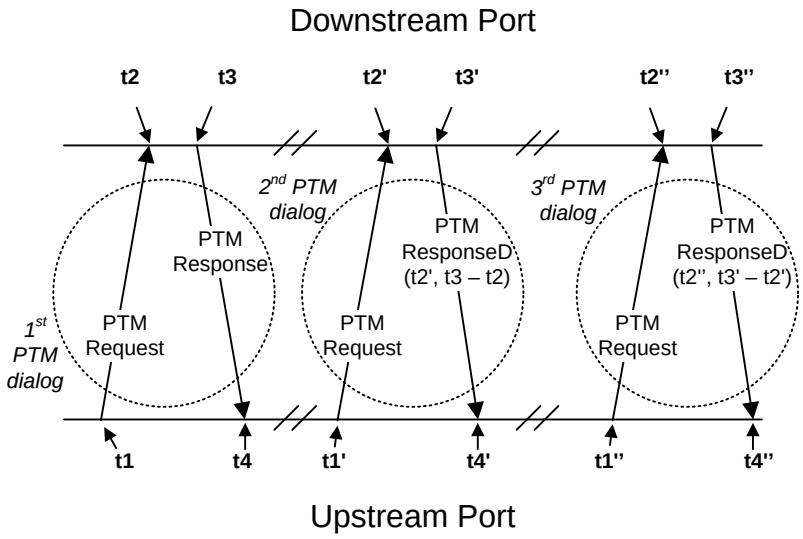


Figure 6-24: Precision Time Measurement Link Protocol

Figure 6-24 illustrates the PTM link protocol. The points t1, t2, t3, and t4 in the above diagram represent timestamps captured locally by each Port as they transmit and receive PTM Messages. The component associated with each Port stores these timestamps from the 1st PTM dialog in internal registers for use in the 2nd PTM dialog, and so on for subsequent PTM dialogs.

The Upstream Port, on behalf of the PTM Requester, initiates the PTM dialog by transmitting a PTM Request message.

The Downstream Port, on behalf of the PTM Responder, has knowledge of or access (directly or indirectly) to the PTM Master Time.

During each dialog, the Downstream Port populates the PTM ResponseD message based on timestamps stored during previous PTM dialogs, as defined in Section 6.22.3.2.

Once each component has historical timestamps from the preceding dialog, the component associated with the Upstream Port can combine its timestamps with those passed in the PTM ResponseD message to calculate the PTM Master Time using the following formula:

$$\text{PTM Master Time at t1'} = \text{t2'} - \frac{((\text{t4} - \text{t1}) - (\text{t3} - \text{t2}))}{2}$$

The values t1, t2, t3, t4, and t2' indicate the timestamps captured during the PTM dialog as illustrated in Figure 6-24.

PTM capable components would typically record the results of these timestamp calculations, and may make them available to software via implementation specific means. Herein, this document refers to this resultant timing information as the component's "PTM context".

For a Switch implementing PTM, the time synchronization mechanism(s) within the Switch itself are implementation-specific.

IMPLEMENTATION NOTE

PTM Theory and Operation

The timestamps captured during the PTM dialogs enable the calculation of the timing relationship between the PTM Requester and PTM Responder. The value (t3-t2) measures the time consumed by the PTM Responder for a given PTM dialog. The time (t4-t1) is the time from request to response. Therefore ((t4 – t1) – (t3 – t2)) effectively gives the round trip message transit time between the two components, and that quantity divided by 2 approximates the Link delay - the time difference between t1 and t2. It is assumed that the Link transit times from PTM Requester to PTM Responder and back again are symmetric, which is typically a good assumption (see also the Implementation Note on PTM Timestamp Capture Mechanisms).

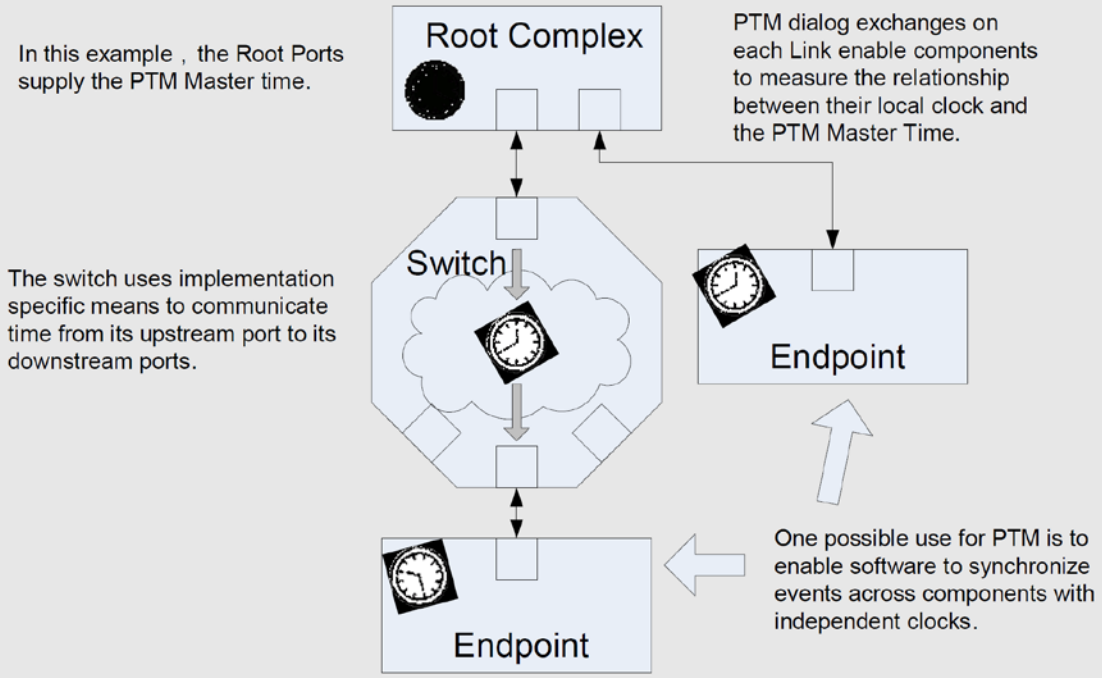


Figure 6-25: Precision Time Measurement Example

Figure 6-25 illustrates a simple device hierarchy employing PTM. Each Upstream Port initiates PTM dialogs to establish the relationship between its local time and the PTM Master Time provided by the Root Port.

In this example, the Switch initiates PTM dialogs on its Upstream Port to obtain the PTM Master Time for use in fulfilling PTM Request Message received at its Downstream Ports. This Switch employs implementation specific means to communicate the PTM Master Time from its Upstream Port to its Downstream Ports.

PTM capable components can make their PTM context available for inspection by software, enabling software to translate timing information between local times and PTM Master Time. In

turn, this capability enables software to coordinate events across multiple components with very fine precision.

Similarly, it is strongly recommended that platforms implementing PTM also make the PTM Master Time available to software.

6.22.3 Configuration and Operational Requirements

Software must not have the PTM Enable bit Set in the PTM Control register on a Function associated with an Upstream Port unless the associated Downstream Port on the Link already has the PTM Enable bit Set in its associated PTM Control register.

PTM support by a Function is indicated by the presence of a PTM Extended Capability structure. It is not required that all Endpoints in a hierarchy support PTM, and it is not required for software to enable PTM in all Endpoints that do support it.

If a PTM Message is received by a Port that does not support PTM, or by a Downstream Port when the PTM Enable bit is clear, the Message must be treated as an Unsupported Request. This is a reported error associated with the Receiving Port (see Section 6.2). A Properly formed PTM Response received by an Upstream Ports that support PTM, but for which the PTM Enable bit is clear, must be silently discarded.

As observed through PTM, the PTM Master Time must satisfy the following behavioral requirements:

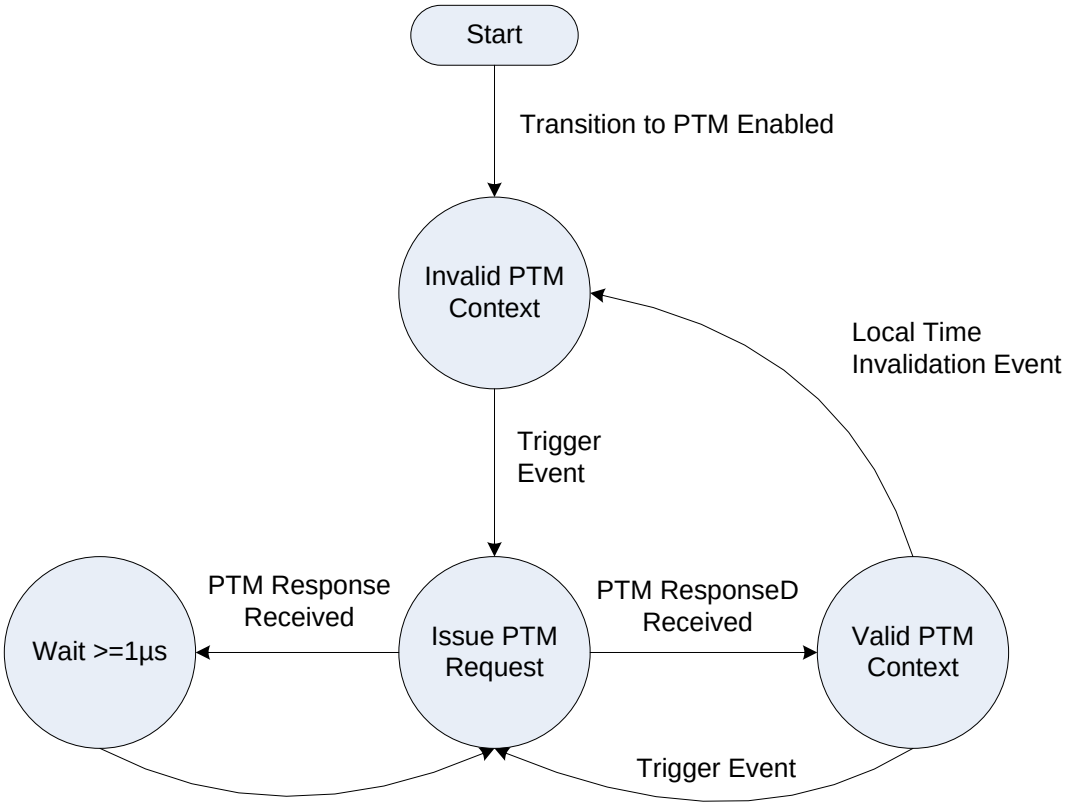
- Time values must be monotonic, and strictly increasing.
- The perceived granularity must be no greater than the value reported in the Local Clock Granularity field of the PTM Capability register.
- The perceived time must start no later than when the PTM Root processes its first PTM Request Message.

Referring to Figure 6-24, the following rules define timestamp capture:

- A PTM Requester must update its stored t1 timestamp when transmitting a PTM Request Message, even if that transmission is a replay.
- A PTM Responder must update its stored t2 timestamp when receiving a PTM Request Message, even if received TLP is a duplicate.
- A PTM Responder must update its stored t3 timestamp when transmitting a PTM Response or ResponseD Message, even if that transmission is a replay.
- A PTM Requester must update its stored t4 timestamp when receiving a PTM Response Message, even if received TLP is a duplicate.
    - Timestamps must be based on the STP Symbol or Token that frames the TLP, as if observing the first bit of that Symbol or Token at the Port's pins. Typically this will require an implementation specific adjustment to compensate for the inability to directly measure the time at the actual pins, as the time will commonly be measured at some internal point in the Rx or Tx path. The accuracy and consistency of this measurement are not bounded by this specification, but it is strongly recommended that the highest practical level of accuracy and consistency be achieved.

6.22.3.1 *PTM Requester Role*

- Support for the PTM Requester role is indicated by setting the PTM Requester Capable bit in the PTM Capability register.
- PTM Requesters are permitted to request PTM Master Time only when PTM is enabled. The mechanism for directing a PTM Requester to issue such a request is implementation specific.
    - Upstream Ports obtain PTM Master Time via PTM dialogs as described in section 2.2.8.10.
    - The mechanism by which RCiEPs request PTM Master Time is implementation specific.
- Once having issued a PTM Request Message, the Upstream Port must not issue another PTM Request Message prior to the receipt of a PTM Response Message, PTM ResponseD Message, Reset, or the passage of 100 μs without a corresponding PTM Message from the Downstream Port.
- Upon receiving a PTM Response, the Upstream Port must wait at least 1μs before issuing another PTM Request Message.
- For Multi-Function Devices (MFDs) containing multiple PTM Requesters, the Upstream Port associated with that MFD must issue a single PTM dialog during each PTM context refresh. PTM Requesters within the MFD maintain their individual PTM contexts using this one, Device-wide PTM dialog. The mechanism for refreshing multiple PTM contexts from one PTM dialog is implementation specific.
- It is strongly recommended that an Upstream Port invalidate its internal PTM context when the relationship between PTM Master Time and the Upstream Port's local time changes, as determined by implementation specific criteria. For example, this may occur as a result of a transition to a non-D0 state or due to accumulated PPM drift. These events are grouped under the label "Local Time Invalidation Event" in Figure 6-26.



1.

Figure 6-26: PTM Requester Operation

6.22.3.2 *PTM Responder Role*

- Support for the PTM Responder role is indicated by setting the PTM Responder Capable bit in the PTM Capability register.
- Switches and Root Complexes are permitted to implement the PTM Responder Role.
  - A PTM capable Switch, when enabled for PTM by setting the PTM Enable bit in the PTM Control register associated with the Switch Upstream Port, must respond to all PTM Request Messages received at any of its Downstream Ports.
  - The mechanism by which Root Complexes communicate PTM Master Time to RCiEPs is implementation specific.
- PTM Responders must populate PTM ResponseD Messages as follows (refer to Figure 6-24 and the accompanying implementation note):
  - The PTM Master Time field is a 64-bit value containing the value of PTM Master Time at the receipt of the PTM Request Message for the current PTM Dialog. In Figure 6-24, for the 2nd PTM dialog, this is the PTM Master Time at time t2'.
  - The Propagation Delay field is a 32-bit value containing the interval between the receipt of the PTM Request Message and the transmission of the PTM Response Message for the previous PTM dialog. In Figure 6-24, for the 2nd PTM dialog, this is the time interval between t2 and t3 captured during the 1st PTM dialog.
  - The unit of measurement for both fields is one ns.
- Switch Downstream Ports and Root Ports acting as PTM Responders must respond to each PTM Request Message received at their Downstream Ports with either PTM Response or PTM ResponseD according to the following rules:
  - A PTM Responder must not send a PTM Response or PTM ResponseD Message without first receiving a PTM Request Message.
  - Upon receipt of a PTM Request Message, a PTM Responder must attempt to issue a PTM Response or PTM ResponseD Message within 10 μs.
  - A PTM Responder must issue PTM Response when the Downstream Port does not have historical timestamps (t3 – t2) with which to fulfill a PTM Request Message.
  - A PTM Responder must issue PTM ResponseD when it has stored copies of the values required to populate the PTM ResponseD Message: historical timestamps (t3 –t2) and the PTM Master Time at the receipt of the most recent PTM Request Message (time t2').
  - A PTM Responder is permitted to issue PTM Response when it has stored copies of the historical timestamps (t3 – t2) but must request the PTM Master Time from elsewhere. In this case, it is permitted to issue PTM Response messages in response to PTM Request Messages while it retrieves the PTM Master Time if that retrieval is expected to take more than 10 μs.
  - The perceived granularity of the historical timestamps and PTM Master Time values transmitted by a PTM Responder must not exceed that reported in the Local Clock Granularity field of the PTM Capability register.

6.22.3.3 *PTM Time Source Role -- Rules Specific to Switches*

In addition to the requirements listed above for the PTM Requester and PTM Responder Roles, Switches must follow these requirements:

- When the Upstream Port is associated with a Multi-Function Device, only a single Function associated with that Upstream Port is permitted to implement the PTM Extended Capability structure. For Switches, all PTM functionality associated with the Switch must be controlled through that structure. It is not required that the Function implementing the PTM Extended Capability structure be the Switch Upstream Port Function.
- The PTM Extended Capability structure for a Switch must indicate support for both the PTM Requester and PTM Responder roles.
- The PTM Extended Capability in the Upstream Port controls all Switches in that Upstream Port.
- A Switch is permitted to act as a PTM Root, or to issue PTM Requests on its Upstream Port to obtain the PTM Master Time for use in fulfilling PTM Requests received at its Downstream Ports. In the latter case the Switch must account for any internal delays within the Switch.
- A Switch is permitted to maintain a local PTM context for use in fulfilling PTM Requests received on its Downstream Ports.
- A Switch which is not acting as a PTM Root must invalidate its local context no more than 10 ms from the last PTM dialog on its Upstream Port. The Switch must then refresh its local PTM context prior to issuing further PTM ResponseD Messages on its Downstream Ports. This requirement for periodic refreshes is optional if it is guaranteed by implementation-specific means that the Switch's local clock is phase locked with PTM Master Time.
- Any Switch implementing a local clock for the purpose of maintaining a local PTM context must report the granularity of this clock as defined in the PTM Capabilities structure (section 7.9.16).

IMPLEMENTATION NOTE

PTM Timestamp Capture Mechanisms

PTM uses services from both the Data Link and Transaction Layers. Accuracy requires that time measurements be taken as close to the Physical Layer as possible. Conversely, the messaging protocol itself properly belongs to the Transaction Layer. The PTM message protocol applies to a single Link, where the Upstream Port is the requester and the Downstream Port is the responder.

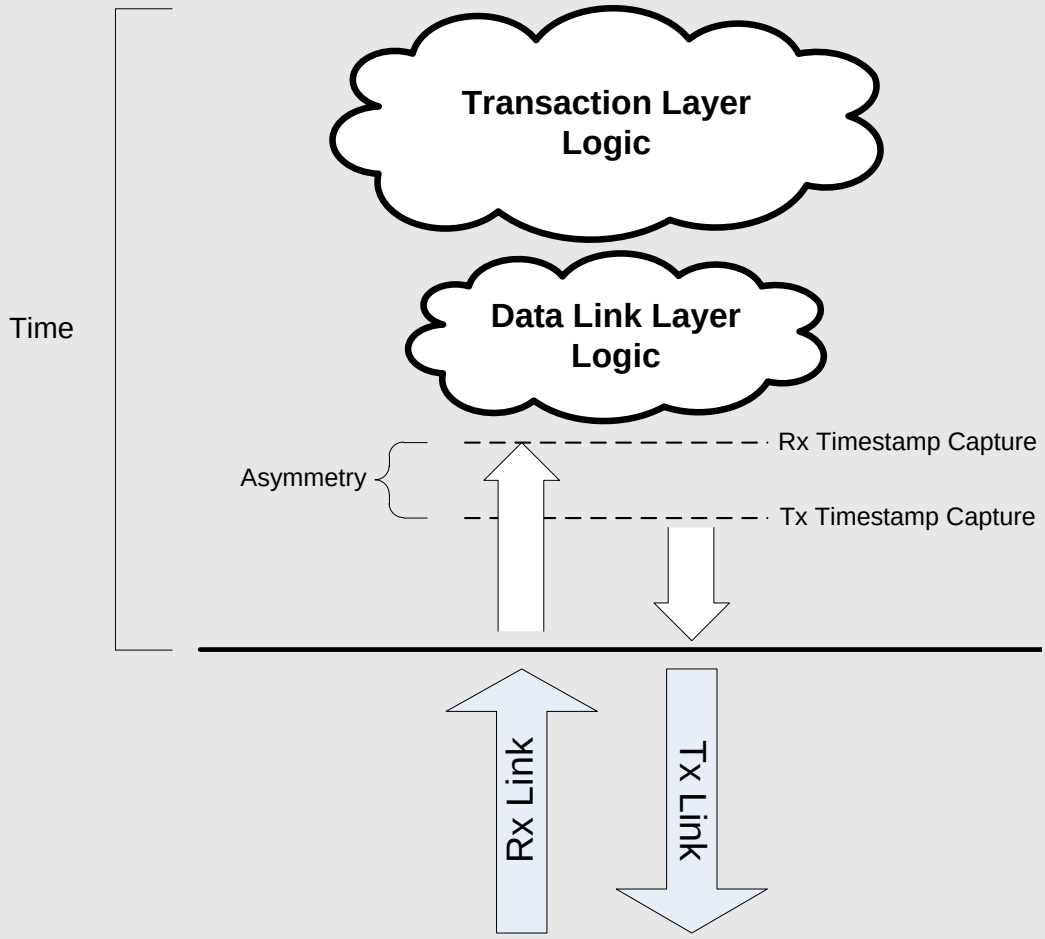


Figure 6-27: PTM Timestamp Capture Example

Figure 6-27 illustrates how to select suitable timestamp capture points. For some implementations, the logic within the Transaction Layer and Data Link Layers is non-deterministic. Implementation details and current conditions have considerable impact on exactly when a particular packet may encounter any particular processing step. This makes it effectively impossible to capture any timestamp that accurately records the time of a particular physical event if timestamps are captured in the higher layers.

6.23 Readiness Notifications (RN)

Readiness Notifications (RN) is intended to reduce the time software needs to wait before issuing Configuration Requests to a Device or Function following DRS Events or FRS Events. RN includes both the Device Readiness Status (DRS) and Function Readiness Status (FRS) mechanisms. These mechanisms provide a direct indication of Configuration-Readiness (see Terms and Acronyms entry for "Configuration-Ready"). When used, DRS and FRS allow an improved behaviour over the CRS mechanism, and eliminate its associated periodic polling time of up to 1 second following a reset.

It is permitted that system software/firmware provide mechanisms that supersede the FRS and/or DRS mechanisms, however such software/firmware mechanisms are outside the scope of this specification.

IMPLEMENTATION NOTE

Optimizing Configuration Readiness

It is strongly recommended that implementers of system firmware/software avoid unnecessary delays wherever possible. It is strongly recommended that hardware be designed to eliminate or minimize required delays, and to make full use of the mechanisms provided in this specification and related specifications to communicate what, if any, delays are required. Hardware implementers should appropriately document implementation behavior to enable system firmware/software to implement optimal behaviors.

Even with good documentation, some cases may at first appear problematic – for example, how can system firmware benefit from the Device Readiness Status (DRS) mechanism, when it is necessary to read from Root Port Configuration space to do so? In such cases, platform specific knowledge is required, i.e. that the Root Port supports Immediate Readiness.

6.23.1 Device Readiness Status (DRS)

When implemented, DRS must be used to indicate when a Device is Configuration-Ready following any of the following Device-level occurrences, which are subsequently referred to as "DRS Events":

- Exit from Cold Reset
- Exit from Warm Reset, Hot Reset, Loopback, or Disabled
- Exit from L2/L3 Ready
- Any other scenario where the Port transitions from DL_Down to DL_Up status.

The DRS Message protocol requirements include the following:

- There is no enable or disable mechanism for DRS. For Downstream Ports that support DRS, the DRS Supported bit in the Link Capabilities 2 register must be Set. For Upstream Ports that support DRS, it is strongly recommended that the DRS Supported bit in the Link Capabilities 2 register be Set. It is expressly permitted for Upstream Ports to send DRS Messages even when the DRS Supported bit is Clear.

- A DRS Message must be transmitted by a DRS-capable Upstream Port following every DL_Down to DL_Up transition when all non-VF Functions on the Logical Bus associated with that Upstream Port become ready.
  - A Type 0 Function is ready when it is Configuration-Ready.
  - A Type 1 Function that is a Switch Upstream Port is ready when it is Configuration-Ready and all Functions on its secondary bus are Configuration-Ready.
  - A Type 1 Function that is not a Switch Upstream Port is ready when the Function itself is Configuration-Ready.
- After a Device transmits a DRS Message, non-VF Functions indicated as Configuration-Ready by that DRS Message must not return Completions with CRS unless a subsequent DRS Event occurs.

Additional requirements relating to Switches implementing DRS include:

- Must support DRS functionality in all Ports
- Implementation at each Downstream Port of the DRS Signaling Control field.
- For any physically-integrated Device that appears beneath a Switch Downstream Port, the DRS sent by the Switch does not indicate Configuration Readiness for that Device
  - For such a Device, implementation and use of DRS is independent of the Switch

Additional requirements for Root Ports and Switch Downstream Ports include:

Implementation of the DRS Message Received bit, which indicates receipt of a DRS Message

IMPLEMENTATION NOTE

DRS Messages and ACS Source Validation

Functions are permitted to transmit DRS Messages before they have been assigned a Bus Number. Such messages will have a Requester ID with a Bus Number of 00h. If the Downstream Port has ACS Source Validation enabled, these Messages (see Section 6.12.1.1) will likely be detected as an ACS Violation error.

6.23.2 Function Readiness Status (FRS)

When implemented, FRS must be used to indicate a specific Function as being Configuration-Ready following any of the following Function-level occurrences, which are subsequently referred to as "FRS Events":

- Function Level Reset (FLR)
- Completion of $D3_{hot}$ to D0 transition
- Setting or Clearing of VF Enable in a PF (SR-IOV)

The FRS Message protocol requirements include the following:

- ❑ The Requester ID of the FRS Message must indicate the Function that has changed readiness status (see section 2.2.8.6.4)
- ❑ The FRS Reason field in the FRS Message must indicate why that Function changed readiness status.
- ❑ After a Function transmits an FRS Message, the indicated Function(s) must not return Completions with CRS unless a subsequent DRS Event or FRS Event occurs

Additional requirements for Switches implementing FRS include:

- ❑ Must support FRS functionality in the Upstream Port and all Downstream Ports
- ❑ The ability to transmit FRS Messages Upstream when required by the FRS protocol

Additional requirements for Physical Functions (PFs) include:

- ❑ The ability to transmit FRS Message Upstream when the VF Enable or VF Disable process completes

Additional requirements for Root Ports and Root Complex Event Collectors implementing FRS include:

- ❑ Must implement the FRS Queuing Extended Capability (see Section 7.8.9)

6.23.3 FRS Queuing

Root Ports and Root Complex Event Collectors that support FRS must implement the FRS Queuing Extended Capability (see Section 7.8.9).

For a Root Port, the FRS Message Queue contains FRS Messages received by the Root Port or generated by the Root Port.

For a Root Complex Event Collector, the FRS Message Queue contains FRS Messages generated by RCiEPs associated with the Root Complex Event Collector (see Section 7.9.10) or generated by the Root Complex Event Collector.

The FRS Message Queue must satisfy the following requirements:

- ❑ The FRS Message Queue must be empty following Reset.
- ❑ For a Root Port, the FRS Message Queue must be emptied when the Link goes to DL_Down.
- ❑ FRS Messages must be queued in the order received.
- ❑ If the FRS Message Queue is not full at the time an FRS Message is received or is internally generated, that FRS Message must be entered in the queue and the FRS Message Received bit must set to 1b.

- If the FRS Message Queue is full at the time an FRS Message is received or is internally generated, that FRS Message must be discarded and the FRS Message Overflow bit must be set to 1b. The pre-existing FRS Message Queue must be preserved.
- The oldest FRS Message must be visible in the FRS Message Queue register (see Section 7.8.9.4).
- Writing the FRS Message Queue register must remove the oldest element from the queue.
- When either FRS Message Received or FRS Message Overflow transitions from 0b to 1b, an interrupt must be generated if enabled.

6.24 Enhanced Allocation

The Enhanced Allocation (EA) Capability is an optional Capability that allows the allocation of I/O, Memory and Bus Number resources in ways not possible with the BAR and Base/Limit mechanisms in the Type 0 and Type 1 Configuration Headers.

It is only permitted to apply EA to certain functions, based on the hierarchal structure of the functions as seen in PCI configuration space, and based on certain aspects of how functions exist within a platform environment (see Figure 6-28).

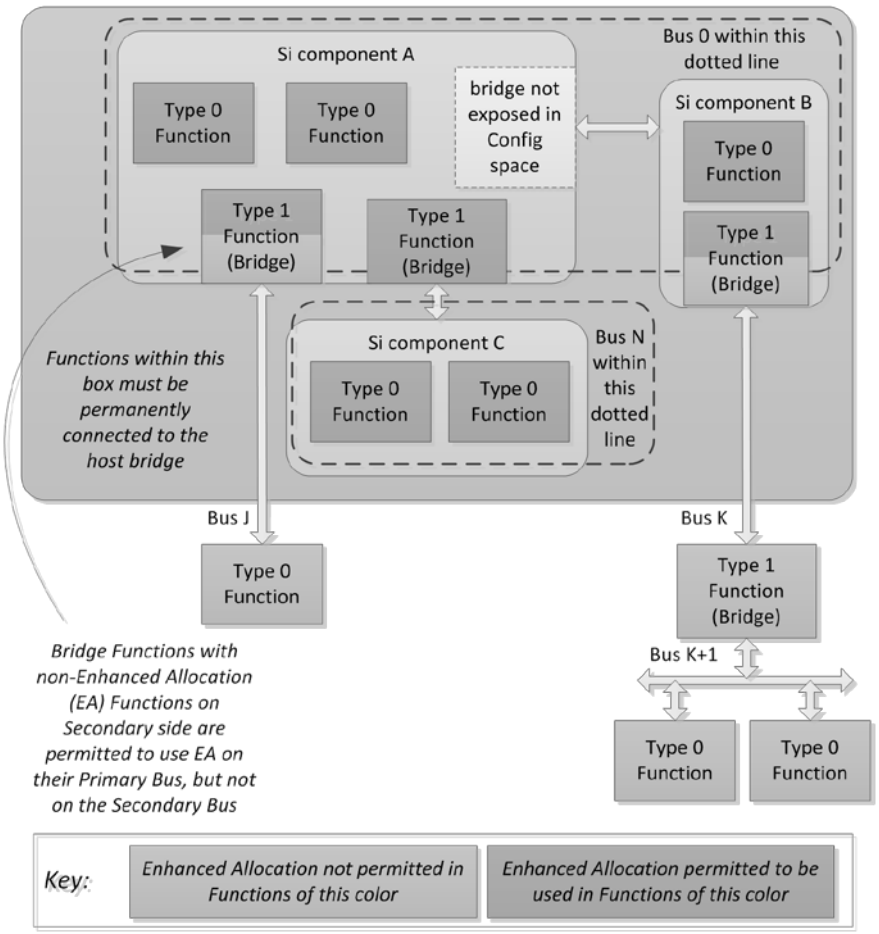


Figure 6-28: Example Illustrating Application of Enhanced Allocation

Only functions that are permanently connected to the host bridge are permitted to use EA. A bridge function (i.e., any function with a Type 1 Configuration Header), is permitted to use EA for both its Primary Side and Secondary Side if and only if the function(s) behind the bridge are also permanently connected (below one or more bridges) to the host bridge, as shown for "Si component C" in Figure 6-28.

A bridge function is permitted to use EA only for its Primary Side if the function(s) behind the bridge are not permanently connected to the bridge, as with the bridges above Bus J and Bus K in Figure 6-28, and in this case the non-EA resource allocation mechanisms in the Type 1 Header for Bus numbers, MMIO ranges and I/O ranges are used for the Secondary side of the bridge. System software must ensure that the allocated Bus numbers are within the range indicated in the Fixed Secondary Bus Number and Fixed Subordinate Bus Number registers of the EA capability. System software must ensure that the allocated MMIO and I/O ranges are within ranges indicated with the corresponding Properties in the EA capability for resources to be allocated behind the bridge. For Bus numbers, MMIO and I/O ranges behind the bridge, hardware is permitted to indicate overlapping ranges in multiple bridge functions, however, in such cases, system software must ensure that the ranges actually assigned are non-overlapping.

Functions that rely exclusively on EA for I/O and Memory address allocation must hardwire all bits of all BARs in the PCI Header to 0. Such Functions must be clearly documented as relying on EA for correct operation, and platform integrators must ensure that only EA-aware firmware/software are used with such Functions.

When a Function allocates resources using EA and indicates that a resource range is associated with an equivalent BAR number, the Function must not request resources through the equivalent BAR, which must be indicated by hardwiring all bits of the equivalent BAR to 0.

For a bridge function that is permitted to implement EA based on the rules above, it is permitted, but not required, for the bridge function to use EA mechanisms to indicate resource ranges that are located behind the bridge Function. In the example shown in in Figure 6-28, the bridge above Bus N is permitted to use EA mechanisms to indicate the resources used by the two functions in "Si component C", or that bridge is permitted to not indicate the resources used by the two functions in "Si component C". System firmware/software must comprehend that such bridge functions are not required to indicate inclusively all resources behind the bridge, and as a result system firmware/software must make a complete search of all functions behind the bridge to comprehend the resources used by those functions.

A Function with an Expansion ROM is permitted use the existing mechanism or the EA mechanism, but is not permitted to support both. If a Function uses the EA mechanism (EA entry with BEI of 8), the Expansion ROM Base Address Register (offset 30h) must be hardwired to 0. The Enable bit of the EA entry is equivalent to the Expansion ROM Enable bit. If a Function uses Expansion ROM Base Address Register mechanism, no EA entry with a BEI of 8 is permitted.

The requirements for enabling and/or disabling the decode of I/O and/or Memory ranges are unchanged by EA, including but not limited to the Memory Space and I/O Space enable bits in the Command register.

Any resource allocated using EA must not overlap with any other resource allocated using EA, except as permitted above for identifying permitted address ranges for resources behind a bridge.

6.25 Emergency Power Reduction State

Emergency Power Reduction State is an optional mechanism to request that Functions quickly reduce their power consumption. Emergency Power Reduction is a fail-safe mechanism intended to be used to prevent system damage and is not intended to provide normal dynamic power management.

If a Function implements Emergency Power Reduction State, it must also implement the Power Budgeting extended capability and must report Power Budgeting values for this state (see Section 7.8.1). Devices that are integrated on the system board are not required to implement the Power Budgeting extended capability, but if they do so, they must meet the preceding requirement.

Functions enter and exit this state based on either autonomously or via external requests. External requests may be either following a signaling protocol defined in an applicable form factor specification, or by a vendor-specific method. Table 6-14 defines how the Emergency Power Reduction Supported and Emergency Power Reduction Initialization Required fields determine the mechanisms that are allowed to trigger entry and exit from this state (see Section 7.5.3.15).

Table 6-14: Emergency Power Reduction Supported Values

Emergency Power Reduction Supported	Emergency Power Reduction Initialization Required	Entry/Exit Permitted by		
		Form Factor Mechanism	Vendor Specific Mechanism(s)	Autonomous Mechanisms
00b	0	No	Yes	Yes
	1	No	No	No
01b	Any	No	Yes	Yes
10b	Any	Yes	Yes	Yes
11b	Reserved			

Functions may indicate that they require re-initialization on exit from this state:

- If the Emergency Power Reduction Initialization Required bit is Clear (see Section 7.5.3.15):
  - On entry to this state, the Function either operates normally (perhaps with reduced performance), or enters a device specific "power reduction dormant state". The Upstream Port of the Device remains operating. Outstanding requests initiated by or directed to the Function must complete normally.
  - On exit from this state, the Function operates normally (perhaps resuming normal performance). Functions that entered a "power reduction dormant state" exit that state. In either case, no software intervention is required.
- If the Emergency Power Reduction Initialization Required bit is Set (see Section 7.5.3.15):
  - On entry to this state, the Function ceases normal operation. The Upstream Port of the associated Device is permitted to enter DL_Down.
    - If the Upstream Port remains in DL_Up, outstanding requests directed to or initiated by the Function must complete normally.

        - If the Upstream Port enters DL_Down, outstanding request behavior is defined in Section 2.9.1. This transition may result in a Surprise Down error.
        - Sticky bits must be preserved in this state.
    - On exit from this state, software intervention is required to resume normal operation. The mechanism used to indicate to software when this is required is outside the scope of this specification (e.g., a device specific interrupt). If the Upstream Port entered DL_Down, all Functions of the Device are reset and a full reconfiguration is required (see Section 2.9.2).

The following rules apply to the Emergency Power Reduction State:

- A Device supports Emergency Power Reduction State if at least one Function in the Upstream Port indicates support (i.e., Emergency Power Reduction Supported is non-zero).
- Emergency Power Reduction State is associated with a Device. All Functions in a Device that support it enter and exit this state at the same time.
- Functions where the Emergency Power Reduction Supported field is 00b are not affected by the Emergency Power Reduction State of the Device as long as the Upstream Port remains in DL_Up. The Emergency Power Reduction Detected bit is RsvdZ.
- Functions where the Emergency Power Reduction Supported field is 01b or 10b:
    - Set the Emergency Power Reduction Detected bit when the Device enters Emergency Power Reduction State.
    - Clear the Emergency Power Reduction Detected bit when requested if the Device has exited the Emergency Power Reduction State.
- For Switches, Downstream Switch Ports enter and exit Emergency Power Reduction State at the same time as the associated Upstream Switch Port. The corresponding fields in Configuration Space are reserved for Downstream Switch Ports.
- For SR-IOV Devices, VFs enter and exit Emergency Power Reduction State at the same time as their PF. The corresponding fields in Configuration Space are reserved for VFs.
- Encoding 10b shall not be used unless the associated form factor specification defines a mechanism for requesting Emergency Power Reduction.
- It is strongly recommended that the Emergency Power Reduction Supported field be initialized by hardware or firmware within the Function prior to initial device enumeration. This initialization is permitted to be deferred to device driver load when this is not practical (e.g., when there is no firmware ROM).

IMPLEMENTATION NOTE

Diagnostic Checking of Emergency Power Reduction Detected

The Emergency Power Reduction Detected bit permits system software to detect that Emergency Power Reduction State was entered, even momentarily. The Emergency Power Reduction Request bit can be used by software to request entry. Normally, software would use a system specific method to enter the Emergency Power Reduction State using external mechanisms.

IMPLEMENTATION NOTE

Emergency Power Reduction State: Example Add-in Card

Figure 6-29 shows an example multi-Device add-in card supporting Emergency Power Reduction. Note that Device C does not support the Emergency Power Reduction State. Device C might be a Switch that fans out to Devices A and B.

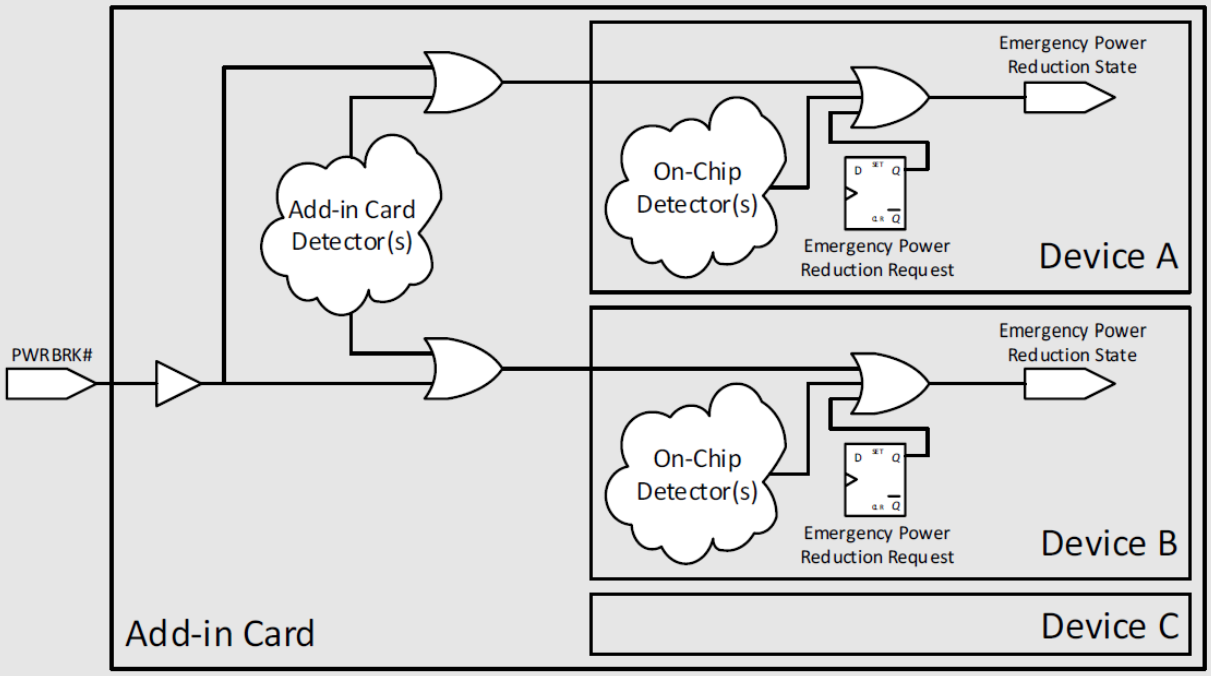


Figure 6-29: Emergency Power Reduction State: Example Add-in Card

6.26 Hierarchy ID Message

When software initializes a PCI Hierarchy, it assigns unique Bus and Device numbers to each component so that every Function in the Hierarchy has a unique Routing ID within that Hierarchy. To ensure that Routing IDs are unique in large systems that contain more than one Hierarchy and in clustered systems that contain multiple Hierarchies, additional information is required to augment the Routing ID to produce a unique number. Functions can be uniquely identified by the combination of:

- Unique Identifier for the System (or Root Complex)
- Unique Identifier for the Hierarchy within that Root Complex
- Routing ID within that Hierarchy

The Hierarchy ID message (see Section 2.2.8.6.5) is used to provide the additional information needed for a Function to uniquely identify itself in a multi-hierarchy platform.

Hierarchy ID Messages are generated by a Downstream Port upon software request. Received messages at an Upstream Port are reported in the Hierarchy ID Extended Capability (see Section 7.9.18).

Hierarchy ID Messages are a PCI-SIG-Defined Type 1 VDM. Hierarchy ID Messages can safely be sent at any time and components that do not comprehend them will silently ignore them.

Hierarchy ID Messages typically are sent from a Downstream Port at the top of the Hierarchy (e.g., a Root Port). In systems where the Root Port does not support Hierarchy ID Messages, Hierarchy ID Messages can be sent from Switch Downstream Ports.

The Hierarchy ID Message is intended for use by software, firmware, and/or hardware. When using the Hierarchy ID Message, all bits of the Hierarchy ID, System GUID, System GUID Authority ID fields must be compared, without regard to any internal structure. How this information is used is outside the scope of this specification.

Layout of the Hierarchy ID Message is shown in Figure 2-30. Fields in the Hierarchy ID Message are as follows:

Hierarchy ID contains the Segment Group Number associated with this Hierarchy (as defined by the *PCI Firmware Specification*). This field can be used in conjunction with the Routing ID to uniquely identify a Function within a System. The value 0000h indicates the default (or only) Hierarchy of the Root Complex. Non-zero values indicate additional Hierarchies.

System GUID[143:0], in conjunction with System GUID Authority ID, provides a globally unique identification for a System.

System GUID[143:136] is byte 14 in the Hierarchy ID Message.
System GUID[135:128] is byte 15 in the Hierarchy ID Message.
System GUID[127:120] is byte 16 in the Hierarchy ID Message.
System GUID[119:112] is byte 17 in the Hierarchy ID Message.
System GUID[111:104] is byte 18 in the Hierarchy ID Message.
System GUID[103: 96] is byte 19 in the Hierarchy ID Message.
System GUID[95: 88] is byte 20 in the Hierarchy ID Message.
System GUID[87: 80] is byte 21 in the Hierarchy ID Message.
System GUID[79: 72] is byte 22 in the Hierarchy ID Message.
System GUID[71: 64] is byte 23 in the Hierarchy ID Message.
System GUID[63: 56] is byte 24 in the Hierarchy ID Message.
System GUID[55: 48] is byte 25 in the Hierarchy ID Message.
System GUID[47: 40] is byte 26 in the Hierarchy ID Message.
System GUID[39: 32] is byte 27 in the Hierarchy ID Message.
System GUID[31: 24] is byte 28 in the Hierarchy ID Message.
System GUID[23: 16] is byte 29 in the Hierarchy ID Message.
System GUID[15: 8] is byte 30 in the Hierarchy ID Message.
System GUID[7: 0] is byte 31 in the Hierarchy ID Message.

System GUID Authority ID identifies the mechanism used to ensure that the System GUID is globally unique. The mechanism for choosing which Authority ID to use for a given system is implementation specific. The defined values are shown in Table 6-15.

Table 6-15: System GUID Authority ID Encoding

Authority ID	Description
00h	**None** – System GUID[143:0] is not meaningful. System GUID[143:0] must be 0.
01h	**Timestamp** – System GUID[63:0] contains a timestamp associated with the particular system. Encoding is a Unix 64 bit time (number of seconds since midnight UTC January 1, 1970). The mechanism of choosing the timestamp to represent a system is implementation specific. System GUID[143:64] must be 0.
02h	**IEEE EUI-48** – System GUID[47:0] contains a 48 bit Extended Unique Identifier (EUI-48) associated with the particular system. Encoding is defined by the IEEE. See EUI-48[112] for details. EUI-48 values are frequently used as network interface MAC addresses. The mechanism of choosing the EUI-48 value to represent a system is implementation specific. System GUID[143:48] must be 0.
03h	**IEEE EUI-64** – System GUID[63:0] contains a 64 bit Extended Unique Identifier (EUI-64) associated with the particular system. Encoding is defined by the IEEE. See EUI-64[113] for details. The mechanism of choosing the EUI-64 value to represent a system is implementation specific. System GUID[143:64] must be 0.
04h	**RFC-4122 UUID** – System GUID[127:0] contain a UUID as defined by the IETF in RFC-4122[114]. This definition is technically equivalent to ITU-T Rec. X.667[115] | ISO/IEC 9834-8. The mechanism of choosing the UUID value to represent a system is implementation specific. System GUID[143:128] must be 0

[112] https://standards.ieee.org/develop/regauth/tut/eui48.pdf

[113] https://standards.ieee.org/develop/regauth/tut/eui64.pdf

[114] https://tools.ietf.org/html/rfc4122

[115] http://www.itu.int/rec/T-REC-X.667-201210-I/en

Authority ID	Description
05h	**IPv6 Address** – System GUID[127:0] contains the unique IPv6 address of one of the network interfaces of the system. The mechanism of choosing the IPv6 value to represent a system is implementation specific. System GUID[143:128] must be 0.
06h to 7Fh	**Reserved** – System GUID[143:0] contains a unique value. The mechanism used to ensure uniqueness is outside the scope of this specification.
80h to FFh	**PCI-SIG Vendor Specific** – System GUID Authority ID values 80h to FFh are reserved for PCI-SIG vendor-specific usage. System GUID[143:128] contains a PCI-SIG assigned Vendor ID. System GUID[127:0] contain a unique number assigned by that vendor. The mechanism used for assigning numbers is implementation specific. One possible mechanism would be to use the serial number assigned to the system. The mechanism used to choose between these System GUID Authority IDs is implementation specific. One usage would be to allow a vendor to define up to 128 distinct 128-bit System GUID schemes.

IMPLEMENTATION NOTE

System GUID Consistency and Stability

To support the purpose of System GUID, software should ensure that a single system uses identical System GUID and System GUID Authority ID values everywhere.

Implementers should carefully consider their stability requirements for the System GUID value. For example, some use cases may require that the value not change when the system is rebooted. In those cases, a mechanism that picks the EUI-48 value associated with the first Ethernet MAC address discovered might be problematic if the result changes due to hardware failure, system reconfiguration, or variations/parallelism in the discovery algorithm.

IMPLEMENTATION NOTE

Hierarchy ID vs. Device Serial Number

The Device Serial Number mechanism can also be used to uniquely identify a component (see Section 7.9.3). Device Serial Number may be a more expensive solution to this problem if it involves a ROM associated with each component.

IMPLEMENTATION NOTE

Virtual Functions and Hierarchy ID

The Hierarchy ID capability can be emulated by the Virtualization Intermediary (VI). Doing so provides VF software access to this Hierarchy ID information.

When VF hardware needs access to this information, the VF should implement the Hierarchy ID capability. This provides access to both VF software and hardware.

In some situations, the VF should get the same information as the PF. In other situations, particularly those involving migration of Virtual Machines, it may be appropriate to present the VF with Hierarchy ID information that differs from the associated PF and from other VFs associated with that PF.

The following mechanisms are supported:

	VF Hierarchy ID Capability	Hierarchy ID VF Configurable	Hierarchy ID Writeable	VF Software has access	VF Hardware has access	VF Hierarchy Data / GUID
1	Not Present	n/a	n/a	No	No	Not Emulated
2				Yes	No	Emulated
3	Present	0b	0b	Yes	Yes	Same as PF
4		1b	0b	Yes	Yes	Same as PF
5		1b	1b	Yes	Yes	Configured by VI

In mechanism 1, the the Virtualization Intermediary does not emulate the capability. VF software and hardware have no access.

In mechanism 2, the Virtualization Intermediary emulates the capability and returns whatever Hierarchy ID information is desired. VF software has access. VF hardware does not have access.

In mechanisms 3 and 4, VF information is the same as the PF and is automatically filled in from received Hierarchy ID messages. Both VF hardware and software have access.

In mechanism 5, VF information is configured by software (probably the VI). Both VF hardware and software have access.

6.27 Flattening Portal Bridge (FPB)

6.27.1 Introduction

The Flattening Portal Bridge (FPB) is an optional mechanism which can be used to improve the scalability and runtime reallocation of Routing IDs and Memory Space resources.

For non-ARI Functions associated with an Upstream Port, the Routing ID consists of a 3-bit Function Number portion, which is determined by the construction of the Upstream Port

hardware, and a 13-bit Bus Number and Device number portion, determined by the Downstream Port above the Upstream port.

For ARI Functions associated with an Upstream Port, the Routing ID consists of an 8-bit Function Number portion, and only the 8-bit Bus Number portion is determined by the Downstream Port above the Upstream port.

A bridge that implements the FPB capability can itself also be referred to as an FPB. The FPB capability can be applied to any logical bridge, as illustrated in Figure 6-30.

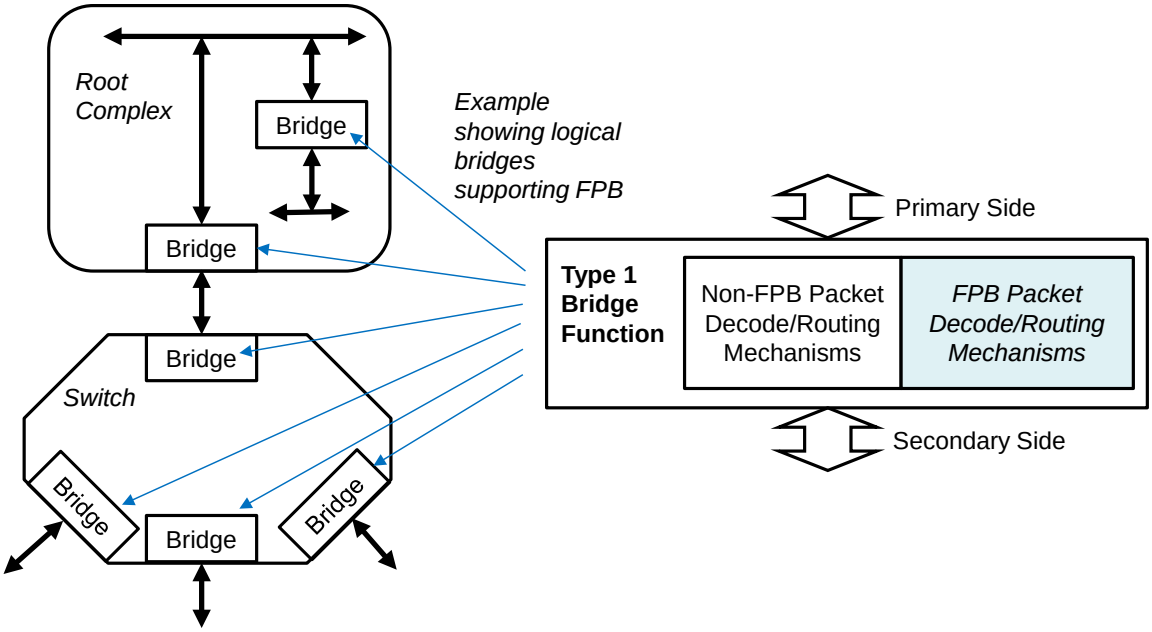


Figure 6-30: FPB High Level Diagram and Example Topology

FPB changes the way Bus Numbers are consumed by Switches to reduce waste, by "flattening" the way Bus Numbers are used inside of Switches and by Downstream Ports (see Figure 6-31).

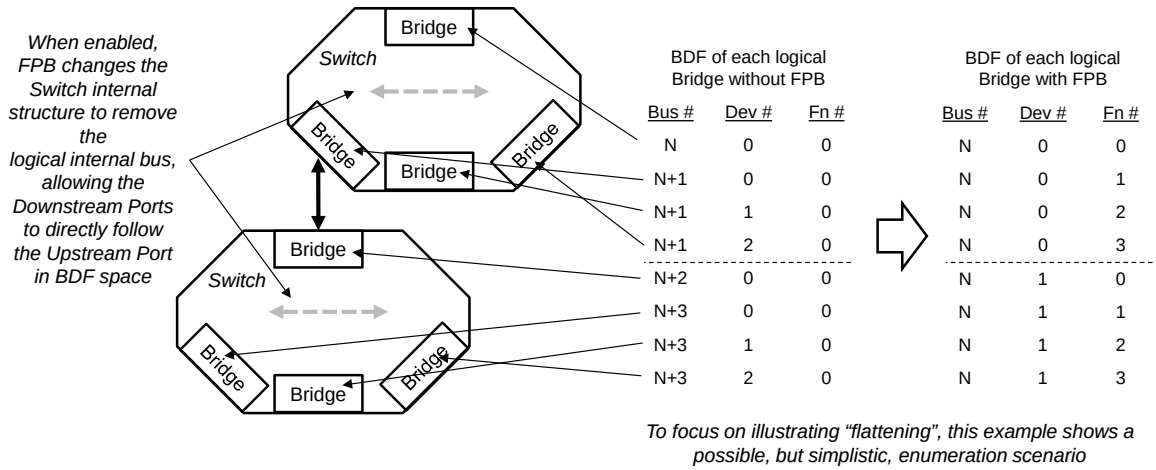


Figure 6-31: Example Illustrating "Flattening" of a Switch

FPB defines mechanisms for system software to allocate Routing IDs and Memory Space resources in non-contiguous ranges, enabling system software to assign pools of these resources from which it can allocate "bins" to Functions below the FPB. This is done using a bit vector where each bit when Set assigns a corresponding range of resources to the Secondary Side of the bridge (see Figure 6-32).

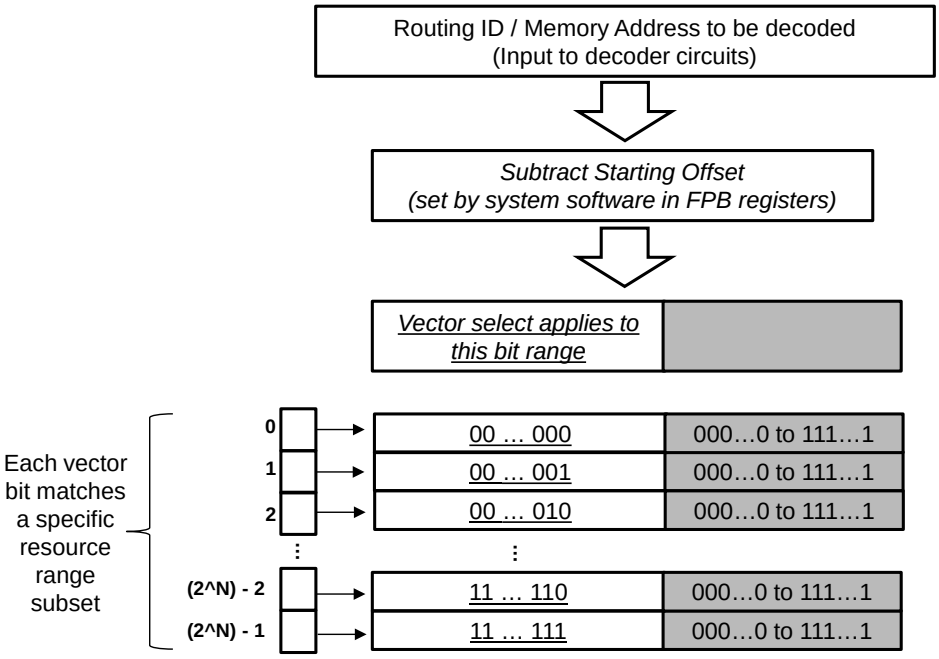


Figure 6-32: Vector Mechanism for Address Range Decoding

This allows system software to assign Routing IDs and/or Memory Space resources required by a device hot-add without having to rebalance other, already assigned resource ranges, and to return to the pool resources freed, for example by a hot remove event.

FPB is defined to allow both the non-FPB and FPB mechanisms to operate simultaneously, such that, for example, it is possible for system firmware/software to implement a policy where the non-FPB mechanisms continue to be used in parts of the system where the FPB

mechanisms are not required (see Figure 6-33). In this figure, the decode logic is assumed to provide a '1' output when a given TLP is decoded as being associated with the bridge's Secondary Side. The non-FPB decode mechanisms apply as without FPB, so for example only the Bus Number portion (bits 15:8) of a Routing ID is tested by the non-FPB decode logic when evaluating an ID routed TLP.

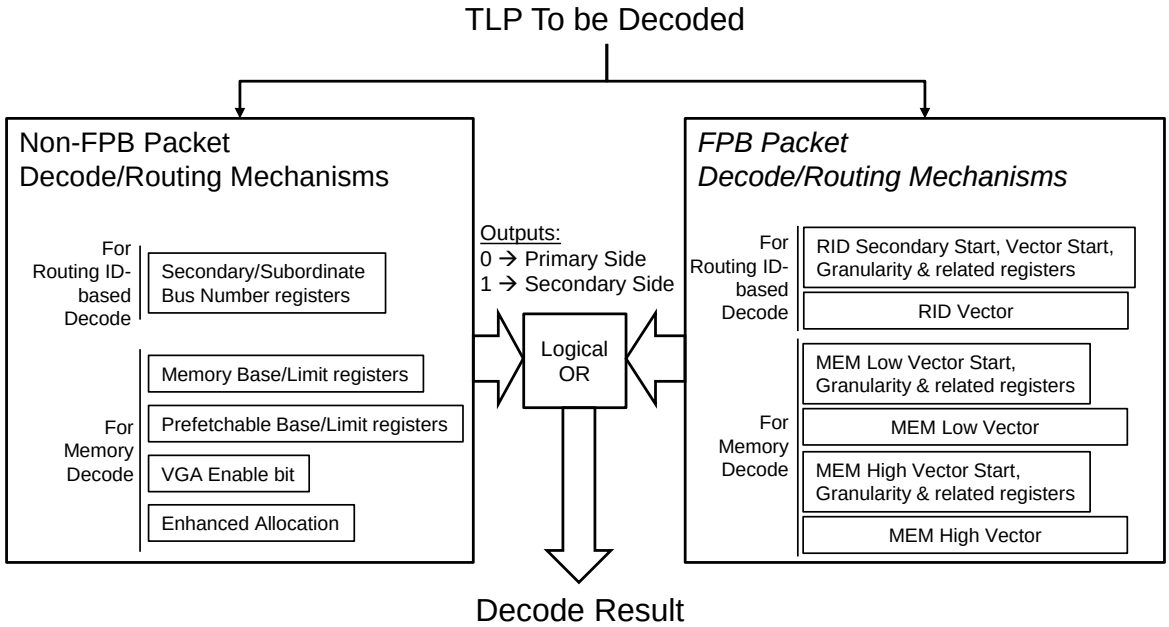


Figure 6-33: Relationship between FPB and non-FPB Decode Mechanisms

It is important to recognize that, although FPB adds additional ways for a specific bridge to decode a given TLP, FPB does not change anything about the fundamental ways that bridges operate within the Switch and Root Complex architectural structures. FPB uses the same architectural concepts to provide management mechanisms for three different resource types:

1. Routing IDs
2. Memory below 4 GB ("MEM Low")
3. Memory above 4 GB ("MEM High")

A hardware implementation of FPB is permitted to support any combination of these three mechanisms. For each mechanism, FPB uses a bit-vector to indicate, for a specific subset range of the selected resource type, if resources within that range are associated with the Primary or Secondary side of the FPB. Hardware implementations are permitted to implement a small range of sizes for these vectors, and system firmware/software is enabled to make the most effective use of the available vector by selecting an initial offset at which the vector is applied, and a granularity for the individual bits within the vector to indicate the size of the resource range to which the bits in a given vector apply.

6.27.2 Hardware and Software Requirements

The following rules apply when any of the FPB mechanisms are used:

- If system software violates any of the rules concerning FPB, the hardware behavior is undefined.
- It is permitted to implement FPB in any PCI bridge (Type 1) Function, and every Function that implements FPB must implement the FPB Capability (see Section 7.y).
- If a Switch implements FPB then the Upstream Port and all Downstream Ports of the Switch must implement FPB.
- Software is permitted to enable FPB at some Switch Ports and not others.
- A Root Complex is permitted to implement FPB on some Root Ports but not on others.
- A Type 1 Function is permitted to implement the FPB mechanisms applying to any one, two or three of these elemental mechanisms:
  - Routing IDs (RID)
  - Memory below 4 GB ("MEM Low")
  - Memory above 4 GB ("MEM High")
- System software is permitted to enable any combination (including all or none) of the elemental mechanisms supported by a specific FPB.
- The error handling and reporting mechanisms, except where explicitly modified in this section, are unaffected by FPB.
- Following any reset of the FPB Function, the FPB hardware must Clear all bits in all implemented vectors.
- Once enabled (through the FPB RID Decode Mechanism Enable, FPB MEM Low Decode Mechanism Enable, and/or FPB MEM High Decode Mechanism Enable bits), if system software subsequently disables an FPB mechanism, the values of the entries in the associated vector are undefined, and if system software subsequently re-enables that FPB mechanism the FPB hardware must Clear all bits in the associated vector.
- If an FPB is implemented with the No_Soft_Reset bit Clear, when that FPB is cycled through $D0 \rightarrow D3_{hot} \rightarrow D0$, then all FPB mechanisms must be disabled, and the FPB must Clear all bits in all implemented vectors.
- If an FPB is implemented with the No_Soft_Reset bit Set, when that FPB is cycled through $D0 \rightarrow D3_{hot} \rightarrow D0$, then all FPB configuration state must not change, and the entries in the FPB vectors must be retained by hardware.
- Hardware is not required to perform any type of bounds checking on FPB calculations, and system software must ensure that the FPB parameters are correctly programmed
  - It is explicitly permitted for system software to program Vector Start values that cause the higher order bits of the corresponding vector to surpass the resource range associated with a given FPB, but in these cases system software must ensure that those higher order bits of the vector are Clear.

- Examples of errors that system software must avoid include duplication of resource allocation, combinations of start offsets with set vector bits that could create "wrap-around" or bounds errors

The following rules apply to the FPB Routing ID (RID) mechanism:

- FPB hardware must consider a specific range of RIDs to be associated with the Secondary side of the FPB if the Bus Number portion falls within the Bus Number range indicated by the values programmed in the Secondary and Subordinate Bus Number registers logically OR'd with the value programmed into the corresponding entry in the FPB RID Vector.
- If it is intended to use only the FPB RID mechanism for BDF decoding, then system software must ensure that both the Secondary and Subordinate Bus Number registers are 0.
- System software must ensure that the FPB routing mechanisms are configured such that Configuration Requests targeting Functions Secondary side of the FPB will be routed by the FPB from the Primary to Secondary side of the FPB.

When ARI is not enabled, The FPB RID mechanism can be applied with different granularities, programmable by system software through the FPB RID Vector Granularity field in the FPB RID Vector Control 1 register. Figure 6-34 illustrates the relationships between the layout of RIDs and the supported granularities. The reader may find it helpful to refer to this figure when considering the requirements defined below and in the definition of the Flattening Portal Bridge (FPB) Capability (see Section 7.8.6).

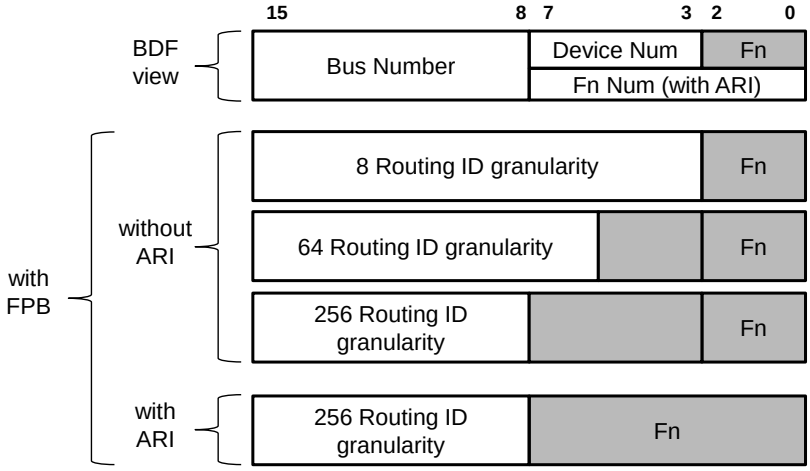


Figure 6-34: Routing IDs (RIDs) and Supported Granularities

- System software must program the FPB RID Vector Granularity and FPB RID Vector Start fields in the FPB RID Vector Control 1 register per the constraints described in the descriptions of those fields.
- For all FPBs other than those associated with Upstream Ports of Switches:
  - When ARI Forwarding is not supported, or when the ARI Forwarding Enable bit in the Device Control 2 register is Clear, FPB hardware must convert a Type 1 Configuration Request received on the Primary side of the FPB to a Type 0 Configuration Request on the Secondary side of the FPB when bits 15:3 of the Routing ID of the Type 1

Configuration Request matches the value in the RID Secondary Start field in the FPB RID Vector Control 2 register, and system software must configure the FPB accordingly.

- When the ARI Forwarding Enable bit in the Device Control 2 register is Set, FPB hardware must convert a Type 1 Configuration Request received on the Primary side of the FPB to a Type 0 Configuration Request on the Secondary side of the FPB when the Bus Number portion of the Routing ID of the Type 1 Configuration Request matches the value in the Bus Number address (bits 15:8 only) of the Secondary Start field in the FPB RID Vector Control 2 register, and system software must configure the FPB accordingly.

- For FPBs associated with Upstream Ports of Switches only, when the FPB RID Decode Mechanism Enable bit is Set, FPB hardware must use the FPB Num Sec Dev field of the FPB Capability register to indicate the quantity of Device Numbers associated with the Secondary Side of the Upstream Port bridge, which must be used by the FPB in addition to the RID Secondary Start field in the FPB RID Vector Control 2 register to determine when a Configuration Request received on the Primary side of the FPB targets one of the Downstream Ports of the Switch, determining in effect when such a Request must be converted form a Type 1 Configuration Request to a Type 0 Configuration Request, and system software must configure the FPB appropriately.
  - System software configuring FPB must comprehend that the logical internal structure of a Switch will change depending on the value of the FPB RID Decode Mechanism Enable bit in the Upstream Port of a Switch.
  - Downstream Ports must use their corresponding RID values, and their Requester IDs and Completer IDs, as determined by the Upstream Port's FPB Num Sec Dev and RID Secondary Start values
- FPBs must implement bridge mapping for INTx virtual wires (see Section <2.2.8.1>)
- Hardware and software must apply this algorithm (or the logical equivalent) to determine which entry in the FPB RID Vector applies to a given Routing ID (RID) address:
  - IF the RID is below the value of FPB RID Vector Start, then the RID is out of range (below the start) and so cannot be associated with the Secondary side of the bridge, ELSE
  - calculate the offset within the vector by first subtracting the value of FPB RID Vector Start, then dividing this according to the value of FPB RID Vector Granularity to determine the bit index within the vector.
  - IF the bit index value is greater than the length indicated by FPB RID Vector Size Supported, then the RID is out of range (beyond the top of the range covered by the vector) and so cannot be associated with the Secondary side of the bridge, ELSE
  - if the bit value within the vector at the calculated bit index location is 1b, THEN the RID address is associated with the Secondary side of the bridge, ELSE the RID address is associated with the Primary side of the bridge.

The following rules apply to the FPB MEM Low mechanism:

The FPB MEM Low mechanism can be applied with different granularities, programmable by system software through the FPB MEM Low Vector Granularity field in the FPB MEM Low

Vector Control register. Figure 6-35 illustrates the relationships between the layout of addresses in the memory address space below 4 GB to which the FPB MEM Low mechanism applies. The reader may find it helpful to refer to this figure when considering the requirements defined below and in the definition of the Flattening Portal Bridge (FPB) Capability (see Section <7.y>).

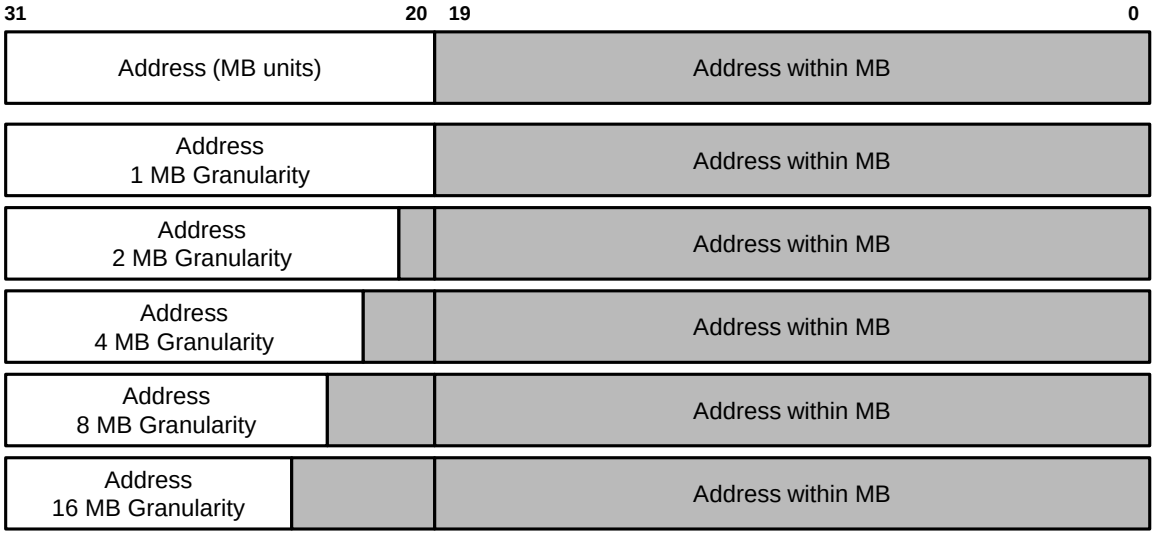


Figure 6-35: Addresses in Memory Below 4 GB and Effect of Granularity

- System software must program the FPB MEM Low Vector Granularity and FPB MEM Low Vector Start fields in the FPB MEM Low Vector Control register per the constraints described in the descriptions of those fields.
- FPB hardware must consider a specific Memory address to be associated with the Secondary side of the FPB if that Memory address falls within any of the ranges indicated by the values programmed in other bridge Memory decode registers (enumerated below) logically OR'd with the value programmed into the corresponding entry in the FPB MEM Low Vector. Other bridge Memory decode registers include:
  - Memory Space Enable bit in the Command register
  - Memory Base/Limit registers
  - Prefetchable Base/Limit registers
  - VGA Enable bit in the Bridge Control register
  - Enhanced Allocation (EA) Capability (if supported)
  - FPB MEM High mechanism (if supported and enabled)
- Hardware and software must apply this algorithm (or the logical equivalent) to determine which entry in the FPB MEM Low Vector applies to a given Memory address:
  - If the Memory address is below the value of FPB MEM Low Vector Start, then the Memory address is out of range (below) and so is not associated with the Secondary side of the bridge by means of this mechanism, else

- calculate the offset within the vector by first subtracting the value of FPB MEM Low Vector Start, then dividing this according to the value of FPB MEM Low Vector Granularity to determine the bit index within the vector.
- If the bit index value is greater than the length indicated by FPB MEM Low Vector Size Supported, then the Memory address is out of range (above) and so is not associated with the Secondary side of the bridge by means of this mechanism, else
- if the bit value within the vector at the calculated bit index location is 1b, then the Memory address is associated with the Secondary side of the bridge, else the Memory address is associated with the Primary side of the bridge.

The following rules apply to the FPB MEM High mechanism:

- System software must program the FPB MEM High Vector Granularity and FPB MEM High Vector Start Lower fields in the FPB MEM High Vector Control 1 register per the constraints described in the descriptions of those fields.
- FPB hardware must consider a specific Memory address to be associated with the Secondary side of the FPB if that Memory address falls within any of the ranges indicated by the values programmed in other bridge Memory decode registers (enumerated below) logically OR'd with the value programmed into the corresponding entry in the FPB MEM Low Vector. Other bridge Memory decode registers include:
  - Memory Space Enable bit in the Command register
  - Memory Base/Limit registers
  - Prefetchable Base/Limit registers
  - VGA Enable bit in the Bridge Control register
  - Enhanced Allocation (EA) Capability (if supported)
  - FPB MEM Low mechanism (if supported and enabled)
- Hardware and software must apply this algorithm to determine which entry in the FPB MEM High Vector applies to a given Memory address:
  - If the Memory address is below the value of FPB MEM High Vector Start, then the Memory address is out of range (below) and so is not associated with the Secondary side of the bridge by means of this mechanism, else
  - calculate the offset within the vector by first subtracting the value of FPB MEM High Vector Start, then dividing this according to the value of FPB MEM High Vector Granularity to determine the bit index within the vector.
  - If the bit index value is greater than the length indicated by FPB MEM High Vector Size Supported, then the Memory address is out of range (above) and so is not associated with the Secondary side of the bridge by means of this mechanism, else
  - if the bit value within the vector at the calculated bit index location is 1b, then the Memory address is associated with the Secondary side of the bridge, else the Memory address is associated with the Primary side of the bridge.

IMPLEMENTATION NOTE

FPB Address Decoding

FPB uses a bit vector mechanism to decode ranges of Routing IDs, and Memory Addresses above and below 4 GB. A bridge supporting FPB contains the following for each resource type/range where it supports the use of FPB:

- A Bit vector
- A Start Address
- A Granularity

These are used by the bridge to determine if a given address is part of the range decoded by FPB as associated with the secondary side of the bridge. An address that is determined not to be associated with the secondary side of the bridge using either or both of the non-FPB decode mechanisms and the FPB decode mechanisms is (by default) associated with the primary side of the bridge. Here, when we use the term "associated" we mean, for example, that the bridge will apply the following handling to TLPs:

- Associated with Primary, Received at Primary → Unsupported Request (UR)
- Associated with Primary, Received at Secondary → Forward upstream
- Associated with Secondary, Received at Primary → Forward downstream
- Associated with Secondary, Received at Secondary → Unsupported Request (UR)

In FPB, every bit in the vector represents a range of resources, where the size of that range is determined by the selected granularity. If a bit in the vector is Set, it indicates that TLPs addressed to an address within the corresponding range are to be associated with the secondary side of the bridge. The specific range of resources each bit represents is dependent on the index of that bit, and the values in the Start Address & Granularity. The Start Address indicates the lowest address described by the bit vector. The Granularity indicates the size of the region that is represented by each bit. Each successive bit in the vector applies to the subsequent range, increasing with each bit according to the Granularity.

For example, consider a bridge using FPB to describe a MEM Low range. FPB MEM Low Vector Start has been set to FC0h, indicating that the range described by the bit vector starts at address FC00 0000h. FPB MEM Low Vector Granularity has been set to 0000b, indicating that each bit represents a 1 MB range.

From these values we can determine that bit 0 of the vector represents a 1MB range starting at FC000 0000h (FC00 0000h-FC0F FFFFh), bit 1 represents FC10 0000h-FC1F FFFFh, etc.

Bits in the vector that are set to 0 indicate that the range is not included in the range described by FPB. In the above example, If bit 0 is Clear, packets addressed to anywhere between FC00 0000h and FC0F FFFFh should not be routed to the secondary bus of the bridge due to FPB.

IMPLEMENTATION NOTE

Hardware and Software Considerations for FPB

FPB is intended to address a class of issues with PCI/PCIe architecture that relate to resource allocation inefficiency. These issues can be categorized as "static" or "dynamic" use case scenarios, where static use cases refer to scenarios where resources are allocated at system boot and then typically not changed again, and dynamic use cases refer to scenarios where run-time resource rebalancing (e.g. allocation of new resources, freeing of resources no longer needed) is required, due to hot add/remove, or by other needs.

In the Static cases there are limits on the size of hierarchies and number of Endpoints due to the use of additional Bus Numbers and the lack of use of Device Numbers caused by the PCI/PCIe architectural definition for Switches and Downstream Ports. FPB addresses this class of problems by "flattening" the use of Routing IDs (RIDs) so that Switches and Downstream Ports are able to make more efficient use of the available RIDs.

For the Dynamic cases, without FPB, the "best known method" to avoid rebalancing has been to reserve large ranges of Bus Numbers and Memory Space in the bridge above the relevant Port or Endpoint such that hopefully any future needs can be satisfied within the pre-allocated ranges. This leads to potentially unused allocations, which makes the Routing ID issues worse, and in a resource constrained platform this approach is difficult to implement, even for relatively simple cases, where, for example, one might have an add-in card implementing a single Endpoint replaced by another add-in card that has a Switch and two Endpoints, so that although an initial allocation of just one Bus would have been sufficient, the initial allocation breaks immediately with the new add-in card.

For Memory Space the pre-allocation approach is problematic when hot-plugged Endpoints may require the allocation of Memory Space below 4 GB, which by its nature is a limited resource, which is quickly used up by pre-allocation of even relatively small amounts, and for which pre-allocation is unattractive because of the multiple system elements placing demands on system address space allocation below 4 GB.

FPB includes mechanisms to enable discontinuous resource range allocation/reallocation for both Requester IDs and Memory Space. The intent is to allow system software the ability to maintain resource "pools" which can be allocated (and freed back to) at run-time, without disrupting other operations in progress as is required with rebalancing.

To support the run time use of FPB by system software, FPB hardware implementations should avoid introducing stalls or other types of disruptions to transactions in flight, including during the times that system software is modifying the state of the FPB hardware. It is not, however, expected that hardware will attempt to identify cases where system software erroneously modifies the FPB configuration in a way that does affect transactions in flight. Just as with the non-FPB mechanisms, it is the responsibility of system software to ensure that system operation is not corrupted due to a reconfiguration operation.

It is not explicitly required that system firmware/software perform the enabling and/or disabling of FPB mechanisms in a particular sequence, however care should be taken to implement resource allocation operations in a hierarchy such that the hardware and software elements of the system are not corrupted or caused to fail.

6.28 Vital Product Data (VPD)

Vital Product Data (VPD) is the information that uniquely defines items such as the hardware, software, and microcode elements of a system. The VPD provides the system with information on various FRUs (Field Replaceable Unit) including Part Number, Serial Number, and other detailed information. VPD also provides a mechanism for storing information such as performance and failure data on the device being monitored. The objective, from a system point of view, is to collect this information by reading it from the hardware, software, and microcode components.

Support of VPD within add-in cards is optional depending on the manufacturer. Though support of VPD is optional, add-in card manufacturers are encouraged to provide VPD due to its inherent benefits for the add-in card, system manufacturers, and for Plug and Play.

The mechanism for accessing VPD is documented in Section 7.9.19.

VPD for PCI Express is unchanged from the definition in the *PCI Local Bus Specification*, Revision 3.0. That definition, in turn, was based on earlier versions of the *PCI Local Bus Specification* as well as the *Plug and Play ISA Specification,* Version 1.0a.

Vital Product Data is made up of Small and Large Resource Data Types.

Table 6-16: Small Resource Data Type Tag Bit Definitions

Offset	Field Name		
Byte 0	Value = 0xxx xyyyb		
	Bit 7	Small Resource Type	0b
	Bits 6:3	Small Item Name	xxxx
	Bits 2:0	Length in bytes	yy
Bytes 1 to n	Actual information		

Table 6-17: Large Resource Data Type Tag Bit Definitions

Offset	Field Name		
Byte 0	Value = 1xxx xxxxb		
	Bit 7	Large Resource Type	0b
	Bits 6:0	Large Item Name	xxxxxxx
Byte 1	Length in bytes of data items bits[7:0] (lsb)		
Byte 2	Length in bytes of data items bits[15:8] (msb)		
Bytes 3 to n	Actual data items		

The first VPD tag is the Identifier String (02h) and provides the product name of the device.

One VPD-R (10h) tag is used as a header for the read-only keywords. The VPD-R list (including tag and length) must checksum to zero. Attempts to write the read-only data will be executed as a no-op.

One VPD-W (11h) tag is used as a header for the read-write keywords. The storage component containing the read/write data is a non-volatile device that will retain the data when powered off.

The last tag must be the End Tag (0Fh).

A small example of the resource data type tags used in a typical VPD is shown in Table 6-18.

Table 6-18: Resource Data Type Flags for a Typical VPD

TAG 02h	Identifier String	Large Resource Data Type
TAG 10h	VPD-R list containing one or more VPD keywords	Large Resource Data Type
TAG 11h	VPD-W list containing one or more VPD keywords	Large Resource Data Type
TAG 0Fh	End Tag	Small Resource Data Type

6.28.1 VPD Format

Information fields within a VPD resource type consist of a three-byte header followed by some amount of data (see Figure 6-36). The three-byte header contains a two-byte keyword and a one-byte length. A keyword is a two-character (ASCII) mnemonic that uniquely identifies the information in the field. The last byte of the header is binary and represents the length value (in bytes) of the data that follows.

Keyword		Length	Data
Byte 0	Byte 1	Byte 2	Bytes 3 through n

Figure 6-36: VPD Format

VPD keywords are listed in two categories: read-only fields and read/write fields. Unless otherwise noted, keyword data fields are provided as ASCII characters. Use of ASCII allows keyword data to be moved across different enterprise computer systems without translation difficulty.

An example of the "add-in card serial number" VPD item is as follows:

Table 6-19: Example of Add-in Serial Card Number

Byte 0	53h "S"	Keyword: SN
Byte 1	4Eh "N"	
Byte 3	08h	Length: 8
Byte 4	30h "0"	Data: "00000194"
Byte 5	30h "0"	
Byte 6	30h "0"	
Byte 7	30h "0"	
Byte 8	30h "0"	
Byte 9	31h "1"	
Byte 10	39h "9"	
Byte 11	34h "4"	

6.28.2 VPD Definitions

This section describes the current VPD large and small resource data tags plus the VPD keywords. This list may be enhanced at any time. Companies wishing to define a new keyword should contact the PCISIG. All unspecified values are reserved for SIG assignment.

6.28.2.1 *VPD Large and Small Resource Data Tags*

VPD is contained in four types of Large and Small Resource Data Tags. The following tags and VPD keyword fields may be provided in PCI devices.

Table 6-20: VPD Large and Small Resource Data Tags

Tag	Description
Large resource type Identifier String Tag (02h)	This tag is the first item in the VPD storage component. It contains the name of the add-in card in alphanumeric characters.
Large resource type VPD-R Tag (10h)	This tag contains the read only VPD keywords for an add-in card.
Large resource type VPD-W Tag (11h)	This tag contains the read/write VPD keywords for an add-in card.
Small resource type End Tag (0Fh)	This tag identifies the end of VPD in the storage component.

6.28.2.2 *Read-Only Fields*

Table 6-21: VPD Read-Only Fields

Keyword	Name	Description
PN	Add-in Card Part Number	This keyword is provided as an extension to the Device ID (or Subsystem ID) in the Configuration Space header in Figure 6-36.
EC	Engineering Change Level of the Add-in Card	The characters are alphanumeric and represent the engineering change level for this add-in card.
FG	Fabric Geography	Reserved for legacy use by PICMIG specifications.
LC	Location	Reserved for legacy use by PICMIG specifications.
MN	Manufacture ID	This keyword is provided as an extension to the Vendor ID (or Subsystem Vendor ID) in the Configuration Space header in Figure 6-36. This allows vendors the flexibility to identify an additional level of detail pertaining to the sourcing of this device.
PG	PCI Geography	Reserved for legacy use by PICMIG specifications.
SN	Serial Number	The characters are alphanumeric and represent the unique add-in card Serial Number.
Vx	Vendor Specific	This is a vendor specific item and the characters are alphanumeric. The second character (x) of the keyword can be 0 through Z.
CP	Extended Capability	This field allows a new capability to be identified in the VPD area. Since dynamic control/status cannot be placed in VPD, the data for this field identifies where, in the device's memory or I/O address space, the control/status registers for the capability can be found. Location of the control/status registers is identified by providing the index (a value between 0 and 5) of the Base Address register that defines the address range that contains the registers, and the offset within that Base Address register range where the control/status registers reside. The data area for this field is four bytes long. The first byte contains the ID of the extended capability. The second byte contains the index (zero based) of the Base Address register used. The next two bytes contain the offset (in little-endian order) within that address range where the control/status registers defined for that capability reside.
RV	Checksum and Reserved	The first byte of this item is a checksum byte. The checksum is correct if the sum of all bytes in VPD (from VPD address 0 up to and including this byte) is zero. The remainder of this item is reserved space (as needed) to identify the last byte of read-only space. The read-write area does not have a checksum. This field is required.

6.28.2.3 *Read/Write Fields*

Table 6-22: VPD Read/Write Fields

Keyword	Name	Description
Vx	Vendor Specific	This is a vendor specific item and the characters are alphanumeric. The second character (x) of the keyword can be 0 through Z.
Yx	System Specific	This is a system specific item and the characters are alphanumeric. The second character (x) of the keyword can be 0 through 9 and B through Z.
YA	Asset Tag Identifier	This is a system specific item and the characters are alphanumeric. This keyword contains the system asset identifier provided by the system owner.
RW	Remaining Read/Write Area	This descriptor is used to identify the unused portion of the read/write space. The product vendor initializes this parameter based on the size of the read/write space or the space remaining following the Vx VPD items. One or more of the Vx, Yx, and RW items are required.

6.28.2.4 *VPD Example*

The following is an example of a typical VPD.

Table 6-23: VPD Example

Offset	Item Value
0	Large Resource Type ID String Tag (02h) 82h “Product Name”
1	Length 0021h
3	Data “ABCD Super-Fast Widget Controller”
36	Large Resource Type VPD-R Tag (10h) 90h
37	Length 0059h
39	VPD Keyword “PN”
41	Length 08h
42	Data “6181682A”
50	VPD Keyword “EC”
52	Length 0Ah
53	Data “4950262536”
63	VPD Keyword “SN”
65	Length 08h
66	Data “00000194”
74	VPD Keyword “MN”
76	Length 04h
77	Data “1037”

Offset	Item Value
81	VPD Keyword “RV”
83	Length 2Ch
84	Data Checksum
85	Data Reserved (00h)
128	Large Resource Type VPD-W Tag (11h) 91h
129	Length 007Ch
131	VPD Keyword “V1”
133	Length 05h
134	Data “65A01”
139	VPD Keyword “Y1”
141	Length 0Dh
142	Data “Error Code 26”
155	VPD Keyword “RW”
157	Length 61h
158	Data Reserved (00h)
255	Small Resource Type End Tag (0Fh) 78h

6.29 Native PCIe Enclosure Management

NPEM is an optional PCIe Extended Capability that provides mechanisms for enclosure management. This mechanism is designed to provide management for enclosures containing PCIe SSDs that is consistent with the established capabilities in the storage ecosystem.

This section defines the architectural aspects of the mechanism. The NPEM extended capability is defined in section 7.9.20.

An enclosure is any platform, box, rack, or set of boxes that contain one or more PCIe SSDs. The NPEM capability provides storage related enclosure control (e.g., status LED control) for a PCIe SSD. The NPEM capability may reside in a Downstream port, or an Endpoint (i.e., the PCIe SSD). Figure 6-37 shows an example configuration with a single Downstream Port containing the NPEM capability and vendor specific logic to control the associated LEDs.

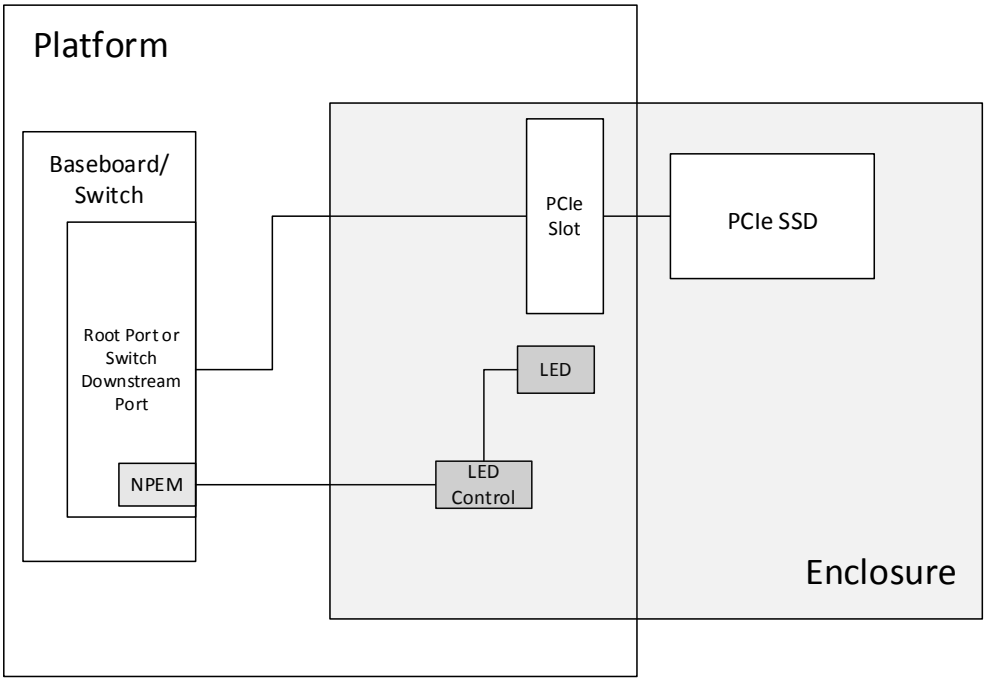


Figure 6-37: Example NPEM Configuration using a Downstream Port

Figure 6-38 shows an example configuration with the NPEM capability located in the Upstream Port (in this case, the SSD function).

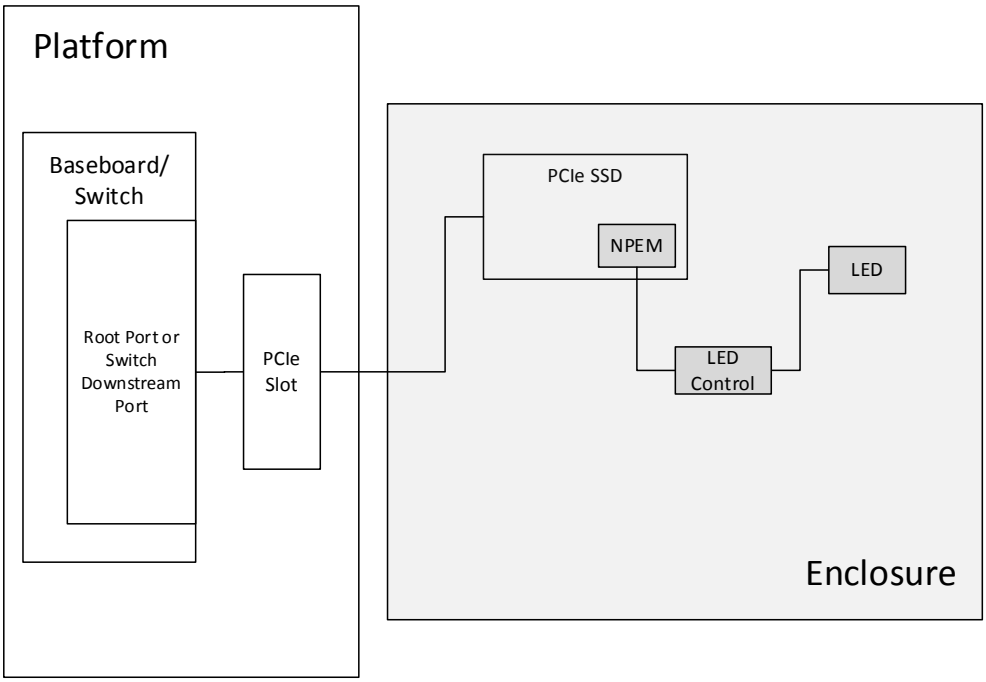


Figure 6-38: Example NPEM Configuration using an Upstream Port

Software issues an NPEM command by writing to the NPEM Control register to change the indications associated with an SSD. NPEM Command is a single write to the NPEM Control

register that changes the state of zero or more bits. NPEM indicates a successful completion to software using the command completed mechanism. Figure 6-39 shows the overall flow.

This specification defines the software interface provided by the NPEM capability. The Port to enclosure interface, enclosure, enclosure to LED interface, number of LEDs per SSD, and associated LED blink patterns are all outside the scope of this specification.

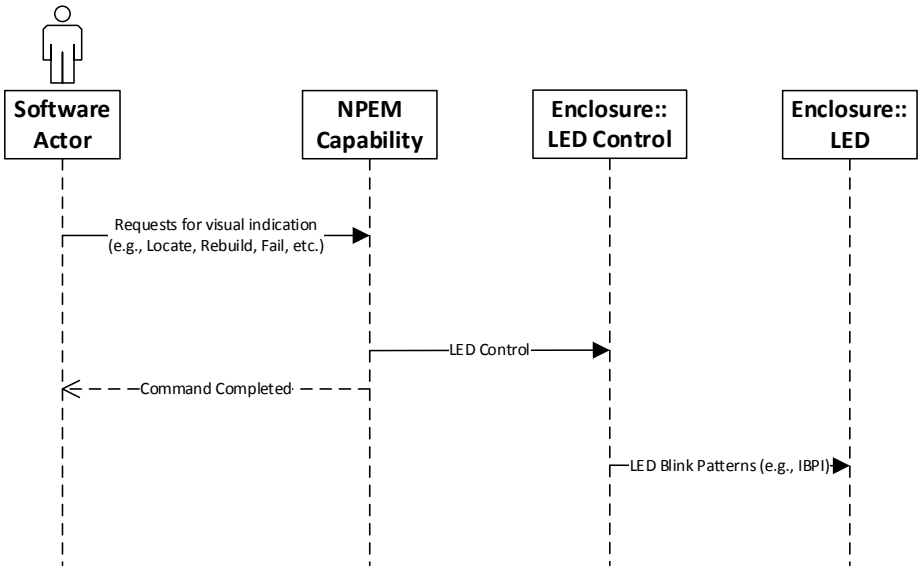


Figure 6-39: NPEM Command Flow

NPEM provides a mechanism for system software to issue a reset to the LED control element within the enclosure by means of the NPEM Reset mechanism, which is independent of the PCIe Link itself. The NPEM command completed mechanism also applies to NPEM Reset.

Storage system admin or software controls the indications for various device states through the NPEM capability.

IMPLEMENTATION NOTE

NPEM States

Table 6-24 shows an example of NPEM states and a possible meaning that some enclosures may assign to the architected NPEM states.

Table 6-24: NPEM States

NPEM State	Actor	Definition
OK	System Admin or Storage Software	OK state may mean the drive is functioning normally. This state may implicitly mean that an SSD is present, powered on, and working normally as seen by the software. A more granular indication of drive not physically present or present but not powered up are both outside the scope of this specification.
Locate	System Admin	Locate state may mean the specific drive is being identified by an admin.
Fail	Storage Software	Fail state may mean the drive is not functioning properly
Rebuild	Storage Software	Rebuild state may mean this drive is part of a multi-drive storage volume/array that is rebuilding or reconstructing data from redundancy on to this specific drive.
PFA	Storage Software	PFA stands for Predicted Failure Analysis. This state may mean the drive is still functioning normally but predicted to fail soon.
Hot Spare	Storage Software	Hot Spare state may mean this drive is marked to be automatically used as a replacement for a failed drive and contents of the failed drive may be rebuilt on this drive.
In A Critical Array	Storage Software	In A Critical Array state may mean the drive is part of a multi-drive storage array and that array is degraded.
In A Failed Array	Storage Software	NPEM In A Failed Array state may mean the drive is part of a multi-drive storage array and that array is failed.
Invalid Device Type	Storage Software	Invalid Device Type state may mean the drive is not the right type for the connector (e.g., An enclosure supports SAS and NVMe drives and this drive state indicates that a SAS drive is plugged into an NVMe slot).
Disabled	Storage Software	Disabled state may mean the drive in this slot is disabled. A removal of this drive from the slot may be safe. The power from this slot may be removed.

7

7 Software Initialization and Configuration

The PCI Express Configuration model supports two Configuration Space access mechanisms:

- PCI-compatible Configuration Access Mechanism (CAM)
- PCI Express Enhanced Configuration Access Mechanism (ECAM)

The PCI-compatible mechanism supports 100% binary compatibility with Conventional PCI or later aware operating systems and their corresponding bus enumeration and configuration software.

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

7.1 Configuration Topology

To maintain compatibility with PCI software configuration mechanisms, all PCI Express elements have a PCI-compatible Configuration Space. Each PCI Express Link originates from a logical PCI-PCI Bridge and is mapped into Configuration Space as the secondary bus of this Bridge. The Root Port is a PCI-PCI Bridge structure that originates a PCI Express Link from a PCI Express Root Complex (see Figure 7-1).

A PCI Express Switch not using FPB Routing ID mechanisms is represented by multiple PCI-PCI Bridge structures connecting PCI Express Links to an internal logical PCI bus (see Figure 7-2). The Switch Upstream Port is a PCI-PCI Bridge; the secondary bus of this Bridge represents the Switch's internal routing logic. Switch Downstream Ports are PCI-PCI Bridges bridging from the internal bus to buses representing the Downstream PCI Express Links from a PCI Express Switch. Only the PCI-PCI Bridges representing the Switch Downstream Ports may appear on the internal bus. Endpoints, represented by Type 0 Configuration Space headers, are not permitted to appear on the internal bus.

A PCI Express Endpoint is mapped into Configuration Space as a single Function in a Device, which might contain multiple Functions or just that Function. PCI Express Endpoints and Legacy Endpoints are required to appear within one of the Hierarchy Domains originated by the Root Complex, meaning that they appear in Configuration Space in a tree that has a Root Port as its head. Root Complex Integrated Endpoints (RCiEPs) and Root Complex Event Collectors do not appear within one of the Hierarchy Domains originated by the Root Complex. These appear in Configuration Space as peers of the Root Ports.

Unless otherwise specified, requirements in the Configuration Space definition for a device apply to single Function devices as well as to each Function individually of a Multi-Function Device.

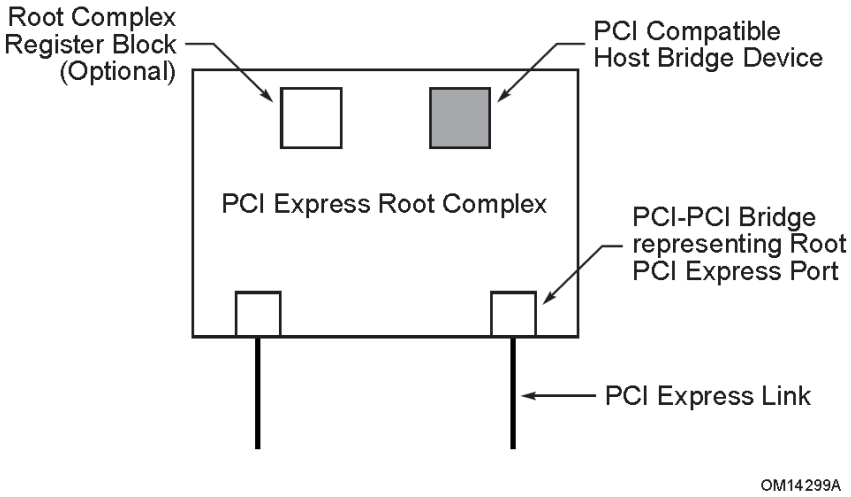


Figure 7-1: PCI Express Root Complex Device Mapping

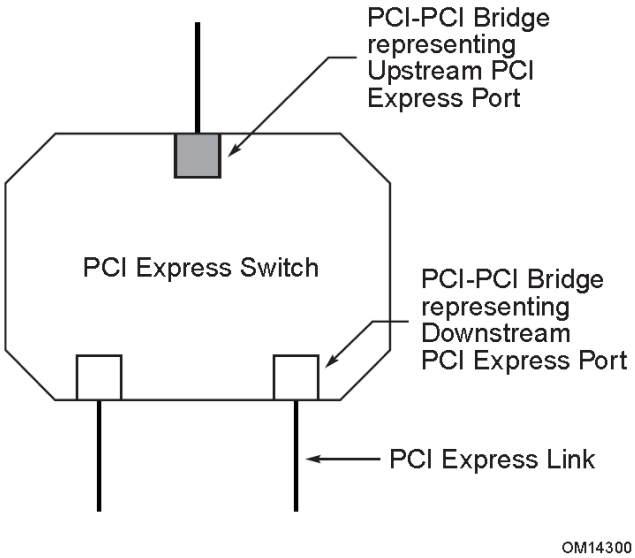


Figure 7-2: PCI Express Switch Device Mapping [116]

7.2 PCI Express Configuration Mechanisms

PCI Express extends the Configuration Space to 4096 bytes per Function as compared to 256 bytes allowed by *PCI Local Bus Specification*. PCI Express Configuration Space is divided into a PCI-compatible region, which consists of the first 256 bytes of a Function's Configuration Space, and a PCI Express Extended Configuration Space which consists of the remaining Configuration Space (see Figure 7-3). The PCI-compatible Configuration Space can be accessed using either the mechanism defined in the *PCI Local Bus Specification* or the PCI Express Enhanced Configuration Access Mechanism (ECAM) described later in this section. Accesses made using either access

[116] Future PCI Express Switches may be implemented as a single Switch device component (without the PCI-PCI Bridges) that is not limited by legacy compatibility requirements imposed by existing PCI software.

mechanism are equivalent. The PCI Express Extended Configuration Space can only be accessed by using the ECAM.[117]

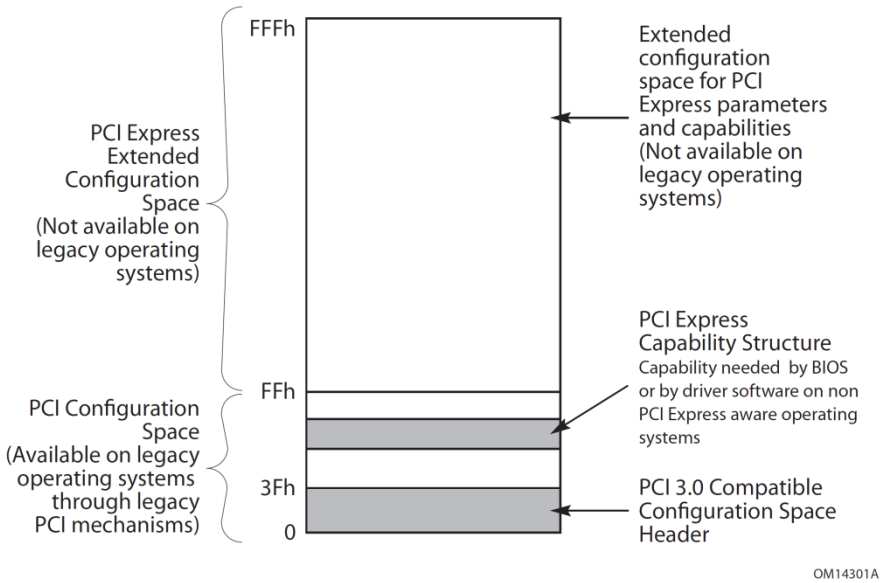


Figure 7-3: PCI Express Configuration Space Layout

7.2.1 PCI-compatible Configuration Mechanism

The PCI-compatible PCI Express Configuration Mechanism supports the PCI Configuration Space programming model defined in the *PCI Local Bus Specification*. By adhering to this model, systems incorporating PCI Express interfaces remain compliant with conventional PCI bus enumeration and configuration software.

In the same manner as PCI device Functions, PCI Express device Functions are required to provide a Configuration Space for software-driven initialization and configuration. Except for the differences described in this chapter, the PCI Express Configuration Space headers are organized to correspond with the format and behavior defined in the *PCI Local Bus Specification* (Section 6.1).

The PCI-compatible Configuration Access Mechanism uses the same Request format as the ECAM. For PCI-compatible Configuration Requests, the Extended Register Address field must be all zeros.

7.2.2 PCI Express Enhanced Configuration Access Mechanism (ECAM)

For systems that are PC-compatible, or that do not implement a processor-architecture-specific firmware interface standard that allows access to the Configuration Space, the ECAM is required as defined in this section.

---

[117] The ECAM operates independently from the mechanism defined in the *PCI Local Bus Specification* for generation of configuration transactions; there is no implied ordering between the two.

For systems that implement a processor-architecture-specific firmware interface standard that allows access to the Configuration Space, the operating system uses the standard firmware interface, and the hardware access mechanism defined in this section is not required. For example, for systems that are compliant with *Developer's Interface Guide for 64-bit Intel Architecture-based Servers (DIG64), Version 2.1*,[118] the operating system uses the SAL firmware service to access the Configuration Space.

In all systems, device drivers are encouraged to use the application programming interface (API) provided by the operating system to access the Configuration Space of its device and not directly use the hardware mechanism.

The ECAM utilizes a flat memory-mapped address space to access device configuration registers. In this case, the memory address determines the configuration register accessed and the memory data updates (for a write) or returns the contents of (for a read) the addressed register. The mapping from memory address space to PCI Express Configuration Space address is defined in Table 7-1.

The size and base address for the range of memory addresses mapped to the Configuration Space are determined by the design of the host bridge and the firmware. They are reported by the firmware to the operating system in an implementation-specific manner. The size of the range is determined by the number of bits that the host bridge maps to the Bus Number field in the configuration address. In Table 7-1, this number of bits is represented as n, where $1 \leq n \leq 8$. A host bridge that maps n memory address bits to the Bus Number field supports Bus Numbers from 0 to $2^n - 1$, inclusive, and the base address of the range is aligned to a $2^{(n+20)}$-byte memory address boundary. Any bits in the Bus Number field that are not mapped from memory address bits must be Clear.

For example, if a system maps three memory address bits to the Bus Number field, the following are all true:

- $n = 3$.
- Address bits A[63:23] are used for the base address, which is aligned to a 2^{23}-byte (8-MB) boundary.
- Address bits A[22:20] are mapped to bits [2:0] in the Bus Number field.
- Bits [7:3] in the Bus Number field are set to Clear.
- The system is capable of addressing Bus Numbers between 0 and 7, inclusive.

A minimum of one memory address bit ($n = 1$) must be mapped to the Bus Number field. Systems are encouraged to map additional memory address bits to the Bus Number field as needed to support a larger number of buses. Systems that support more than 4 GB of memory addresses are encouraged to map eight bits of memory address ($n = 8$) to the Bus Number field. Note that in systems that include multiple host bridges with different ranges of Bus Numbers assigned to each host bridge, the highest Bus Number for the system is limited by the number of bits mapped by the host bridge to which the highest bus number is assigned. In such a system, the highest Bus Number assigned to a particular host bridge would be greater, in most cases, than the number of buses assigned to that host bridge. In other words, for each host bridge, the number of bits mapped to

---

[118] *Developer's Interface Guide for 64-bit Intel Architecture-based Servers (DIG64), Version 2.1*, January 2002, www.dig64.org

the Bus Number field, *n*, must be large enough that the highest Bus Number assigned to each particular bridge must be less than or equal to $2^n - 1$ for that bridge.

In some processor architectures, it is possible to generate memory accesses that cannot be expressed in a single Configuration Request, for example due to crossing a DW aligned boundary, or because a locked access is used. A Root Complex implementation is not required to support the translation to Configuration Requests of such accesses.

Table 7-1: Enhanced Configuration Address Mapping

Memory Address[119]	PCI Express Configuration Space
A[(20 + *n* – 1):20]	Bus Number 1 ≤ *n* ≤ 8
A[19:15]	Device Number
A[14:12]	Function Number
A[11:8]	Extended Register Number
A[7:2]	Register Number
A[1:0]	Along with size of the access, used to generate Byte Enables

Note: for Requests targeting Extended Functions in an ARI Device, A[19:12] represents the (8-bit) Function Number, which replaces the (5-bit) Device Number and (3-bit) Function Number fields above.

The system hardware must provide a method for the system software to guarantee that a write transaction using the ECAM is completed by the completer before system software execution continues.

[119] This address refers to the byte-level address from a software point of view.

IMPLEMENTATION NOTE

Ordering Considerations for the Enhanced Configuration Access Mechanism

The ECAM converts memory transactions from the host CPU into Configuration Requests on the PCI Express fabric. This conversion potentially creates ordering problems for the software, because writes to memory addresses are typically posted transactions but writes to Configuration Space are not posted on the PCI Express fabric.

Generally, software does not know when a posted transaction is completed by the completer. In those cases in which the software must know that a posted transaction is completed by the completer, one technique commonly used by the software is to read the location that was just written. For systems that follow the PCI ordering rules throughout, the read transaction will not complete until the posted write is complete. However, since the PCI ordering rules allow non-posted write and read transactions to be reordered with respect to each other, the CPU must wait for a non-posted write to complete on the PCI Express fabric to be guaranteed that the transaction is completed by the completer.

As an example, software may wish to configure a device Function's Base Address register by writing to the device using the ECAM, and then read a location in the memory-mapped range described by this Base Address register. If the software were to issue the memory-mapped read before the ECAM write was completed, it would be possible for the memory-mapped read to be re-ordered and arrive at the device before the Configuration Write Request, thus causing unpredictable results.

To avoid this problem, processor and host bridge implementations must ensure that a method exists for the software to determine when the write using the ECAM is completed by the completer.

This method may simply be that the processor itself recognizes a memory range dedicated for mapping ECAM accesses as unique, and treats accesses to this range in the same manner that it would treat other accesses that generate non-posted writes on the PCI Express fabric, i.e., that the transaction is not posted from the processor's viewpoint. An alternative mechanism is for the host bridge (rather than the processor) to recognize the memory-mapped Configuration Space accesses and not to indicate to the processor that this write has been accepted until the non-posted Configuration Transaction has completed on the PCI Express fabric. A third alternative would be for the processor and host bridge to post the memory-mapped write to the ECAM and for the host bridge to provide a separate register that the software can read to determine when the Configuration Write Request has completed on the PCI Express fabric. Other alternatives are also possible.

IMPLEMENTATION NOTE

Generating Configuration Requests

Because Root Complex implementations are not required to support the generation of Configuration Requests from accesses that cross DW boundaries, or that use locked semantics, software should take care not to cause the generation of such accesses when using the memory-mapped ECAM unless it is known that the Root Complex implementation being used will support the translation.

7.2.2.1 *Host Bridge Requirements*

For those systems that implement the ECAM, the PCI Express Host Bridge is required to translate the memory-mapped PCI Express Configuration Space accesses from the host processor to PCI Express configuration transactions. The use of Host Bridge PCI class code is Reserved for backwards compatibility; host Bridge Configuration Space is opaque to standard PCI Express software and may be implemented in an implementation specific manner that is compatible with PCI Host Bridge Type 0 Configuration Space. A PCI Express Host Bridge is not required to signal errors through a Root Complex Event Collector. This support is optional for PCI Express Host Bridges.

7.2.2.2 *PCI Express Device Requirements*

Devices must support an additional 4 bits for decoding configuration register access, i.e., they must decode the Extended Register Address[3:0] field of the Configuration Request header.

IMPLEMENTATION NOTE

Device-Specific Registers in Configuration Space

It is strongly recommended that PCI Express devices place no registers in Configuration Space other than those in headers or Capability structures architected by applicable PCI specifications.

Device-specific registers that have legitimate reasons to be placed in Configuration Space (e.g., they need to be accessible before Memory Space is allocated) should be placed in a Vendor-Specific Capability structure (Section 7.9.4), a Vendor-Specific Extended Capability structure (Section 7.9.5, or Section 7.9.6).

Device-specific registers accessed in the run-time environment by drivers should be placed in Memory Space that is allocated by one or more Base Address registers. Even though PCI-compatible or PCI Express Extended Configuration Space may have adequate room for run-time device-specific registers, placing them there is highly discouraged for the following reasons:

- Not all Operating Systems permit drivers to access Configuration Space directly.
- Some platforms provide access to Configuration Space only via firmware calls, which typically have substantially lower performance compared to mechanisms for accessing Memory Space.
- Even on platforms that provide direct access to a memory-mapped PCI Express Enhanced Configuration Mechanism, performance for accessing Configuration Space will typically be significantly lower than for accessing Memory Space since:
    - Configuration Reads and Writes must usually be DWORD or smaller in size,
    - Configuration Writes are usually not posted by the platform, and
    - Some platforms support only one outstanding Configuration Write at a time.

7.2.3 Root Complex Register Block

A Root Port or RCiEP may be associated with an optional 4096-byte block of memory mapped registers referred to as the Root Complex Register Block (RCRB). These registers are used in a manner similar to Configuration Space and can include PCI Express Extended Capabilities and other implementation specific registers that apply to the Root Complex. The structure of the RCRB is described in Section 7.6.2.

Multiple Root Ports or internal devices are permitted to be associated with the same RCRB. The RCRB memory-mapped registers must not reside in the same address space as the memory-mapped Configuration Space or Memory Space.

A Root Complex implementation is not required to support accesses to an RCRB that cross DWORD aligned boundaries or accesses that use locked semantics.

IMPLEMENTATION NOTE

Accessing Root Complex Register Block

Because Root Complex implementations are not required to support accesses to a RCRB that cross DW boundaries, or that use locked semantics, software should take care not to cause the generation of such accesses when accessing a RCRB unless the Root Complex will support the access.

7.3 Configuration Transaction Rules

7.3.1 Device Number

With non-ARI Devices, PCI Express components are restricted to implementing a single Device Number on their primary interface (Upstream Port), but are permitted to implement up to eight independent Functions within that Device Number. Each internal Function is selected based on decoded address information that is provided as part of the address portion of Configuration Request packets.

Except when FPB Routing ID mechanisms are used (see Section 6.27), Downstream Ports that do not have ARI Forwarding enabled must associate only Device 0 with the device attached to the Logical Bus representing the Link from the Port. Configuration Requests targeting the Bus Number associated with a Link specifying Device Number 0 are delivered to the device attached to the Link; Configuration Requests specifying all other Device Numbers (1-31) must be terminated by the Switch Downstream Port or the Root Port with an Unsupported Request Completion Status (equivalent to Master Abort in PCI).

Non-ARI Devices must not assume that Device Number 0 is associated with their Upstream Port, but must capture their assigned Device Number as discussed in Section 2.2.6.2. Non-ARI Devices must respond to all Type 0 Configuration Read Requests, regardless of the Device Number specified in the Request.

Switches, and components wishing to incorporate more than eight Functions at their Upstream Port, are permitted to implement one or more “virtual switches” represented by multiple Type 1 (PCI-PCI Bridge) Configuration Space headers as illustrated in Figure 7-2. These virtual switches

serve to allow fan-out beyond eight Functions. FPB provides a "flattening" mechanism that, when enabled, causes the virtual bridges of the Downstream Ports to appear in configuration space at RID addresses following the RID of the Upstream Port (see Section 6.27). Since Switch Downstream Ports are permitted to appear on any Device Number, in this case all address information fields (Bus, Device, and Function Numbers) must be completely decoded to access the correct register. Any Configuration Request targeting an unimplemented Bus, Device, or Function must return a Completion with Unsupported Request Completion Status.

With an ARI Device, its Device Number is implied to be 0 rather than specified by a field within an ID. The traditional 5-bit Device Number and 3-bit Function Number fields in its associated Routing IDs, Requester IDs, and Completer IDs are interpreted as a single 8-bit Function Number. See Section 6.13. Any Type 0 Configuration Request targeting an unimplemented Function in an ARI Device must be handled as an Unsupported Request.

If an ARI Downstream Port has ARI Forwarding enabled, the logic that determines when to turn a Type 1 Configuration Request into a Type 0 Configuration Request no longer enforces a restriction on the traditional Device Number field being 0.

The following section provides details of the Configuration Space addressing mechanism.

7.3.2 Configuration Transaction Addressing

PCI Express Configuration Requests use the following addressing fields:

- Bus Number – PCI Express maps logical PCI Bus Numbers onto PCI Express Links such that PCI-compatible configuration software views the Configuration Space of a PCI Express Hierarchy as a PCI hierarchy including multiple bus segments.
- Device Number – Device Number association is discussed in Section 7.3.1 and in Section 6.27. When an ARI Device is targeted and the Downstream Port immediately above it is enabled for ARI Forwarding, the Device Number is implied to be 0, and the traditional Device Number field is used instead as part of an 8-bit Function Number field. See Section 6.13.
- Function Number – PCI Express also supports Multi-Function Devices using the same discovery mechanism as PCI. A Multi-Function Device must fully decode the Function Number field. It is strongly recommended that a single-Function Device also fully decode the Function Number field. With ARI Devices, discovery and enumeration of Extended Functions require ARI-aware software. See Section 6.13.
- Extended Register Number and Register Number – Specify the Configuration Space address of the register being accessed (concatenated such that the Extended Register Number forms the more significant bits).

7.3.3 Configuration Request Routing Rules

For Endpoints, the following rules apply:

- If Configuration Request Type is 1,
    - Follow the rules for handling Unsupported Requests

- If Configuration Request Type is 0,
  - Determine if the Request addresses a valid local Configuration Space of an implemented Function
    - If so, process the Request
    - If not, follow rules for handling Unsupported Requests

For Root Ports, Switches, and PCI Express-PCI Bridges, the following rules apply:

- Propagation of Configuration Requests from Downstream to Upstream as well as peer-to-peer are not supported
  - Configuration Requests are initiated only by the Host Bridge
- If Configuration Request Type is 0,
  - Determine if the Request addresses a valid local Configuration Space of an implemented Function
    - If so, process the Request
    - If not, follow rules for handling Unsupported Requests
- If Configuration Request Type is 1, apply the following tests, in sequence, to the Bus Number and Device Number fields:
  - If in the case of a PCI Express-PCI Bridge, equal to the Bus Number assigned to secondary PCI bus or, in the case of a Switch or Root Complex, equal to the Bus Number and decoded Device Numbers assigned to one of the Root (Root Complex) or Downstream Ports (Switch), or if required based on the FPB Routing ID mechanism,
    - Transform the Request to Type 0 by changing the value in the Type[4:0] field of the Request (see Table 2-3) – all other fields of the Request remain unchanged
    - Forward the Request to that Downstream Port (or PCI bus, in the case of a PCI Express-PCI Bridge)
  - If not equal to the Bus Number of any of Downstream Ports or secondary PCI bus, but in the range of Bus Numbers assigned to either a Downstream Port or a secondary PCI bus, or if required based on the FPB Routing ID mechanism,
    - Forward the Request to that Downstream Port interface without modification
  - Else (none of the above)
    - The Request is invalid – follow the rules for handling Unsupported Requests
- PCI Express-PCI Bridges must terminate as Unsupported Requests any Configuration Requests for which the Extended Register Address field is non-zero that are directed towards a PCI bus that does not support Extended Configuration Space.

Note: This type of access is a consequence of a programming error.

Additional rule specific to Root Complexes:

- Configuration Requests addressing Bus Numbers assigned to devices within the Root Complex are processed by the Root Complex
  - The assignment of Bus Numbers to the devices within a Root Complex may be done in an implementation specific way.

For all types of devices:

All other Configuration Space addressing fields are decoded according to the *PCI Local Bus Specification*.

7.3.4 PCI Special Cycles

PCI Special Cycles (see the *PCI Local Bus Specification* for details) are not directly supported by PCI Express. PCI Special Cycles may be directed to PCI bus segments behind PCI Express-PCI Bridges using Type 1 configuration cycles as described in the *PCI Local Bus Specification*.

7.4 Configuration Register Types

Configuration register fields are assigned one of the attributes described in Table 7-2. All PCI Express components, with the exception of the Root Complex and system-integrated devices, initialize register fields to specified default values. Root Complexes and system-integrated devices initialize register fields as required by the firmware for a particular system implementation.

Table 7-2: Register and Register Bit-Field Types

Register Attribute	Description
HwInit	**Hardware Initialized** – Register bits are initialized by firmware or hardware mechanisms such as pin strapping or serial EEPROM. (System firmware hardware initialization is only allowed for system-integrated devices.) Bits are fixed in value and read-only after initialization and can only change in value or be reset (for write-once by firmware) with Fundamental Reset (see Section 6.6.1). HwInit register bits are not modified by an FLR.
RO	**Read-only** – Register bits are read-only and cannot be altered by software. Register bits are permitted to be initialized by hardware mechanisms such as pin strapping or serial EEPROM. If the optional feature that would Set the bit is not implemented, the bit is hardwired to 0b.
RW	**Read-Write** – Register bits are read-write and are permitted to be either Set or Cleared by software to the desired state.
RW1C	**Write-1-to-clear status** – Register bits indicate status when read. A Set bit indicates a status event which is Cleared by writing a 1b. Writing a 0b to RW1C bits has no effect. If the optional feature that would Set the bit is not implemented, the bit is read-only and hardwired to 0b.

Register Attribute	Description
ROS	**Sticky - Read-only** – Register bits are read-only and cannot be altered by software. If the optional feature that would Set the bit is not implemented, the bit is hardwired to 0b. Bits are neither initialized nor modified by hot reset or FLR. Where noted, devices that consume Aux power must preserve sticky register bit values when Aux power consumption (via either Aux Power PM Enable or PME_En) is enabled. In these cases, register bits are neither initialized nor modified by hot, warm, or cold reset (see Section 6.6).
RWS	**Sticky - Read-Write** – Register bits are read-write and are Set or Cleared by software to the desired state. Bits are neither initialized nor modified by hot reset or FLR. Where noted, devices that consume Aux power must preserve sticky register bit values when Aux power consumption (via either Aux Power PM Enable or PME_En) is enabled. In these cases, register bits are neither initialized nor modified by hot, warm, or cold reset (see Section 6.6).
RW1CS	**Sticky - Write-1-to-clear status** – Register bits indicate status when read. A Set bit indicates a status event which is Cleared by writing a 1b. Writing a 0b to RW1CS bits has no effect. If the optional feature that would Set the bit is not implemented, the bit is read-only and hardwired to 0b. Bits are neither initialized nor modified by hot reset or FLR. Where noted, devices that consume Aux power must preserve sticky register bit values when Aux power consumption (via either Aux Power PM Enable or PME_En) is enabled. In these cases, register bits are neither initialized nor modified by hot, warm, or cold reset (see Section 6.6).
RsvdP	**Reserved and Preserved** – Reserved for future RW implementations. Register bits are read-only and must return zero when read. Software must preserve the value read for writes to bits.
RsvdZ	**Reserved and Zero** – Reserved for future RW1C implementations. Register bits are read-only and must return zero when read. Software must use 0b for writes to bits.

7.5 PCI and PCIe Capabilities Required by the Base Spec for all Ports

The following registers and capabilities are required by this specification in all Functions.

7.5.1 PCI-Compatible Configuration Registers

The first 256 bytes of a Function's Configuration Space form the PCI-compatible region. This region completely aliases the conventional PCI Configuration Space of the Function. Legacy PCI devices can also be accessed with the ECAM without requiring any modifications to the device hardware or device driver software.

Layout of the Configuration Space and format of individual configuration registers are depicted following the little-endian convention.

7.5.1.1 *Type 0/1 Common Configuration Space*

Figure 7-4 details allocation for common register fields of Type 0 and Type 1 Configuration Space headers for PCI Express device Functions.

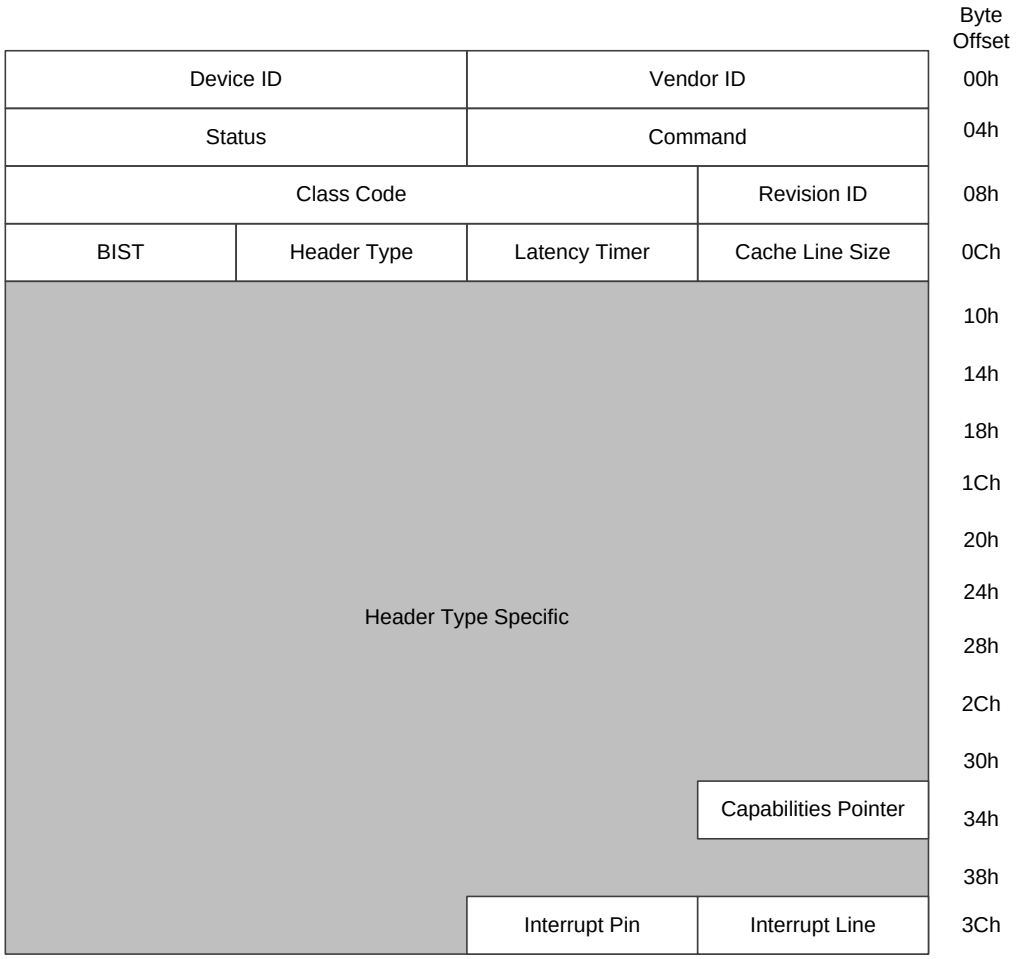


Figure 7-4: Common Configuration Space Header

These registers are defined for both Type 0 and Type 1 Configuration Space headers. The PCI Express-specific interpretation of these registers is defined in this section.

7.5.1.1.1 Vendor ID Register (Offset 00h)

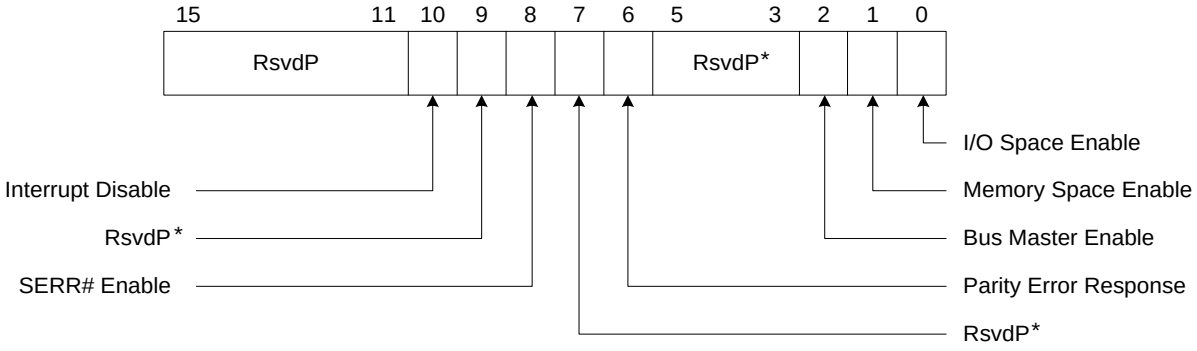
The Vendor ID register is HwInit and the value in this register identifies the manufacturer of the Function. In keeping with PCI-SIG procedures, valid vendor identifiers must be allocated by the PCI-SIG to ensure uniqueness. Each vendor must have at least one Vendor ID. It is recommended that software read the Vendor ID register to determine if a Function is present, where a value of FFFFh indicates that no Function is present.

7.5.1.1.2 Device ID Register (Offset 02h)

The Device ID register is HwInit and the value in this register identifies the particular Function. The Device ID must be allocated by the vendor. The Device ID, in conjunction with the Vendor ID and Revision ID, are used as one mechanism for software to determine which driver should be loaded. The vendor must ensure that the chosen values do not result in the use of an incompatible device driver.

7.5.1.1.3 Command Register (Offset 04h)

Table 7-3 defines the Command register and the layout of the register is shown in Figure 7-5. Individual bits in the Command register may or may not be implemented depending on the feature set supported by the Function. For PCI Express to PCI/PCI-X Bridges, refer to the *PCI Express to PCI/PCI-X Bridge Specification* for requirements for this register.



* These bits were used in previous versions of the programming model. Careful consideration should be given to any attempt to repurpose them.

Figure 7-5: Command Register

Table 7-3: Command Register

Bit Location	Register Description	Attributes
0	**I/O Space Enable** - Controls a Function's response to I/O Space accesses. When this bit is Clear, all received I/O accesses are caused to be handled as Unsupported Requests. When this bit is Set, the Function is enabled to decode the address and further process I/O Space accesses. For a Function with a Type 1 Configuration Space header, this bit controls the response to I/O Space accesses received on its Primary Side. Default value of this bit is 0b. This bit is permitted to be hardwired to 0b if a Function does not support I/O Space accesses.	RW
1	**Memory Space Enable** - Controls a Function's response to Memory Space accesses. When this bit is Clear, all received Memory Space accesses are caused to be handled as Unsupported Requests. When this bit is Set, the Function is enabled to decode the address and further process Memory Space accesses. For a Function with a Type 1 Configuration Space header, this bit controls the response to Memory Space accesses received on its Primary Side. Default value of this bit is 0b. This bit is permitted to be hardwired to 0b if a Function does not support Memory Space accesses.	RW

Bit Location	Register Description	Attributes
2	**Bus Master Enable** – Controls the ability of a Function to issue Memory[120] and I/O Read/Write Requests, and the ability of a Port to forward Memory and I/O Read/Write Requests in the Upstream direction *Functions with a Type 0 Configuration Space header:* When this bit is Set, the Function is allowed to issue Memory or I/O Requests. When this bit is Clear, the Function is not allowed to issue any Memory or I/O Requests. Note that as MSI/MSI-X interrupt Messages are in-band memory writes, setting the Bus Master Enable bit to 0b disables MSI/MSI-X interrupt Messages as well. Requests other than Memory or I/O Requests are not controlled by this bit. Default value of this bit is 0b. This bit is hardwired to 0b if a Function does not generate Memory or I/O Requests. *Functions with a Type 1 Configurations Space header:* This bit controls forwarding of Memory or I/O Requests by a Port in the Upstream direction. When this bit is 0b, Memory and I/O Requests received at a Root Port or the Downstream side of a Switch Port must be handled as Unsupported Requests (UR), and for Non-Posted Requests a Completion with UR Completion Status must be returned. This bit does not affect forwarding of Completions in either the Upstream or Downstream direction. The forwarding of Requests other than Memory or I/O Requests is not controlled by this bit. Default value of this bit is 0b.	RW
3	**Special Cycle Enable** – This bit was originally described in the *PCI Local Bus Specification*. Its functionality does not apply to PCI Express and the bit must be hardwired to 0b.	RO
4	**Memory Write and Invalidate** – This bit was originally described in the *PCI Local Bus Specification* and the *PCI-to-PCI Bridge Architecture Specification*. Its functionality does not apply to PCI Express and the bit must be hardwired to 0b. For PCI Express to PCI/PCI-X Bridges, refer to the *PCI Express to PCI/PCI-X Bridge Specification* for requirements for this register.	RO
5	**VGA Palette Snoop** – This bit was originally described in the *PCI Local Bus Specification* and the *PCI-to-PCI Bridge Architecture Specification*. Its functionality does not apply to PCI Express and the bit must be hardwired to 0b.	RO

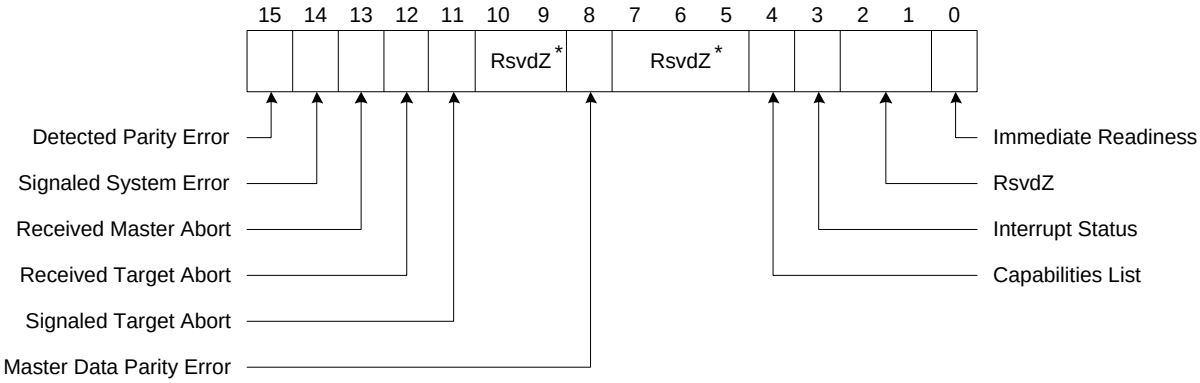
[120] The AtomicOp Requester Enable bit in the Device Control 2 register must also be Set in order for an AtomicOp Requester to initiate AtomicOp Requests, which are Memory Requests.

Bit Location	Register Description	Attributes
6	**Parity Error Response** – See Section 7.5.1.1.14. This bit controls the logging of poisoned TLPs in the Master Data Parity Error bit in the Status register. An RCiEP that is not associated with a Root Complex Event Collector is permitted to hardwire this bit to 0b. Default value of this bit is 0b.	RW
7	**IDSEL Stepping/Wait Cycle Control** – This bit was originally described in the *PCI Local Bus Specification*. Its functionality does not apply to PCI Express and the bit must be hardwired to 0b.	RO
8	**SERR# Enable** – See Section 7.5.1.1.14. When Set, this bit enables reporting upstream of Non-fatal and Fatal errors detected by the Function. Note that errors are reported if enabled either through this bit or through the PCI Express specific bits in the Device Control register (see Section 7.5.3.4). In addition, for Functions with Type 1 Configuration Space headers, this bit controls transmission by the primary interface of ERR_NONFATAL and ERR_FATAL error Messages forwarded from the secondary interface. This bit does not affect the transmission of forwarded ERR_COR messages. An RCiEP that is not associated with a Root Complex Event Collector is permitted to hardwire this bit to 0b. Default value of this bit is 0b.	RW
9	**Fast Back-to-Back Transactions Enable** – This bit was originally described in the *PCI Local Bus Specification*. Its functionality does not apply to PCI Express and the bit must be hardwired to 0b.	RO

Bit Location	Register Description	Attributes
10	**Interrupt Disable** – Controls the ability of a Function to generate INTx emulation interrupts. When Set, Functions are prevented from asserting INTx interrupts. Any INTx emulation interrupts already asserted by the Function must be deasserted when this bit is Set. As described in Section 2.2.8.1, INTx interrupts use virtual wires that must, if asserted, be deasserted using the appropriate Deassert_INTx message(s) when this bit is Set. Only the INTx virtual wire interrupt(s) associated with the Function(s) for which this bit is Set are affected. For Functions with a Type 0 Configuration Space header that generate INTx interrupts, this bit is required. For Functions with a Type 0 Configuration Space header that do not generate INTx interrupts, this bit is optional. If not implemented, this bit must be hardwired to 0b. For Functions with a Type1 Configuration Space header that generate INTx interrupts on their own behalf, this bit is required. This bit has no effect on interrupts forwarded from the secondary side. For Functions with a Type1 Configuration Space header that do not generate INTx interrupts on their own behalf this bit is optional. If not implemented, this bit must be hardwired to 0b. Default value of this bit is 0b.	RW

7.5.1.1.4 Status Register (Offset 06h)

Table 7-4 defines the Status register and the layout of the register is shown in Figure 7-4. Functions may not need to implement all bits, depending on the feature set supported by the Function. For PCI Express to PCI/PCI-X Bridges, refer to the *PCI Express to PCI/PCI-X Bridge Specification* for requirements for this register.



* These bits were used in previous versions of the programming model. Careful consideration should be given to any attempt to repurpose them.

Figure 7-6: Status Register

Table 7-4: Status Register

Bit Location	Register Description	Attributes
0	**Immediate Readiness** – This optional bit, when Set, indicates the Function is guaranteed to be ready to successfully complete valid configuration accesses at any time following any reset that the host is capable of issuing Configuration Requests to this Function. When this bit is Set, for accesses to this Function, software is exempt from all requirements to delay configuration accesses following any type of reset, including but not limited to the timing requirements defined in Section 6.6. How this guarantee is established is beyond the scope of this document. It is permitted that system software/firmware provide mechanisms that supersede the indication provided by this bit, however such software/firmware mechanisms are outside the scope of this specification.	RO
3	**Interrupt Status** – When Set, indicates that an INTx emulation interrupt is pending internally in the Function. Note that INTx emulation interrupts forwarded by Functions with a Type 1 Configuration Space header from the secondary side are not reflected in this bit. Setting the Interrupt Disable bit has no effect on the state of this bit. Functions that do not generate INTx interrupts are permitted to hardwire this bit to 0b. Default value of this bit is 0b.	RO

Bit Location	Register Description	Attributes
4	**Capabilities List** – Indicates the presence of an Extended Capability list item. Since all PCI Express device Functions are required to implement the PCI Express Capability structure, this bit must be hardwired to 1b.	RO
5	**66 MHz Capable** – This bit was originally described in the *PCI Local Bus Specification*. Its functionality does not apply to PCI Express and the bit must be hardwired to 0b.	RO
7	**Fast Back-to-Back Transactions Capable** – This bit was originally described in the *PCI Local Bus Specification*. Its functionality does not apply to PCI Express and the bit must be hardwired to 0b.	RO
8	**Master Data Parity Error** – See Section 7.5.1.1.14 This bit is Set by a Function with a Type 0 Configuration Space header if the Parity Error Response bit in the Command register is 1b and either of the following two conditions occurs: • Function receives a Poisoned Completion • Function transmits a Poisoned Request This bit is Set by a Function with a Type 1 Configuration Space header if the Parity Error Response bit in the Command register is 1b and either of the following two conditions occurs: • Port receives a Poisoned Completion going Downstream • Port transmits a Poisoned Request Upstream If the Parity Error Response bit is 0b, this bit is never Set. Default value of this bit is 0b.	RW1C
10:9	**DEVSEL Timing** – This field was originally described in the *PCI Local Bus Specification*. Its functionality does not apply to PCI Express and the field must be hardwired to 00b.	RO
11	**Signaled Target Abort** – See Section 7.5.1.1.14. This bit is Set when a Function completes a Posted or Non-Posted Request as a Completer Abort error. This applies to a Function with a Type 1 Configuration Space header when the Completer Abort was generated by its Primary Side. Functions with a Type 0 Configuration Space header that do not signal Completer Abort are permitted to hardwire this bit to 0b. Default value of this bit is 0b.	RW1C

Bit Location	Register Description	Attributes
12	**Received Target Abort** – See Section 7.5.1.1.14. This bit is Set when a Requester receives a Completion with Completer Abort Completion Status. On a Function with a Type 1 Configuration Space header, the bit is Set when the Completer Abort is received by its Primary Side. Functions with a Type 0 Configuration Space header that do not make Non-Posted Requests on their own behalf are permitted to hardwire this bit to 0b. Default value of this bit is 0b.	RW1C
13	**Received Master Abort** – See Section 7.5.1.1.14. This bit is Set when a Requester receives a Completion with Unsupported Request Completion Status. On a Function with a Type 1 Configuration Space header, the bit is Set when the Unsupported Request is received by its Primary Side. Functions with a Type 0 Configuration Space header that do not make Non-Posted Requests on their own behalf are permitted to hardwire this bit to 0b. Default value of this bit is 0b.	RW1C
14	**Signaled System Error** – See Section 7.5.1.1.14. This bit is Set when a Function sends an ERR_FATAL or ERR_NONFATAL Message, and the SERR# Enable bit in the Command register is 1b. Functions with a Type 0 Configuration Space header that do not send ERR_FATAL or ERR_NONFATAL Messages are permitted to hardwire this bit to 0b. Default value of this bit is 0b.	RW1C
15	**Detected Parity Error** – See Section 7.5.1.1.14. This bit is Set by a Function whenever it receives a Poisoned TLP, regardless of the state the Parity Error Response bit in the Command register. On a Function with a Type 1 Configuration Space header, the bit is Set when the Poisoned TLP is received by its Primary Side. Default value of this bit is 0b.	RW1C

7.5.1.1.5 Revision ID Register (Offset 08h)

The Revision ID register is HwInit and the value in this register specifies a Function specific revision identifier. The value is chosen by the vendor. Zero is an acceptable value. The Device ID, in conjunction with the Vendor ID and Revision ID, are used as one mechanism for software to determine which driver should be loaded. The vendor must ensure that the chosen values do not result in the use of an incompatible device driver.

7.5.1.1.6 Class Code Register (Offset 09h)

The Class Code register is read-only and is used to identify the generic operation of the Function and, in some cases, a specific register level programming interface. The register layout is shown in Figure 7-7 and described in Table 7-5. Encodings for base class, sub-class, and programming interface are provided in the *PCI Code and ID Assignment Specification*. All unspecified encodings are Reserved.

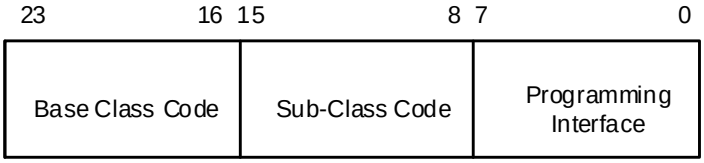


Figure 7-7: Class Code Register

Table 7-5: Header Type Register

Bit Location	Register Description	Attributes
7:0	**Programming Interface** - This field identifies a specific register-level programming interface (if any) so that device independent software can interact with the Function. Encodings for this field are provided in the *PCI Code and ID Assignment Specification*. All unspecified encodings are Reserved.	RO
15:8	**Sub-Class Code** – Specifies a base class sub-class, which identifies more specifically the operation of the Function. Encodings for sub-class are provided in the *PCI Code and ID Assignment Specification*. All unspecified encodings are Reserved.	RO
23:16	**Base Class Code** – A code that broadly classifies the type of operation the Function performs. Encodings for base class, are provided in the *PCI Code and ID Assignment Specification*. All unspecified encodings are Reserved.	RO

7.5.1.1.7 Cache Line Size Register (Offset 0Ch)

The Cache Line Size register is programmed by the system firmware or the operating system to system cache line size. However, note that legacy PCI-compatible software may not always be able to program this register correctly especially in the case of Hot-Plug devices. This read-write register is implemented for legacy compatibility purposes but has no effect on any PCI Express device behavior. For PCI Express to PCI/PCI-X Bridges, refer to the *PCI Express to PCI/PCI-X Bridge Specification* for requirements for this register. The default value of this register is 00h.

7.5.1.1.8 Latency Timer Register (Offset 0Dh)

This register is also referred to as Primary Latency Timer for Type 1 Configuration Space header Functions. The Latency Timer was originally described in the *PCI Local Bus Specification* and the *PCI-*

to-PCI Bridge Architecture Specification. Its functionality does not apply to PCI Express. This register must be hardwired to 00h.

7.5.1.1.9 Header Type Register (Offset 0Eh)

This register identifies the layout of the second part of the predefined header (beginning at byte 10h in Configuration Space) and also whether or not the Device might contain multiple Functions. The register layout is shown in Figure 7-8 and Table 7-6 describes the bits in the register.

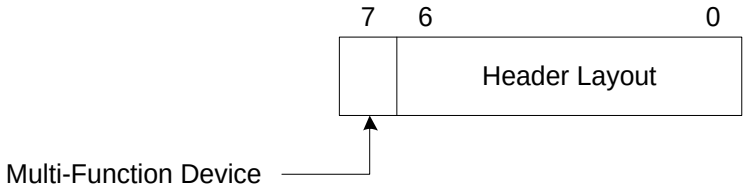


Figure 7-8: Header Type Register

Table 7-6: Header Type Register

Bit Location	Register Description	Attributes
6:0	**Header Layout** - This field identifies the layout of the second part of the predefined header. For Functions that implement a Type 0 Configuration Space header the encoding 000 0000b must be used. For Functions that implement a Type 1 Configuration Space header the encoding 000 0001b must be used. The encoding 000 0010b is Reserved. This encoding was originally described in the *PC Card Standard Electrical Specification* and is used in previous versions of the programming model. Careful consideration should be given to any attempt to repurpose it. All other encodings are Reserved.	RO
7	**Multi-Function Device** – When Set, indicates that the Device may contain multiple Functions, but not necessarily. Software is permitted to probe for Functions other than Function 0. When Clear, software must not probe for Functions other than Function 0 unless explicitly indicated by another mechanism, such as an ARI or SR-IOV Capability structure. Except where stated otherwise, it is recommended that this bit be Set if there are multiple Functions, and Clear if there is only one Function.	RO

7.5.1.1.10 BIST Register (Offset 0Fh)

This register is used for control and status of BIST. Functions that do not support BIST must hardwire the register to 00h. A Function whose BIST is invoked must not prevent normal operation of the PCI Express Link. Table 7-7 describes the bits in the register and Figure 7-9 shows the register layout.

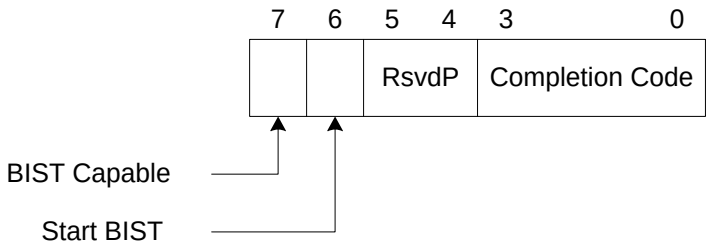


Figure 7-9: BIST Register

Table 7-7: BIST Register

Bit Location	Register Description	Attributes
3:0	**Completion Code** – This field encodes the status of the most recent test. A value of 0000b means that the Function has passed its test. Non-zero values mean the Function failed. Function-specific failure codes can be encoded in the non-zero values. This field's value is only meaningful when BIST Capable is Set and Start BIST is Clear. Default value of this field is 0000b. This field must be hardwired to 0000b if BIST Capable is Clear.	RO
6	**Start BIST** – If BIST Capable is Set, Set this bit to invoke BIST. The Function resets the bit when BIST is complete. Software is permitted to fail the device if this bit is not Clear (BIST is not complete) 2 seconds after it had been Set. Writing this bit to 0b has no effect. This bit must be hardwired to 0b if BIST Capable is Clear.	RW/RO (see description)
7	**BIST Capable** – When Set, this bit indicates that the Function supports BIST. When Clear, the Function does not support BIST.	HwInit

7.5.1.1.11 Capabilities Pointer (Offset 34h)

This register is used to point to a linked list of capabilities implemented by this Function. Since all PCI Express Functions are required to implement both the Power Management Capability and the PCI Express Capability structure, these structures must be included somewhere in the linked list; this register may point to either of these Capability Structures or to an optional Capability Structure implemented by the Function. The bottom two bits are Reserved and must be set to 00b. Software must mask these bits off before using this register as a pointer in Configuration Space to the first entry of a linked list of new capabilities.

7.5.1.1.12 Interrupt Line Register (Offset 3Ch)

The Interrupt Line register communicates interrupt line routing information. The register is read/write and must be implemented by any Function that uses an interrupt pin (see following description). Values in this register are programmed by system software and are system architecture

specific. The Function itself does not use this value; rather the value in this register is used by device drivers and operating systems.

7.5.1.1.13 Interrupt Pin Register (Offset 3Dh)

The Interrupt Pin register is a read-only register that identifies the legacy interrupt Message(s) the Function uses (see Section 6.1 for further details). Valid values are 01h, 02h, 03h, and 04h that map to legacy interrupt Messages for INTA, INTB, INTC, and INTD respectively. A value of 00h indicates that the Function uses no legacy interrupt Message(s). The values 05h through FFh are Reserved.

PCI Express defines one legacy interrupt Message for a single Function device and up to four legacy interrupt Messages for a Multi-Function Device. For a single Function device, only INTA may be used.

Any Function on a Multi-Function Device can use any of the INTx Messages. If a device implements a single legacy interrupt Message, it must be INTA; if it implements two legacy interrupt Messages, they must be INTA and INTB; and so forth. For a Multi-Function Device, all Functions may use the same INTx Message or each may have its own (up to a maximum of four Functions) or any combination thereof. A single Function can never generate an interrupt request on more than one INTx Message.

7.5.1.1.14 Error Registers

The Error Control/Status register bits in the Command and Status registers (see Section 7.5.1.1.3 and Section 7.5.1.1.4 respectively) and the Bridge Control and Secondary Status registers of Type 1 Configuration Space header Functions (see Section 7.5.1.3.10 and Section 7.5.1.3.7 respectively) control PCI-compatible error reporting for both PCI and PCI Express device Functions. Mapping of PCI Express errors onto PCI errors is also discussed in Section 6.2.7.1. In addition to the PCI-compatible error control and status, PCI Express error reporting may be controlled separately from PCI device Functions through the PCI Express Capability structure described in Section 7.5.2. The PCI-compatible error control and status register fields do not have any effect on PCI Express error reporting enabled through the PCI Express Capability structure. PCI Express device Functions may implement optional advanced error reporting as described in Section 7.8.4.

For PCI Express Root Ports represented by a Type 1 Configuration Space header:

- The primary side Error Control/Status registers apply to errors detected on the internal logic associated with the Root Complex.
- The secondary side Error Control/Status registers apply to errors detected on the Link originating from the Root Port.

For PCI Express Switch Upstream Ports represented by a Type 1 Configuration Space header:

- The primary side Error Control/Status registers apply to errors detected on the Upstream Link of the Switch.
- The secondary side Error Control/Status registers apply to errors detected on the internal logic of the Switch.

For PCI Express Switch Downstream Ports represented by a Type 1 Configuration Space header:

- The primary side Error Control/Status registers apply to errors detected on the internal logic of the Switch.
- The secondary side Error Control/Status registers apply to errors detected on the Downstream Link originating from the Switch Port.

7.5.1.2 *Type 0 Configuration Space Header*

Figure 7-10 details allocation for register fields of Type 0 Configuration Space header for PCI Express device Functions.

				Byte Offset
Device ID		Vendor ID		00h
Status		Command		04h
Class Code			Revision ID	08h
BIST	Header Type	Latency Timer	Cache Line Size	0Ch
Base Address Registers				10h 14h 18h 1Ch 20h 24h
Cardbus CIS Pointer				28h
Subsystem ID		Subsystem Vendor ID		2Ch
Expansion ROM Base Address				30h
Reserved			Capabilities Pointer	34h
Reserved				38h
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	3Ch

Figure 7-10: Type 0 Configuration Space Header

Section 7.5.1.1 details the PCI Express-specific registers that are valid for all Configuration Space header types. The PCI Express-specific interpretation of registers specific to Type 0 Configuration Space header is defined in this section.

7.5.1.2.1 Base Address Registers (Offset 10h - 24h)

System software must build a consistent address map before booting the machine to an operating system. This means it has to determine how much memory is in the system, and how much address space the Functions in the system require. After determining this information, system software can map the Functions into reasonable locations and proceed with system boot. In order to do this mapping in a device-independent manner, the base registers for this mapping are placed in the

predefined header portion of Configuration Space. It is strongly recommended that power-up firmware/software also support the optional Enhanced Configuration Access Mechanism (ECAM).

Bit 0 in all Base Address registers is read-only and used to determine whether the register maps into Memory or I/O Space. Base Address registers that map to Memory Space must return a 0b in bit 0 (see Figure 7-11). Base Address registers that map to I/O Space must return a 1b in bit 0 (see Figure 7-12).

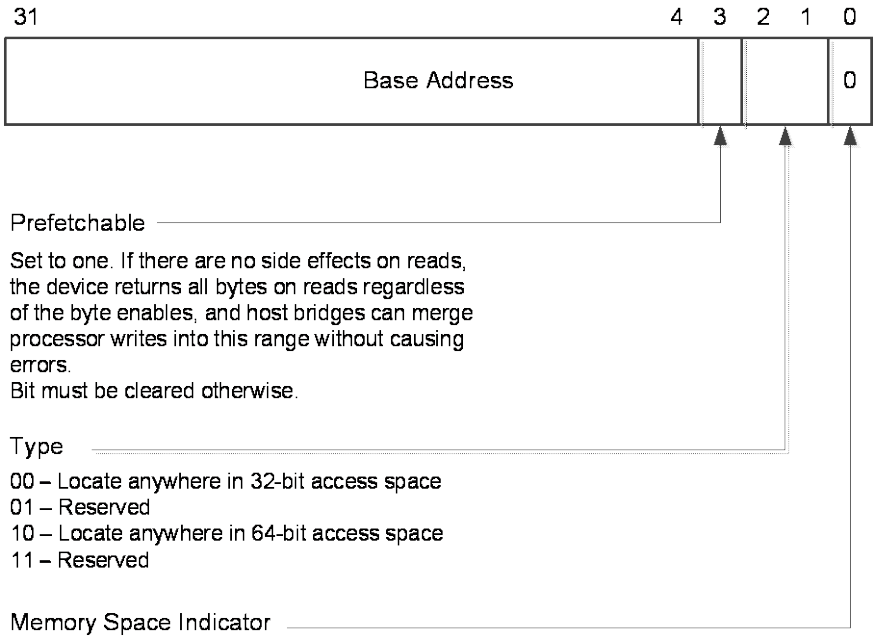


Figure 7-11: Base Address Register for Memory

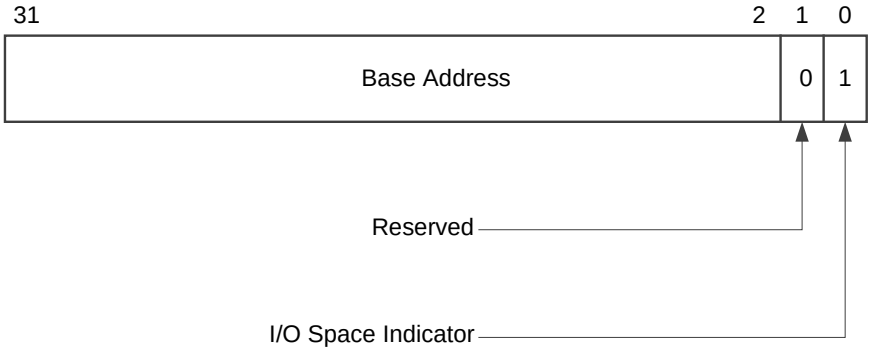


Figure 7-12: Base Register for I/O

Base Address registers that map into I/O Space are always 32 bits wide with bit 0 hardwired to 1b. Bit 1 is Reserved and must return 0b on reads and the other bits are used to map the Function into I/O Space.

Base Address registers that map into Memory Space can be 32 bits or 64 bits wide (to support mapping into a 64-bit address space) with bit 0 hardwired to 0b. For Memory Base Address registers, bits 2 and 1 have an encoded meaning as shown in Table 7-8. Bit 3 should be set to 1b if the data is prefetchable and set to 0b otherwise. A Function is permitted to mark a range as prefetchable if there are no side effects on reads, the Function returns all bytes on reads regardless

of the byte enables, and host bridges can merge processor writes into this range [121] without causing errors. Bits 3-0 are read-only.

Table 7-8: Memory Base Address Register Bits 2:1 Encoding

Bits 2:1(b)	Meaning
00	Base register is 32 bits wide and can be mapped anywhere in the 32 address bit Memory Space.
01	Reserved[122]
10	Base register is 64 bits wide and can be mapped anywhere in the 64 address bit Memory Space.
11	Reserved

The number of upper bits that a Function actually implements depends on how much of the address space the Function will respond to. A 32-bit Base Address register can be implemented to support a single memory size that is a power of 2 from 16 bytes to 2 GB. A Function that wants a 1 MB memory address space (using a 32-bit Base Address register) would build the top 12 bits of the address register, hardwiring the other bits to 0's. The attributes for some of the bits in the BAR are affected by the Resizable BAR Capability, if it is implemented.

Power-up software can determine how much address space the Function requires by writing a value of all 1's to the register and then reading the value back. The Function will return 0's in all don't-care address bits, effectively specifying the address space required. Unimplemented Base Address registers are hardwired to zero.

This design implies that all address spaces used are a power of two in size and are naturally aligned. Functions are free to consume more address space than required, but decoding down to a 4 KB space for memory is suggested for Functions that need less than that amount. For instance, a Function that has 64 bytes of registers to be mapped into Memory Space may consume up to 4 KB of address space in order to minimize the number of bits in the address decoder. Functions that do consume more address space than they use are not required to respond to the unused portion of that address space if the Function's programming model never accesses the unused space. The Function is permitted to return Unsupported Request for accesses targetting the unused locations. Functions that map control functions into I/O Space must not consume more than 256 bytes per I/O Base Address register or per each entry in the Enhanced Allocation capability. The upper 16 bits of the I/O Base Address register may be hardwired to zero for Functions intended for 16-bit I/O systems, such as PC compatibles. However, a full 32-bit decode of I/O addresses must still be done.

[121] Any device that has a range that behaves like normal memory should mark the range as prefetchable. A linear frame buffer in a graphics device is an example of a range that should be marked prefetchable.

[122] The encoding to support memory space below 1 MB was supported in an earlier version of the *PCI Local Bus Specification*. System software should recognize this encoding and handle it appropriately.

IMPLEMENTATION NOTE

Sizing a 32-bit Base Address Register Example

Decode (I/O or memory) of the appropriate address space is disabled via the Command register before sizing a Base Address register. Software saves the original value of the Base Address register, writes a value of all 1's to the register, then reads it back. Size calculation can be done from the 32 bit value read by first clearing encoding information bits (bits 1:0 for I/O, bits 3:0 for memory), inverting all 32 bits (logical NOT), then incrementing by 1. The resultant 32-bit value is the memory/I/O range size decoded by the register. Note that the upper 16 bits of the result is ignored if the Base Address register is for I/O and bits 31:16 returned zero upon read. The original value in the Base Address register is restored before re-enabling decode in the Command register of the Function.

64-bit (memory) Base Address registers can be handled the same, except that the second 32 bit register is considered an extension of the first (i.e., bits 63:32). Software writes a value of all 1's to both registers, reads them back, and combines the result into a 64-bit value. Size calculation is done on the 64-bit value.

A Type 0 Configuration Space Header has six DWORD locations allocated for Base Address registers starting at offset 10h in Configuration Space. A Type 1 Configurations Space Header has only two DWORD locations. A Function may use any of the locations to implement Base Address registers. An implemented 64-bit Base Address register consumes two consecutive DWORD locations. Software looking for implemented Base Address registers must start at offset 10h and continue upwards through offset 24h. A typical Function requires one memory range for its control functions. Some graphics Functions use two ranges, one for control functions and another for a frame buffer. A Function that wants to map control functions into both memory and I/O Spaces at the same time must implement two Base Address registers (one memory and one I/O). The driver for that Function might only use one space in which case the other space will be unused. Functions are recommended to always map control functions into Memory Space.

A PCI Express Function requesting Memory Space through a BAR must set the BAR's Prefetchable bit unless the range contains locations with read side effects or locations in which the Function does not tolerate write merging. It is strongly encouraged that resources mapped into Memory Space be designed as prefetchable whenever possible. PCI Express Functions other than Legacy Endpoints must support 64-bit addressing for any Base Address register that requests prefetchable Memory Space. The minimum Memory Space address range requested by a BAR is 128 bytes. The attributes for some of the bits in the BAR are affected by the Resizable BAR Capability, if it is implemented.

IMPLEMENTATION NOTE

Additional Guidance on the Prefetchable Bit in Memory Space BARs

PCI Express adapters with Memory Space BARs that request a large amount of non-prefetchable Memory Space (e.g., over 64 MB) may cause shortages of that Space on certain scalable platforms, since many platforms support a total of only 1 GB or less of non-prefetchable Memory Space. This may limit the number of such adapters that can be supported on those platforms. For this reason, it is especially encouraged for BARs requesting large amounts of Memory Space to have their Prefetchable bit Set, since prefetchable Memory Space is more bountiful on most scalable platforms.

While a Memory Space BAR is required to have its Prefetchable bit Set if none of the locations within its range have read side effects and all of the locations tolerate write merging, there are system configurations where having the Prefetchable bit Set will still allow correct operation even if those conditions are not met. For those cases, it may make sense for the adapter to have the Prefetchable bit Set in certain candidate BARs so that the system can map those BARs into prefetchable Memory Space in order to avoid non-prefetchable Memory Space shortages.

On PCI Express systems that meet the criteria enumerated below, setting the Prefetchable bit in a candidate BAR will still permit correct operation even if the BAR's range includes some locations that have read side-effects or cannot tolerate write merging. This is primarily due to the fact that PCI Express Memory Reads always contain an explicit length, and PCI Express Switches never prefetch or do byte merging. Generally only 64-bit BARs are good candidates, since only Legacy Endpoints are permitted to set the Prefetchable bit in 32-bit BARs, and most scalable platforms map all 32-bit Memory BARs into non-prefetchable Memory Space regardless of the Prefetchable bit value.

Here are criteria that are sufficient to guarantee correctness for a given candidate BAR:

- The entire path from the host to the adapter is over PCI Express.
- No conventional PCI or PCI-X devices do peer-to-peer reads to the range mapped by the BAR.
- The PCI Express Host Bridge does no byte merging. (This is believed to be true on most platforms.)
- Any locations with read side-effects are never the target of Memory Reads with the TH bit Set. See Section 2.2.5.
- The range mapped by the BAR is never the target of a speculative Memory Read, either Host initiated or peer-to-peer.

The above criteria are a simplified set that are sufficient to guarantee correctness. Other less restrictive but more complex criteria may also guarantee correctness, but are outside the scope of this specification.

7.5.1.2.2 Cardbus CIS Pointer (Offset 28h)

This register was originally described in the *PC Card Standard Electrical Specification*. Its functionality does not apply to PCI Express. It must be read-only and hardwired to 0000 0000h.

7.5.1.2.3 Subsystem Vendor ID/Subsystem ID (Offset 2Ch/2Eh)

The Subsystem Vendor ID and Subsystem ID registers are used to uniquely identify the adapter or subsystem where the PCI Express component resides. They provide a mechanism for vendors to distinguish their products from one another even though the assemblies may have the same PCI Express component on them (and, therefore, the same Vendor ID and Device ID).

Implementation of these registers is required for all Functions except those that have a Base Class 06h with Sub Class 00h-04h (00h, 01h, 02h, 03h, 04h), or a Base Class 08h with Sub Class 00h-03h (00h, 01h, 02h, 03h). Subsystem Vendor IDs can be obtained from the PCI SIG and are used to identify the vendor of the adapter, motherboard, or subsystem[123]. A Subsystem Vendor ID (SVID) must be a Vendor ID assigned by the PCI-SIG to the vendor of the subsystem. In keeping with PCI-SIG procedures, valid vendor identifiers must be obtained from the PCI-SIG to ensure uniqueness.

Values for the Subsystem ID are vendor assigned. Subsystem ID values, in conjunction with the Subsystem Vendor ID, form a unique identifier for the PCI product. Subsystem ID and Device ID values are distinct and unrelated to each other, and software should not assume any relationship between them.

Values in these registers must be loaded before the Function becomes Configuration-Ready. How these values are loaded is not specified but could be done during the manufacturing process or loaded from external logic (e.g., strapping options, serial ROMs, etc.). These values must not be loaded using Expansion ROM software because Expansion ROM software is not guaranteed to be run during POST in all systems.

[123] A company requires only one Vendor ID. That value can be used in either the Vendor ID register of Configuration Space (e.g., offset 00h) or the Subsystem Vendor ID register of Configuration Space (e.g., offset 2Ch). It is used in the Vendor ID register (e.g., offset 00h) if the company built the silicon. It is used in the Subsystem Vendor ID register (e.g., offset 2Ch) if the company built the assembly. If a company builds both the silicon and the assembly, then the same value would be used in both registers.

If a device is designed to be used exclusively on the system board, the system vendor may use system specific software to initialize these registers after each power-on.

IMPLEMENTATION NOTE

Subsystem Vendor ID and Subsystem ID

The Subsystem Vendor ID and Subsystem ID fields, taken together, allow software to uniquely identify a PCI circuit board product. Vendors should therefore not reuse Subsystem ID values across multiple product types that share a common Subsystem Vendor ID. It is acceptable, although not preferred, to reuse the Subsystem ID value on products with the same Subsystem Vendor ID in cases where the products are in the same family and generation, and differ only in capacity or performance. Note also that it is acceptable for vendors to use multiple unique Subsystem ID values over time for a single product type, such as to indicate some internal difference such as component selection.

7.5.1.2.4 Expansion ROM Base Address Register (Offset 30h)

Some Functions, especially those that are intended for use on add-in cards, require local EPROMs for expansion ROM (refer to the *PCI Firmware Specification* for a definition of ROM contents). This register is defined to handle the base address and size information for this expansion ROM. Figure 7-13 shows how this register is organized. The register functions exactly like a 32-bit Base Address register except that the encoding (and usage) of the bottom bits is different. The upper 21 bits correspond to the upper 21 bits of the Expansion ROM base address. The number of bits (out of these 21) that a Function actually implements depends on how much address space the Function requires. For instance, a Function that requires a 64 KB area to map its expansion ROM would implement the upper 16 bits in the register, leaving the lower 5 bits (out of the field's 21 bits) hardwired to 0b. Functions that support an expansion ROM must implement this register.

Device independent configuration software can determine how much address space the Function requires by writing a value of all 1's to the address portion of the register and then reading the value back. The Function will return 0's in all don't-care bits, effectively specifying the size and alignment requirements. The amount of address space a Function requests must not be greater than 16 MB.

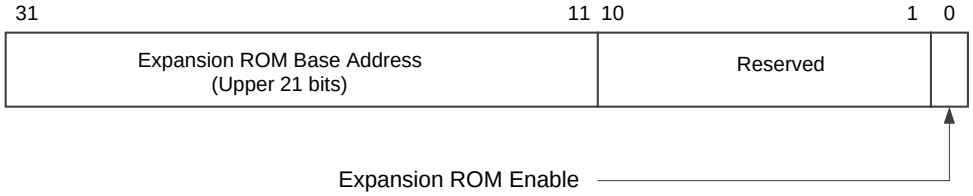


Figure 7-13: Expansion ROM Base Address Register Layout

Bit 0 in the register is used to control whether or not the Function accepts accesses to its expansion ROM. When this bit is 0b, the Function's expansion ROM address space is disabled. When the bit is 1b, address decoding is enabled using the parameters in the other part of the Expansion ROM Base Address register. This allows a Function to be used with or without an expansion ROM depending on system configuration. The Memory Space Enable bit in the Command register has precedence over the Expansion ROM Enable bit. A Function must claim accesses to its expansion

ROM only if both the Memory Space Enable bit and the Expansion ROM Enable bit are Set. The default value of this bit is 0b.

In order to minimize the number of address decoders needed, a Function may share a decoder between the Expansion ROM Base Address register and other Base Address registers or entry in the Enhanced Allocation capability[124]. When expansion ROM decode is enabled, the decoder is used for accesses to the expansion ROM and device independent software must not access the Function through any other Base Address registers or entry in the Enhanced Allocation capability.

7.5.1.2.5 Min_Gnt/Max_Lat Registers (Offset 3Eh/3Fh)

These registers do not apply to PCI Express. They must be read-only and hardwired to 00h.

[124] Note that it is the address decoder that is shared, not the registers themselves. The Expansion ROM Base Address register and other Base Address registers or entries in the Enhanced Allocation capability must be able to hold unique values at the same time.

7.5.1.3 *Type 1 Configuration Space Header*

Figure 7-14 details allocation for register fields of Type 1 Configuration Space header for Switch and Root Ports.

31			0	Byte Offset
Device ID		Vendor ID		00h
Status		Command		04h
Class Code			Revision ID	08h
BIST	Header Type	Primary Latency Timer	Cache Line Size	0Ch
Base Address Register 0				10h
Base Address Register 1				14h
Secondary Latency Timer	Subordinate Bus Number	Secondary Bus Number	Primary Bus Number	18h
Secondary Status		I/O Limit	I/O Base	1Ch
Memory Limit		Memory Base		20h
Prefetchable Memory Limit		Prefetchable Memory Base		24h
Prefetchable Base Upper 32 Bits				28h
Prefetchable Limit Upper 32 Bits				2Ch
I/O Limit Upper 16 Bits		I/O Base Upper 16 Bits		30h
Reserved			Capability Pointer	34h
Expansion ROM Base Address				38h
Bridge Control		Interrupt Pin	Interrupt Line	3Ch

OM14317

Figure 7-14: Type 1 Configuration Space Header

Section 7.5.1.1 details the PCI Express-specific registers that are valid for all Configuration Space header types. The PCI Express-specific interpretation of registers specific to Type 1 Configuration Space header is defined in this section. Register interpretations described in this section apply to PCI-PCI Bridge structures representing Switch and Root Ports; other device Functions such as PCI Express to PCI/PCI-X Bridges with Type 1 Configuration Space headers are not covered by this section.

7.5.1.3.1 Base Address Registers (Offset 10h-14h)

These registers are defined in Section 7.5.2.1. However the number of BARs available within the Type 1 Configuration Space header is different than that of the Type 0 Configuration Space header.

7.5.1.3.2 Primary Bus Number (Offset 18h)

Except as noted, this register is not used by PCI Express Functions but must be implemented as read-write and the default value must be 00h, for compatibility with legacy software. PCI Express Functions capture the Bus (and Device) Number as described (including exceptions) in Section 2.2.6. Refer to *PCI Express to PCI/PCI-X Bridge Specification* for exceptions to this requirement.

7.5.1.3.3 Secondary Bus Number (Offset 19h)

The Secondary Bus Number register is used to record the bus number of the PCI bus segment to which the secondary interface of the bridge is connected. Configuration software programs the value in this register. The Bridge uses this register to determine when and how to respond to an ID-routed TLP observed on its primary interface, notably when to forward the TLP to its secondary interface, in certain cases after performing some conversion. See Section 7.3.3 for Configuration Request routing and conversion rules.

7.5.1.3.4 Subordinate Bus Number (Offset 1Ah)

The Subordinate Bus Number register is used to record the bus number of the highest numbered PCI bus segment which is behind (or subordinate to) the bridge. Configuration software programs the value in this register. The Bridge uses this register to determine when and how to respond to an ID-routed TLP observed on its primary interface, notably when to forward the TLP to its secondary interface. See Section 7.3.3 for Configuration Request routing rules. This register must be implemented as read-write and the default value must be 00h.

7.5.1.3.5 Secondary Latency Timer (Offset 1Bh)

This register does not apply to PCI Express. It must be read-only and hardwired to 00h. For PCI Express to PCI/PCI-X Bridges, refer to the *PCI Express to PCI/PCI-X Bridge Specification* for requirements for this register.

7.5.1.3.6 I/O Base/Limit (Offset 1Ch/1Dh)

The I/O Base and I/O Limit registers are optional and define an address range that is used by the bridge to determine when to forward I/O transactions from one interface to the other.

If a bridge does not implement an I/O address range, then both the I/O Base and I/O Limit registers must be implemented as read-only registers that return zero when read. If a bridge supports an I/O address range, then these registers must be initialized by configuration software so default states are not specified.

If a bridge implements an I/O address range, the upper 4 bits of both the I/O Base and I/O Limit registers are writable and correspond to address bits Address[15:12]. For the purpose of address decoding, the bridge assumes that the lower 12 address bits, Address[11:0], of the I/O base address (not implemented in the I/O Base register) are zero. Similarly, the bridge assumes that the lower 12 address bits, Address[11:0], of the I/O limit address (not implemented in the I/O Limit register) are

FFFh. Thus, the bottom of the defined I/O address range will be aligned to a 4-KB boundary and the top of the defined I/O address range will be one less than a 4-KB boundary.

The I/O Limit register can be programmed to a smaller value than the I/O Base register, if there are no I/O addresses on the secondary side of the bridge. In this case, the bridge will not forward any I/O transactions from the primary bus to the secondary and will forward all I/O transactions from the secondary bus to the primary bus.

The lower four bits of both the I/O Base and I/O Limit registers are read-only, contain the same value, and encode the I/O addressing capability of the bridge according to Table 7-9.

Table 7-9: I/O Addressing Capability

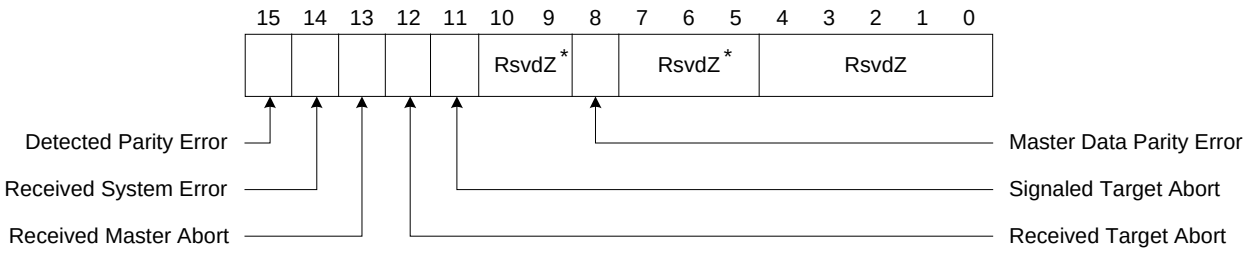
Bits 3:0	I/O Addressing Capability
0h	16-bit I/O addressing
1h	32-bit I/O addressing
2h-Fh	Reserved

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

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

7.5.1.3.7 Secondary Status Register (Offset 1Eh)

Table 7-10 defines the Secondary Status register and Figure 7-15 provides the register layout. For PCI Express to PCI/PCI-X Bridges, refer to the *PCI Express to PCI/PCI-X Bridge Specification* for requirements for this register.



* These bits were used in previous versions of the programming model. Careful consideration should be given to any attempt to repurpose them.

Figure 7-15: Secondary Status Register

Table 7-10: Secondary Status Register

Bit Location	Register Description	Attributes
5	**66 MHz Capable** – This bit was originally described in the *PCI-to-PCI Bridge Architecture Specification*. Its functionality oes not apply to PCI Express and the bit must be hardwired to 0b.	RO
7	**Fast Back-to-Back Transactions Capable** – This bit was originally described in the *PCI-to-PCI Bridge Architecture Specification*. Its functionality does not apply to PCI Express and the bit must be hardwired to 0b.	RO
8	**Master Data Parity Error** – See Section 7.5.1.1.14. This bit is Set by a Function with a Type 1 Configuration Space header if the Parity Error Response Enable bit in the Bridge Control register is Set and either of the following two conditions occurs: Port receives a Poisoned Completion coming Upstream Port transmits a Poisoned Request Downstream If the Parity Error Response Enable bit is Clear, this bit is never Set. Default value of this bit is 0b.	RW1C
10:9	**DEVSEL Timing** – This field was originally described in the *PCI-to-PCI Bridge Architecture Specification*. Its functionality does not apply to PCI Express and the field must be hardwired to 00b.	RO
11	**Signaled Target Abort** – See Section 7.5.1.1.14. This bit is Set when the Secondary Side for Type 1 Configuration Space header Function (for Requests completed by the Type 1 header Function itself) completes a Posted or Non-Posted Request as a Completer Abort error. Default value of this bit is 0b.	RW1C

Bit Location	Register Description	Attributes
12	**Received Target Abort** – See Section 7.5.1.1.14. This bit is Set when the Secondary Side for Type 1 Configuration Space header Function (for Requests initiated by the Type 1 header Function itself) receives a Completion with Completer Abort Completion Status. Default value of this bit is 0b.	RW1C
13	**Received Master Abort** – See Section 7.5.1.1.14. This bit is Set when the Secondary Side for Type 1 Configuration Space header Function (for Requests initiated by the Type 1 header Function itself) receives a Completion with Unsupported Request Completion Status. Default value of this bit is 0b.	RW1C
14	**Received System Error** – See Section 7.5.1.1.14. This bit is Set when the Secondary Side for a Type 1 Configuration Space header Function receives an ERR_FATAL or ERR_NONFATAL Message. Default value of this bit is 0b.	RW1C
15	**Detected Parity Error** – See Section 7.5.1.1.14. This bit is Set by a Function with a Type 1 Configuration Space header when a Poisoned TLP is received by its Secondary Side, regardless of the state the Parity Error Response Enable bit in the Bridge Control register. Default value of this bit is 0b.	RW1C

7.5.1.3.8 Memory Base/Limit (Offset 20h/22h)

The Memory Base and Memory Limit registers define a memory mapped address range which is used by the bridge to determine when to forward memory transactions from one interface to the other (see the *PCI-to-PCI Bridge Architecture Specification* for additional details).

The upper 12 bits of both the Memory Base and Memory Limit registers are read/write and correspond to the upper 12 address bits, Address[31:20], of 32-bit addresses. For the purpose of address decoding, the bridge assumes that the lower 20 address bits, Address[19:0], of the memory base address (not implemented in the Memory Base register) are zero. Similarly, the bridge assumes that the lower 20 address bits, Address[19:0], of the memory limit address (not implemented in the Memory Limit register) are F FFFFh. Thus, the bottom of the defined memory address range will be aligned to a 1 MB boundary and the top of the defined memory address range will be one less than a 1 MB boundary.

The Memory Limit register must be programmed to a smaller value than the Memory Base register if there is no memory-mapped address space on the secondary side of the bridge.

If there is no prefetchable memory space, and there is no memory-mapped space on the secondary side of the bridge, then the bridge will not forward any memory transactions from the primary bus to the secondary bus and will forward all memory transactions from the secondary bus to the primary bus.

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

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

7.5.1.3.9 Prefetchable Memory Base/Limit (Offset 24h/26h)

The Prefetchable Memory Base and Prefetchable Memory Limit registers must indicate that 64-bit addresses are supported.

The Prefetchable Memory Base and Prefetchable Memory Limit registers are optional. They define a prefetchable memory address range which is used by the bridge to determine when to forward memory transactions from one interface to the other.

If a bridge does not implement a prefetchable memory address range, then both Prefetchable Memory Base and Prefetchable Memory Limit registers must be implemented as read-only registers which return zero when read. If a bridge implements a Prefetchable memory address range, then both of these registers must be implemented as read/write registers. If a bridge supports a prefetchable memory address range, then these registers must be initialized by configuration software so default states are not specified.

If the bridge implements a prefetchable memory address range, the upper 12 bits of the register are read/write and correspond to the upper 12 address bits, Address[31:20], of 32-bit addresses. For the purpose of address decoding, the bridge assumes that the lower 20 address bits, Address[19:0], of the prefetchable memory base address (not implemented in the Prefetchable Memory Base register) are zero. Similarly, the bridge assumes that the lower 20 address bits, Address[19:0], of the prefetchable memory limit address (not implemented in the Prefetchable Memory Limit register) are F FFFFh. Thus, the bottom of the defined prefetchable memory address range will be aligned to a 1 MB boundary and the top of the defined memory address range will be one less than a 1 MB boundary.

The Prefetchable Memory Limit register must be programmed to a smaller value than the Prefetchable Memory Base register if there is no prefetchable memory on the secondary side of the bridge. If there is no prefetchable memory, and there is no memory-mapped address space (see the *PCI-to-PCI Bridge Architecture Specification*) on the secondary side of the bridge, then the bridge will not forward any memory transactions from the primary bus to the secondary but will forward all memory transactions from the secondary bus to the primary bus.

The bottom 4 bits of both the Prefetchable Memory Base and Prefetchable Memory Limit registers are read-only, contain the same value, and encode whether or not the bridge supports 64-bit addresses. If these four bits have the value 0h, then the bridge supports only 32-bit addresses. If these four bits have the value 01h, then the bridge supports 64-bit addresses and the Prefetchable Base Upper 32 Bits and Prefetchable Limit Upper 32 Bits registers hold the rest of the 64-bit prefetchable base and limit addresses respectively. All other encodings are Reserved.

7.5.1.3.10 Prefetchable Base/Limit Upper 32 Bits (Offset 28h/2Ch)

The Prefetchable Base Upper 32 Bits and Prefetchable Limit Upper 32 Bits registers are optional extensions to the Prefetchable Memory Base and Prefetchable Memory Limit registers.

If the Prefetchable Memory Base and Prefetchable Memory Limit registers indicate support for 32-bit addressing, then the Prefetchable Base Upper 32 Bits and Prefetchable Limit Upper 32 Bits registers are both implemented as read-only registers that return zero when read. If the Prefetchable Memory Base and Prefetchable Memory Limit registers indicate support for 64-bit addressing, then the Prefetchable Base Upper 32 Bits and Prefetchable Limit Upper 32 Bits registers are implemented as read/write registers. If these registers are implemented as read/write registers, they must be initialized by configuration software so default states are not specified.

If a 64-bit prefetchable memory address range is supported, the Prefetchable Base Upper 32 Bits and Prefetchable Limit Upper 32 Bits registers specify the upper 32 bits, corresponding to Address[63:32], of the 64-bit base and limit addresses which specify the prefetchable memory address range (see the *PCI-to-PCI Bridge Architecture Specification* for additional details).

7.5.1.3.11 I/O Base/Limit Upper 16 Bits (Offset 30h/32h)

The I/O Base Upper 16 Bits and I/O Limit Upper 16 Bits registers are optional extensions to the I/O Base and I/O Limit registers. If the I/O Base and I/O Limit registers indicate support for 16-bit I/O address decoding, then the I/O Base Upper 16 Bits and I/O Limit Upper 16 Bits registers are implemented as read-only registers which return zero when read.

If the I/O Base and I/O Limit registers indicate support for 32-bit I/O addressing, then the I/O Base Upper 16 Bits and I/O Limit Upper 16 Bits registers must be initialized by configuration software so default states are not specified.

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

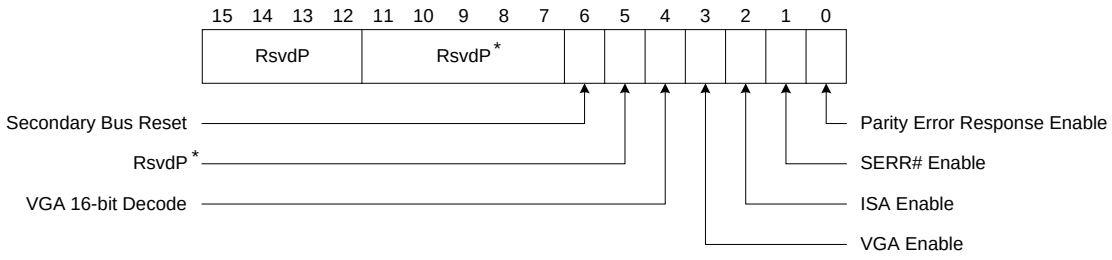
7.5.1.3.12 Expansion ROM Base Address (Offset 38h)

This register is defined in Section 7.5.1.2.4. However the offset of the register within the Type 1 Configuration Space header is different than that of the Type 0 Configuration Space header.

7.5.1.3.13 Bridge Control Register (Offset 3Eh)

The Bridge Control register provides extensions to the Command register that are specific to a Function with a Type 1 Configuration Space header. The Bridge Control register provides many of the same controls for the secondary interface that are provided by the Command register for the primary interface. There are some bits that affect the operation of both interfaces of the bridge.

Table 7-11 defines the Bridge Control register and Figure 7-16 depicts register layout. For PCI Express to PCI/PCI-X Bridges, refer to the *PCI Express to PCI/PCI-X Bridge Specification* for requirements for this register.



* These bits were used in previous versions of the programming model. Careful consideration should be given to any attempt to repurpose them.

Figure 7-16: Bridge Control Register

Table 7-11: Bridge Control Register

Bit Location	Register Description	Attributes
0	**Parity Error Response Enable** – See Section 7.5.1.1.14. This bit controls the logging of poisoned TLPs in the Master Data Parity Error bit in the Secondary Status register. Default value of this bit is 0b.	RW
1	**SERR# Enable** – See Section 7.5.1.1.14. This bit controls forwarding of ERR_COR, ERR_NONFATAL and ERR_FATAL from secondary to primary. Default value of this bit is 0b.	RW
2	**ISA Enable** – Modifies the response by the bridge to ISA I/O addresses. This applies only to I/O addresses that are enabled by the I/O Base and I/O Limit registers and are in the first 64 KB of I/O address space (0000 0000h to 0000 FFFFh). If this bit is Set, the bridge will block any forwarding from primary to secondary of I/O transactions addressing the last 768 bytes in each 1-KB block. In the opposite direction (secondary to primary), I/O transactions will be forwarded if they address the last 768 bytes in each 1-KB block. 0b - forward downstream all I/O addresses in the address range defined by the I/O Base and I/O Limit registers 1b - forward upstream ISA I/O addresses in the address range defined by the I/O Base and I/O Limit registers that are in the first 64 KB of PCI I/O address space (top 768 bytes of each 1-KB block Default value of this bit is 0b.	RW

Bit Location	Register Description	Attributes
3	VGA Enable – Modifies the response by the bridge to VGA compatible addresses. If the VGA Enable bit is Set, the bridge will positively decode and forward the following accesses on the primary interface to the secondary interface (and, conversely, block the forwarding of these addresses from the secondary to primary interface): • Memory accesses in the range 000A 0000h to 000B FFFFh • I/O addresses in the first 64 KB of the I/O address space (Address[31:16] are 0000h) where Address[9:0] are in the ranges 3B0h to 3BBh and 3C0h to 3DFh (inclusive of ISA address aliases determined by the setting of VGA 16-bit Decode) If the VGA Enable bit is Set, forwarding of these accesses is independent of the I/O address range and memory address ranges defined by the I/O Base and Limit registers, the Memory Base and Limit registers, and the Prefetchable Memory Base and Limit registers of the bridge. (Forwarding of these accesses is also independent of the setting of the ISA Enable bit (in the Bridge Control register) when the VGA Enable bit is Set. Forwarding of these accesses is qualified by the I/O Space Enable and Memory Space Enable bits in the Command register.) 0b - do not forward VGA compatible memory and I/O addresses from the primary to the secondary interface (addresses defined above) unless they are enabled for forwarding by the defined I/O and memory address ranges 1b - forward VGA compatible memory and I/O addresses (addresses defined above) from the primary interface to the secondary interface (if the I/O Space Enable and Memory Space Enable bits are set) independent of the I/O and memory address ranges and independent of the ISA Enable bit Functions that do not support VGA are permitted to hardwire this bit to 0b. Default value of this bit is 0b.	RW
4	VGA 16-bit Decode – This bit only has meaning if bit 3 (VGA Enable) of this register is also Set, enabling VGA I/O decoding and forwarding by the bridge. This bit enables system configuration software to select between 10-bit and 16-bit I/O address decoding for all VGA I/O register accesses that are forwarded from primary to secondary. 0b - execute 10-bit address decodes on VGA I/O accesses 1b - execute 16-bit address decodes on VGA I/O accesses Functions that do not support VGA are permitted to hardwire this bit to 0b. Default value of this bit is 0b.	RW
5	**Master Abort Mode** – This bit was originally described in the *PCI-to-PCI Bridge Architecture Specification*. Its functionality does not apply to PCI Express and the bit must be hardwired to 0b.	RO

Bit Location	Register Description	Attributes
6	**Secondary Bus Reset** – Setting this bit triggers a hot reset on the corresponding PCI Express Port. Software must ensure a minimum reset duration (T_{rst}) as defined in the *PCI Local Bus Specification*. Software and systems must honor first-access-following-reset timing requirements defined in Section 6.6., unless the Readiness Notifications mechanism (see Section 6.23) is used or if the Immediate Readiness bit in the relevant Function's Status register is Set. Port configuration registers must not be changed, except as required to update Port status. Default value of this bit is 0b.	RW
7	**Fast Back-to-Back Transactions Enable** – This bit was originally described in the *PCI-to-PCI Bridge Architecture Specification*. Its functionality does not apply to PCI Express and the bit must be hardwired to 0b.	RO
8	**Primary Discard Timer** – This bit was originally described in the *PCI-to-PCI Bridge Architecture Specification*. Its functionality does not apply to PCI Express and the bit must be hardwired to 0b.	RO
9	**Secondary Discard Timer** – This bit was originally described in the *PCI-to-PCI Bridge Architecture Specification*. Its functionality does not apply to PCI Express and the bit must be hardwired to 0b.	RO
10	**Discard Timer Status** – This bit was originally described in the *PCI-to-PCI Bridge Architecture Specification*. Its functionality does not apply to PCI Express and the bit must be hardwired to 0b.	RO
11	**Discard Timer SERR# Enable** – This bit was originally described in the *PCI-to-PCI Bridge Architecture Specification*. Its functionality does not apply to PCI Express and must be hardwired to 0b.	RO

7.5.2 PCI Power Management Capability Structure

This section describes the registers making up the PCI Power Management Interface structure.

Figure 7-17 illustrates the organization of the PCI Power Management Capability structure. This structure is required for all PCI Express Functions.

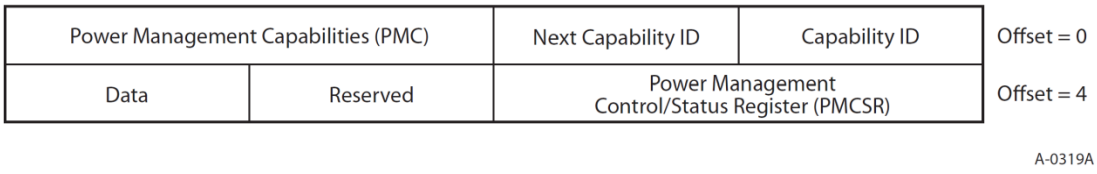


Figure 7-17: Power Management Capability Structure

Note: The 8-bit Data register (Offset 07h) is optional for both Type 0 and Type 1 Functions.

PCI Express device Functions are required to support D0 and D3 device states; PCI-PCI Bridge structures representing PCI Express Ports as described in Section 7.1 are required to indicate PME Message passing capability due to the in-band nature of PME messaging for PCI Express.

The PME_Status bit for the PCI-PCI Bridge structure representing PCI Express Ports, however, is only Set when the PCI-PCI Bridge Function is itself generating a PME. The PME_Status bit is not Set when the Bridge is propagating a PME Message but the PCI-PCI Bridge Function itself is not internally generating a PME.

7.5.2.1 *Power Management Capabilities Register (Offset 00h)*

Figure 7-18 details allocation of register fields for Power Management Capabilities register and Table 7-12 describes the requirements for this register.

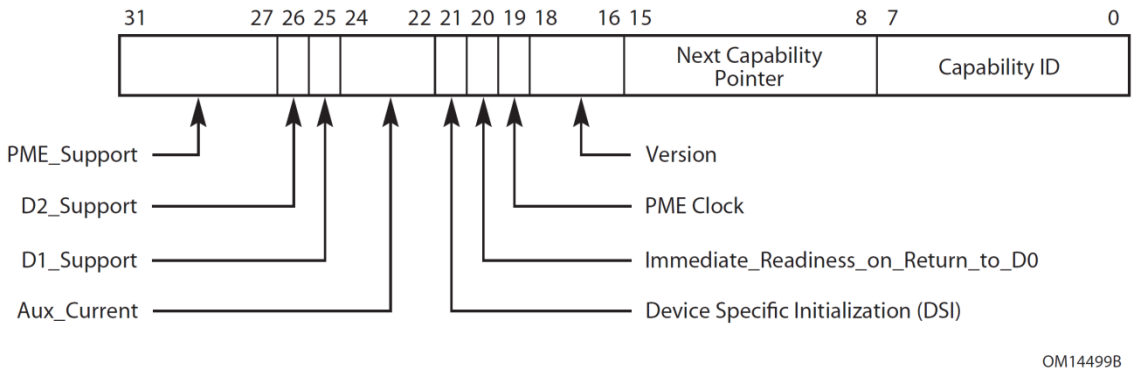


Figure 7-18: Power Management Capabilities Register

Table 7-12: Power Management Capabilities Register

Bit Location	Register Description	Attributes
7:0	**Capability ID** –This field returns 01h to indicate that this is the PCI Power Management Capability. Each Function may have only one item in its capability list with Capability_ID set to 01h.	RO
15:8	**Next Capability Pointer** - This field provides an offset into the Function's Configuration Space pointing to the location of next item in the capabilities list. If there are no additional items in the capabilities list, this field is set to 00h.	RO
18:16	**Version** - Must be hardwired to 011b for Functions compliant to this specification.	RO
19	**PME Clock** – Does not apply to PCI Express and must be hardwired to 0b.	RO

Bit Location	Register Description	Attributes
20	**Immediate_Readiness_on_Return_to_D0** - If this bit is a “1”, this Function is guaranteed to be ready to successfully complete valid accesses immediately after being set to D0. These accesses include Configuration cycles, and if the Function returns to D0initialized, they also include Memory and I/O Cycles. When this bit is “1”, for accesses to this Function, software is exempt from all requirements to delay accesses following a transition to D0, including but not limited to the 10 ms delay ; the delays described in Section 5.9. How this guarantee is established is beyond the scope of this document. It is permitted that system software/firmware provide mechanisms that supersede the indication provided by this bit, however such software/firmware mechanisms are outside the scope of this specification.	RO
21	**Device Specific Initialization** - The DSI bit indicates whether special initialization of this Function is required. When Set indicates that the Function requires a device specific initialization sequence following a transition to the $D0_{uninitialized}$ state. Refer to Section 5.16.3.	RO
24:22	**Aux_Current** - This 3 bit field reports the Vaux auxiliary current requirements for the Function. If this Function implements the Data Register, this field must be hardwired to 000b. If PME_Support is 0 xxxxb (PME assertion from $D3_{cold}$ is not supported), this field must be hardwired to 0000b. For Functions where PME_Support is 1 xxxxb (PME assertion from $D3_{cold}$ is supported), and which do not implement the Data register, the following encodings apply : Encoding Vaux Max. Current Required 111b 375 mA 110b 320 mA 101b 270 mA 100b 220 mA 011b 160 mA 010b 100 mA 001b 55 mA 000b 0 (self powered)	RO
25	**D1_Support** - If this bit is Set, this Function supports the D1 Power Management State. Functions that do not support D1 must always return a value of 0b for this bit.	RO
26	**D2_Support** - If this bit is Set, this Function supports the D2 Power Management State. Functions that do not support D2 must always return a value of 0b for this bit.	RO

Bit Location	Register Description	Attributes
31:27	**PME_Support** - This 5-bit field indicates the power states in which the Function may generate a PME and/or forward PME Messages. A value of 0b for any bit indicates that the Function is not capable of asserting PME while in that power state. bit(27) X XXX1b - PME can be generated from D0 bit(28) X XX1Xb - PME can be generated from D1 bit(29) X X1XXb - PME can be generated from D2 bit(30) X 1XXXb - PME can be generated from $D3_{hot}$ bit(31) 1 XXXXb - PME can be generated from $D3_{cold}$ Bit 31 (PME can be asserted from $D3_{cold}$) represents a special case. Functions that Set this bit require some sort of auxiliary power source. Implementation specific mechanisms are recommended to validate that the power source is available before setting this bit. Each bit that corresponds to a supported D-state must be Set for PCI-PCI Bridge structures representing Ports on Root Complexes/Switches to indicate that the Bridge will forward PME Messages. Bit 31 must only be Set if the Port is still able to forward PME Messages when main power is not available.	RO

7.5.2.2 *Power Management Control/Status Register (Offset 04h)*

This register is used to manage the PCI Function's power management state as well as to enable/monitor PMEs.

The PME Context includes the value of the PME_Status and PME_En bits, implementation specific state needed during $D3_{cold}$ to implement the wakeup functionality (e.g., recognized a Wake on LAN packet and generate a PME Message), as well as any additional implementation specific state that must be preserved during a transition to the $D0_{uninitialized}$ state.

If a Function supports PME generation from $D3_{cold}$, its PME Context is not affected by Reset. This is because the Function's PME functionality itself may have been responsible for the wake event which caused the transition back to D0. Therefore, the PME Context must be preserved for the system software to process.

If PME generation is not supported from $D3_{cold}$, then all PME Context is initialized with the assertion of Reset.

Figure 7-19 details allocation of the register fields for the Power Management Control/Status register and Table 7-13 describes the requirements for this register.

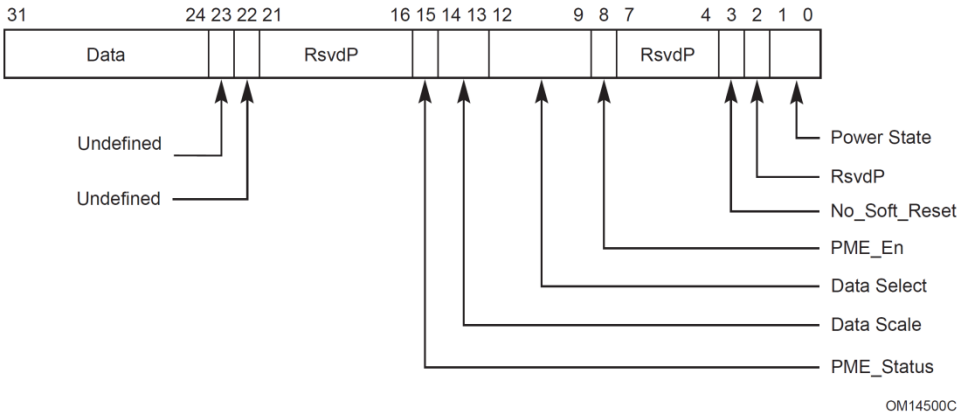


Figure 7-19: Power Management Control/Status Register

Table 7-13: Power Management Control/Status Register Requirements

Bit Location	Register Description	Attributes
1:0	**PowerState** – This 2-bit field is used both to determine the current power state of a Function and to set the Function into a new power state. The definition of the field values is given below. 00b - D0 01b - D1 10b - D2 11b - $D3_{hot}$ If software attempts to write an unsupported, optional state to this field, the write operation must complete normally; however, the data is discarded and no state change occurs. Default value of this field is 00b.	RW
3	**No_Soft_Reset** – This bit indicates the state of the Function after writing the PowerState field to transition the Function from $D3_{hot}$ to D0. When Set, this transition preserves internal Function state. The Function is in $D0_{Active}$ and no additional software intervention is required. When Clear, this transition results in undefined internal Function state. Regardless of this bit, Functions that transition from $D3_{hot}$ to D0 by Fundamental Reset will return to $D0_{Uninitialized}$ with only PME context preserved if PME is supported and enabled.	RO

Bit Location	Register Description	Attributes
8	**PME_En** – When Set, the Function is permitted to generate a PME. When Clear, the Function is not permitted to generate a PME. If PME_Support is 1 xxxxb (PME generation from $D3_{cold}$) or the Function consumes Aux power and Aux power is available this bit is RWS and the bit is not modified by Conventional Reset or FLR If PME_Support is 0 xxxxb, this field is not sticky (RW). If PME_Support is 0 0000b, this bit is permitted to be hardwired to 0b.	RW/RWS
12:9	**Data_Select -** This 4-bit field is used to select which data is to be reported through the Data register and Data_Scale field. If the Data register is not implemented, this field must be hardwired to 0000b. Refer to Section 7.5.2.3 for more details. The default of this field is 0000b	RW
14:13	**Data_Scale -** This field indicates the scaling factor to be used when interpreting the value of the Data register. The value and meaning of this field will vary depending on which data value has been selected by the Data_Select field. This field is a required component of the Data register (offset 7) and must be implemented if the Data register is implemented. If the Data register is not implemented, this field must be hardwired to 00b. Refer to Section 7.5.2.3 for more details.	RO
15	**PME_Status** – This bit is Set when the Function would normally generate a PME signal. The value of this bit is not affected by the value of the PME_En bit. If PME_Support bit 31 of the Power Management Capabilities register is Clear, this bit is permitted to be hardwired to 0b. Functions that consume Aux power must preserve the value of this sticky register when Aux power is available. In such Functions, this register value is not modified by Conventional Reset or FLR.	RW1CS
22	**Undefined – this bit was defined in previous specifications. It should be ignored by software.**	RO
23	Undefined – this bit was defined in previous specifications. It should be ignored by software.	RO

7.5.2.3 *Data (Offset 07h)*

The Data register is an optional, 8-bit read-only register that provides a mechanism for the Function to report state dependent operating power consumed or dissipation.

If the Data register is implemented, then the Data_Select and Data_Scale fields must also be implemented. If this register is not implemented, it must be hardwired to 00h.

Software may check for the presence of the Data register by writing different values into the Data_Select field, looking for non-zero return data in the Data register and/or Data_Scale field. Any non-zero Data register/Data_Select read data indicates that the Data register complex has been implemented.

Table 7-14: Data Register

Bit Location	Register Description	Attributes
7:0	**Data** - This register is used to report the state dependent data requested by the Data_Select field. The value of this register is scaled by the value reported by the Data_Scale field.	RO

The Data register is used by writing the proper value to the Data_Select field in the PMCSR and then reading the Data_Scale field and the Data register. The binary value read from Data is then multiplied by the scaling factor indicated by Data_Scale to arrive at the value for the desired measurement. Table 7-15 shows which measurements are defined and how to interpret the values of each register.

Table 7-15: Power Consumption/Dissipation Reporting

Value in Data_Select	Data Reported	Data_Scale Interpretation	Units/Accuracy
0	D0 Power Consumed	0 = Unknown 1 = 0.1x 2 = 0.01x 3 = 0.001x	Watts
1	D1 Power Consumed		
2	D2 Power Consumed		
3	D3 Power Consumed		
4	D0 Power Dissipated		
5	D1 Power Dissipated		
6	D2 Power Dissipated		
7	D3 Power Dissipated		
8	Common logic power consumption (Multi-Function Devices, Function 0 only) Function 0 of a Multi-Function Device: Power consumption that is not associated with a specific Function. All other Functions: Reserved		
9-15	Reserved	Reserved	TBD

The "Power Consumed" values defined above must include all power consumed from the PCI power planes through the PCI connector pins. If the PCI add-in card provides power to external

devices, that power must be included as well. It must not include any power derived from a battery or an external source. This information is useful for management of the power supply or battery.

The "Power Dissipated" values must provide the amount of heat which will be released into the interior of the computer chassis. This excludes any power delivered to external devices but must include any power derived from a battery or external power source and dissipated inside the computer chassis. This information is useful for fine grained thermal management.

Multi-Function Devices are recommended to report the power consumed by each Function in each corresponding Function's Configuration Space. In a Multi-Function Device, power consumption for circuitry common to multiple Functions is reported in Function 0's Configuration Space through the Data register once the Data_Select field of Function 0's Power Management Control/Status register has been programmed to 1000b. For a Multi-Function Device, power consumption of the device is the sum of this value and, for every Function of the device, the reported value associated with the Function's current Power State.

Multiple component add-in cards implementing power reporting (i.e., multiple components behind a switch or bridge) must have the switch/bridge report the power it uses by itself. Each Function of each component on the add-in card is responsible for reporting the power consumed by that Function.

7.5.3 PCI Express Capability Structure

PCI Express defines a Capability structure in PCI-compatible Configuration Space (first 256 bytes) as shown in Figure 7-3. This structure allows identification of a PCI Express device Function and indicates support for new PCI Express features. The PCI Express Capability structure is required for PCI Express device Functions. The Capability structure is a mechanism for enabling PCI software transparent features requiring support on legacy operating systems. In addition to identifying a PCI Express device Function, the PCI Express Capability structure is used to provide access to PCI Express specific Control/Status registers and related Power Management enhancements.

Figure 7-20 details allocation of register fields in the PCI Express Capability structure.

The PCI Express Capabilities, Device Capabilities, Device Status, and Device Control registers are required for all PCI Express device Functions. Device Capabilities 2, Device Status 2, and Device Control 2 registers are required for all PCI Express device Functions that implement capabilities requiring those registers. For device Functions that do not implement the Device Capabilities 2, Device Status 2, and Device Control 2 registers, these spaces must be hardwired to 0b.

The Link Capabilities, Link Status, and Link Control registers are required for all Root Ports, Switch Ports, Bridges, and Endpoints that are not RCiEPs. For Functions that do not implement the Link Capabilities, Link Status, and Link Control registers, these spaces must be hardwired to 0. Link Capabilities 2, Link Status 2, and Link Control 2 registers are required for all Root Ports, Switch Ports, Bridges, and Endpoints (except for RCiEPs) that implement capabilities requiring those registers. For Functions that do not implement the Link Capabilities 2, Link Status 2, and Link Control 2 registers, these spaces must be hardwired to 0b.

The Slot Capabilities, Slot Status, and Slot Control registers are required in certain Switch Downstream and Root Ports. The Slot Capabilities register is required if the Slot Implemented bit is Set (see Section 7.5.3.2). The Slot Status and Slot Control registers are required if Slot Implemented

is Set or if Data Link Layer Link Active Reporting Capable is Set (see Section 7.5.3.6). Switch Downstream and Root Ports are permitted to implement these registers, even when they are not required. For Functions that do not implement the Slot Capabilities, Slot Status, and Slot Control registers, these spaces must be hardwired to 0b, with the exception of the Presence Detect State bit in the Slot Status register of Downstream Ports, which must be hardwired to 1b (see Section 7.5.3.11). Slot Capabilities 2, Slot Status 2, and Slot Control 2 registers are required for Switch Downstream and Root Ports if the Function implements capabilities requiring those registers. For Functions that do not implement the Slot Capabilities 2, Slot Status 2, and Slot Control 2 registers, these spaces must be hardwired to 0b.

Root Ports and Root Complex Event Collectors must implement the Root Capabilities, Root Status, and Root Control registers. For Functions that do not implement the Root Capabilities, Root Status, and Root Control registers, these spaces must be hardwired to 0b.

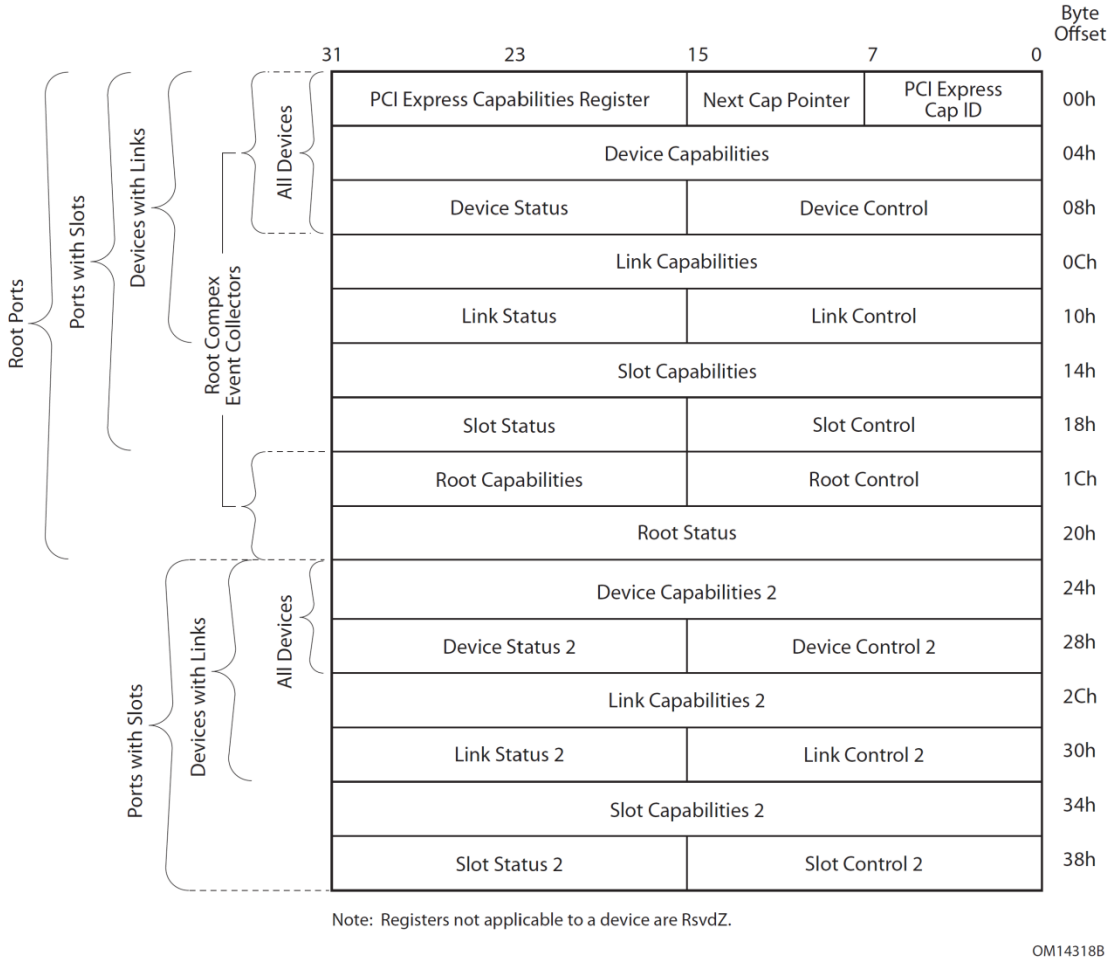


Figure 7-20: PCI Express Capability Structure

7.5.3.1 *PCI Express Capability List Register (Offset 00h)*

The PCI Express Capability List register enumerates the PCI Express Capability structure in the PCI Configuration Space Capability list. Figure 7-21 details allocation of register fields in the PCI Express Capability List register; Table 7-16 provides the respective bit definitions.

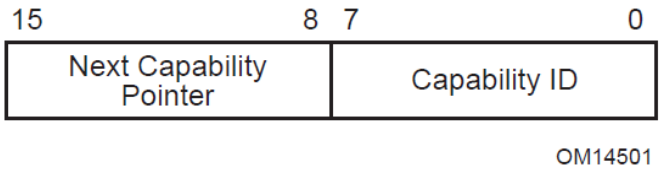


Figure 7-21: PCI Express Capability List Register

Table 7-16: PCI Express Capability List Register

Bit Location	Register Description	Attributes
7:0	**Capability ID** – Indicates the PCI Express Capability structure. This field must return a Capability ID of 10h indicating that this is a PCI Express Capability structure.	RO
15:8	**Next Capability Pointer** – This field contains the offset to the next PCI Capability structure or 00h if no other items exist in the linked list of Capabilities.	RO

7.5.3.2 *PCI Express Capabilities Register (Offset 02h)*

The PCI Express Capabilities register identifies PCI Express device Function type and associated capabilities. Figure 7-22 details allocation of register fields in the PCI Express Capabilities register; Table 7-17 provides the respective bit definitions.

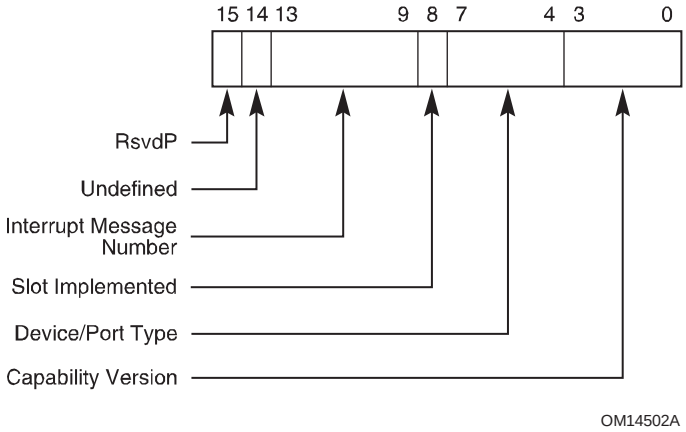


Figure 7-22: PCI Express Capabilities Register

Table 7-17: PCI Express Capabilities Register

Bit Location	Register Description	Attributes
3:0	**Capability Version** – Indicates PCI-SIG defined PCI Express Capability structure version number. A version of the specification that changes the PCI Express Capability structure in a way that is not otherwise identifiable (e.g., through a new Capability field) is permitted to increment this field. All such changes to the PCI Express Capability structure must be software-compatible. Software must check for Capability Version numbers that are greater than or equal to the highest number defined when the software is written, as Functions reporting any such Capability Version numbers will contain a PCI Express Capability structure that is compatible with that piece of software. Must be hardwired to 2h for Functions compliant to this specification.	RO
7:4	**Device/Port Type** – Indicates the specific type of this PCI Express Function. Note that different Functions in a Multi-Function Device can generally be of different types. Defined encodings for Functions that implement a Type 00h PCI Configuration Space header are: 0000b PCI Express Endpoint 0001b Legacy PCI Express Endpoint 1001b RCiEP 1010b Root Complex Event Collector Defined encodings for Functions that implement a Type 01h PCI Configuration Space header are: 0100b Root Port of PCI Express Root Complex 0101b Upstream Port of PCI Express Switch 0110b Downstream Port of PCI Express Switch 0111b PCI Express to PCI/PCI-X Bridge 1000b PCI/PCI-X to PCI Express Bridge All other encodings are Reserved. Note that the different Endpoint types have notably different requirements in Section 1.3.2 regarding I/O resources, Extended Configuration Space, and other capabilities.	RO

Bit Location	Register Description	Attributes
8	**Slot Implemented** – When Set, this bit indicates that the Link associated with this Port is connected to a slot (as compared to being connected to a system-integrated device or being disabled). This bit is valid for Downstream Ports. This bit is undefined for Upstream Ports.	HwInit
13:9	**Interrupt Message Number** – This field indicates which MSI/MSI-X vector is used for the interrupt message generated in association with any of the status bits of this Capability structure. For MSI, the value in this field indicates the offset between the base Message Data and the interrupt message that is generated. Hardware is required to update this field so that it is correct if the number of MSI Messages assigned to the Function changes when software writes to the Multiple Message Enable field in the MSI Message Control register. For MSI-X, the value in this field indicates which MSI-X Table entry is used to generate the interrupt message. The entry must be one of the first 32 entries even if the Function implements more than 32 entries. For a given MSI-X implementation, the entry must remain constant. If both MSI and MSI-X are implemented, they are permitted to use different vectors, though software is permitted to enable only one mechanism at a time. If MSI-X is enabled, the value in this field must indicate the vector for MSI-X. If MSI is enabled or neither is enabled, the value in this field must indicate the vector for MSI. If software enables both MSI and MSI-X at the same time, the value in this field is undefined.	RO
14	The value read from this bit is undefined. In previous versions of this specification, this bit was used to indicate support for TCS Routing. System software should ignore the value read from this bit. System software is permitted to write any value to this bit.	RO

7.5.3.3 *Device Capabilities Register (Offset 04h)*

The Device Capabilities register identifies PCI Express device Function specific capabilities. Figure 7-23 details allocation of register fields in the Device Capabilities register; Table 7-18 provides the respective bit definitions.

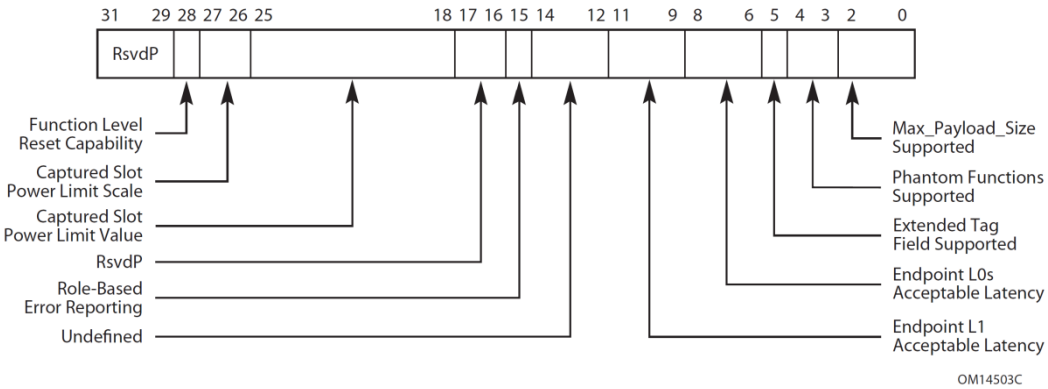


Figure 7-23: Device Capabilities Register

Table 7-18: Device Capabilities Register

Bit Location	Register Description	Attributes
2:0	**Max_Payload_Size Supported** – This field indicates the maximum payload size that the Function can support for TLPs. Defined encodings are: 000b 128 bytes max payload size 001b 256 bytes max payload size 010b 512 bytes max payload size 011b 1024 bytes max payload size 100b 2048 bytes max payload size 101b 4096 bytes max payload size 110b Reserved 111b Reserved The Functions of a Multi-Function Device are permitted to report different values for this field.	RO

Bit Location	Register Description	Attributes
4:3	**Phantom Functions Supported** – This field indicates the support for use of unclaimed Function Numbers to extend the number of outstanding transactions allowed by logically combining unclaimed Function Numbers (called Phantom Functions) with the Tag identifier (see Section 2.2.6.2 for a description of Tag Extensions). With every Function in an ARI Device, the Phantom Functions Supported field must be set to 00b. The remainder of this field description applies only to non-ARI Multi-Function Devices. This field indicates the number of most significant bits of the Function Number portion of Requester ID that are logically combined with the Tag identifier. Defined encodings are: 00b No Function Number bits are used for Phantom Functions. Multi-Function Devices are permitted to implement up to 8 independent Functions. 01b The most significant bit of the Function number in Requester ID is used for Phantom Functions; a Multi-Function Device is permitted to implement Functions 0-3. Functions 0, 1, 2, and 3 are permitted to use Function Numbers 4, 5, 6, and 7 respectively as Phantom Functions. 10b The two most significant bits of Function Number in Requester ID are used for Phantom Functions; a Multi-Function Device is permitted to implement Functions 0-1. Function 0 is permitted to use Function Numbers 2, 4, and 6 for Phantom Functions. Function 1 is permitted to use Function Numbers 3, 5, and 7 as Phantom Functions. 11b All 3 bits of Function Number in Requester ID used for Phantom Functions. The device must have a single Function 0 that is permitted to use all other Function Numbers as Phantom Functions. Note that Phantom Function support for the Function must be enabled by the Phantom Functions Enable field in the Device Control register before the Function is permitted to use the Function Number field in the Requester ID for Phantom Functions.	RO
5	**Extended Tag Field Supported** – This bit, in combination with the 10-Bit Tag Requester Supported bit in the Device Capabilities 2 register, indicates the maximum supported size of the Tag field as a Requester. This bit must be Set if the 10-Bit Tag Requester Supported bit is Set. Defined encodings are: 0b 5-bit Tag field supported 1b 8-bit Tag field supported Note that 8-bit Tag field generation must be enabled by the Extended Tag Field Enable bit in the Device Control register of the Requester Function before 8-bit Tags can be generated by the Requester. See Section 2.2.6.2 for interactions with enabling the use of 10-Bit Tags.	RO

Bit Location	Register Description	Attributes
8:6	**Endpoint L0s Acceptable Latency** – This field indicates the acceptable total latency that an Endpoint can withstand due to the transition from L0s state to the L0 state. It is essentially an indirect measure of the Endpoint's internal buffering. Power management software uses the reported L0s Acceptable Latency number to compare against the L0s exit latencies reported by all components comprising the data path from this Endpoint to the Root Complex Root Port to determine whether ASPM L0s entry can be used with no loss of performance. Defined encodings are: 000b Maximum of 64 ns 001b Maximum of 128 ns 010b Maximum of 256 ns 011b Maximum of 512 ns 100b Maximum of 1 µs 101b Maximum of 2 µs 110b Maximum of 4 µs 111b No limit For Functions other than Endpoints, this field is Reserved and must be hardwired to 000b.	RO
11:9	**Endpoint L1 Acceptable Latency** – This field indicates the acceptable latency that an Endpoint can withstand due to the transition from L1 state to the L0 state. It is essentially an indirect measure of the Endpoint's internal buffering. Power management software uses the reported L1 Acceptable Latency number to compare against the L1 Exit Latencies reported (see below) by all components comprising the data path from this Endpoint to the Root Complex Root Port to determine whether ASPM L1 entry can be used with no loss of performance. Defined encodings are: 000b Maximum of 1 µs 001b Maximum of 2 µs 010b Maximum of 4 µs 011b Maximum of 8 µs 100b Maximum of 16 µs 101b Maximum of 32 µs 110b Maximum of 64 µs 111b No limit For Functions other than Endpoints, this field is Reserved and must be hardwired to 000b.	RO
12	The value read from this bit is undefined. In previous versions of this specification, this bit was used to indicate that an Attention Button is implemented on the adapter and electrically controlled by the component on the adapter. System software must ignore the value read from this bit. System software is permitted to write any value to this bit.	RO

Bit Location	Register Description	Attributes
13	The value read from this bit is undefined. In previous versions of this specification, this bit was used to indicate that an Attention Indicator is implemented on the adapter and electrically controlled by the component on the adapter. System software must ignore the value read from this bit. System software is permitted to write any value to this bit.	RO
14	The value read from this bit is undefined. In previous versions of this specification, this bit was used to indicate that a Power Indicator is implemented on the adapter and electrically controlled by the component on the adapter. System software must ignore the value read from this bit. System software is permitted to write any value to this bit.	RO
15	**Role-Based Error Reporting** – When Set, this bit indicates that the Function implements the functionality originally defined in the Error Reporting ECN for *PCI Express Base Specification, Revision 1.0a*, and later incorporated into *PCI Express Base Specification, Revision 1.1*. This bit must be Set by all Functions conforming to the ECN, *PCI Express Base Specification, Revision 1.1*., or subsequent *PCI Express Base Specification* revisions.	RO
25:18	**Captured Slot Power Limit Value** (Upstream Ports only) – In combination with the Captured Slot Power Limit Scale value, specifies the upper limit on power available to the adapter. Power limit (in Watts) is calculated by multiplying the value in this field by the value in the Captured Slot Power Limit Scale field except when the Captured Slot Power Limit Scale field equals 00b (1.0x) and the Captured Slot Power Limit Value exceeds EFh, then alternative encodings are used (see Section 7.5.3.9). This value is set by the Set_Slot_Power_Limit Message or hardwired to 00h (see Section 6.9). The default value is 00h.	RO
27:26	**Captured Slot Power Limit Scale** (Upstream Ports only) – Specifies the scale used for the Slot Power Limit Value. Range of Values: 00b = 1.0x 01b = 0.1x 10b = 0.01x 11b = 0.001x This value is set by the Set_Slot_Power_Limit Message or hardwired to 00b (see Section 6.9). The default value is 00b.	RO
28	**Function Level Reset Capability** – A value of 1b indicates the Function supports the optional Function Level Reset mechanism described in Section 6.6.2. This bit applies to Endpoints only. For all other Function types this bit must be hardwired to 0b.	RO

7.5.3.4 *Device Control Register (Offset 08h)*

The Device Control register controls PCI Express device specific parameters. Figure 7-24 details allocation of register fields in the Device Control register; Table 7-19 provides the respective bit definitions.

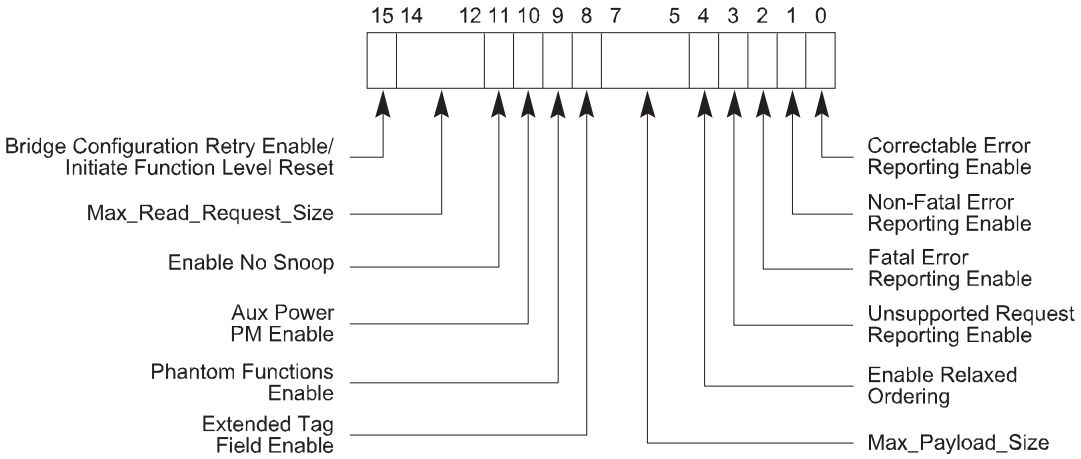


Figure 7-24: Device Control Register

Table 7-19: Device Control Register

Bit Location	Register Description	Attributes
0	**Correctable Error Reporting Enable** – This bit, in conjunction with other bits, controls sending ERR_COR Messages (see Section 6.2.5, Section 6.2.6, and 6.2.10.2 for details). For a Multi-Function Device, this bit controls error reporting for each Function from point-of-view of the respective Function. For a Root Port, the reporting of correctable errors is internal to the root. No external ERR_COR Message is generated. An RCiEP that is not associated with a Root Complex Event Collector is permitted to hardwire this bit to 0b. Default value of this bit is 0b.	RW
1	**Non-Fatal Error Reporting Enable** – This bit, in conjunction with other bits, controls sending ERR_NONFATAL Messages (see Section 6.2.5 and Section 6.2.6 for details). For a Multi-Function Device, this bit controls error reporting for each Function from point-of-view of the respective Function. For a Root Port, the reporting of Non-fatal errors is internal to the root. No external ERR_NONFATAL Message is generated. An RCiEP that is not associated with a Root Complex Event Collector is permitted to hardwire this bit to 0b. Default value of this bit is 0b.	RW

Bit Location	Register Description	Attributes
2	**Fatal Error Reporting Enable** – This bit, in conjunction with other bits, controls sending ERR_FATAL Messages (see Section 6.2.5 and Section 6.2.6 for details). For a Multi-Function Device, this bit controls error reporting for each Function from point-of-view of the respective Function. For a Root Port, the reporting of Fatal errors is internal to the root. No external ERR_FATAL Message is generated. An RCiEP that is not associated with a Root Complex Event Collector is permitted to hardwire this bit to 0b. Default value of this bit is 0b.	RW
3	**Unsupported Request Reporting Enable** – This bit, in conjunction with other bits, controls the signaling of Unsupported Request Errors by sending error Messages (see Section 6.2.5 and Section 6.2.6 for details). For a Multi-Function Device, this bit controls error reporting for each Function from point-of-view of the respective Function. An RCiEP that is not associated with a Root Complex Event Collector is permitted to hardwire this bit to 0b. Default value of this bit is 0b.	RW
4	**Enable Relaxed Ordering** – If this bit is Set, the Function is permitted to set the Relaxed Ordering bit in the Attributes field of transactions it initiates that do not require strong write ordering (see Section 2.2.6.4 and Section 2.4). A Function is permitted to hardwire this bit to 0b if it never sets the Relaxed Ordering attribute in transactions it initiates as a Requester. Default value of this bit is 1b.	RW

Bit Location	Register Description	Attributes
7:5	**Max_Payload_Size** – This field sets maximum TLP payload size for the Function. As a Receiver, the Function must handle TLPs as large as the set value. As a Transmitter, the Function must not generate TLPs exceeding the set value. Permissible values that can be programmed are indicated by the Max_Payload_Size Supported field in the Device Capabilities register (see Section 7.5.3.3). Defined encodings for this field are: 000b 128 bytes max payload size 001b 256 bytes max payload size 010b 512 bytes max payload size 011b 1024 bytes max payload size 100b 2048 bytes max payload size 101b 4096 bytes max payload size 110b Reserved 111b Reserved Functions that support only the 128-byte max payload size are permitted to hardwire this field to 000b. System software is not required to program the same value for this field for all the Functions of a Multi-Function Device. Refer to Section 2.2.2 for important guidance. For ARI Devices, Max_Payload_Size is determined solely by the setting in Function 0. The settings in the other Functions always return whatever value software programmed for each, but otherwise are ignored by the component. Default value of this field is 000b.	RW
8	**Extended Tag Field Enable** – This bit, in combination with the 10-Bit Tag Requester Enable bit in the Device Control 2 register, determines how many Tag field bits a Requester is permitted to use. The following applies when the 10-Bit Tag Requester Enable bit is Clear. If the Extended Tag Field Enable bit is Set, the Function is permitted to use an 8-bit Tag field as a Requester. If the bit is Clear, the Function is restricted to a 5-bit Tag field. See Section 2.2.6.2 for required behavior when the 10-Bit Tag Requester Enable bit is Set. If software changes the value of the Extended Tag Field Enable bit while the Function has outstanding Non-Posted Requests, the result is undefined. Functions that do not implement this capability hardwire this bit to 0b. Default value of this bit is implementation specific.	RW

Bit Location	Register Description	Attributes
9	**Phantom Functions Enable** – This bit, in combination with the 10-Bit Tag Requester Enable bit in the Device Control 2 register, determines how many Tag field bits a Requester is permitted to use. When the 10-Bit Tag Requester Enable bit is Clear, the following paragraph applies. See Section 2.2.6.2 for complete details. When Set, this bit enables a Function to use unclaimed Functions as Phantom Functions to extend the number of outstanding transaction identifiers. If the bit is Clear, the Function is not allowed to use Phantom Functions. Software should not change the value of this bit while the Function has outstanding Non-Posted Requests; otherwise, the result is undefined. Functions that do not implement this capability hardwire this bit to 0b. Default value of this bit is 0b.	RW
10	**Aux Power PM Enable** – When Set this bit, enables a Function to draw Aux power independent of PME Aux power. Functions that require Aux power on legacy operating systems should continue to indicate PME Aux power requirements. Aux power is allocated as requested in the Aux_Current field of the Power Management Capabilities register (PMC), independent of the PME_En bit in the Power Management Control/Status register (PMCSR) (see Chapter 5). For Multi-Function Devices, a component is allowed to draw Aux power if at least one of the Functions has this bit set. Note: Functions that consume Aux power must preserve the value of this sticky register when Aux power is available. In such Functions, this bit is not modified by Conventional Reset. Functions that do not implement this capability hardwire this bit to 0b.	RWS
11	**Enable No Snoop** – If this bit is Set, the Function is permitted to Set the No Snoop bit in the Requester Attributes of transactions it initiates that do not require hardware enforced cache coherency (see Section 2.2.6.5). Note that setting this bit to 1b should not cause a Function to Set the No Snoop attribute on all transactions that it initiates. Even when this bit is Set, a Function is only permitted to Set the No Snoop attribute on a transaction when it can guarantee that the address of the transaction is not stored in any cache in the system. This bit is permitted to be hardwired to 0b if a Function would never Set the No Snoop attribute in transactions it initiates. Default value of this bit is 1b.	RW

Bit Location	Register Description	Attributes
14:12	**Max_Read_Request_Size** – This field sets the maximum Read Request size for the Function as a Requester. The Function must not generate Read Requests with a size exceeding the set value. Defined encodings for this field are: 000b 128 bytes maximum Read Request size 001b 256 bytes maximum Read Request size 010b 512 bytes maximum Read Request size 011b 1024 bytes maximum Read Request size 100b 2048 bytes maximum Read Request size 101b 4096 bytes maximum Read Request size 110b Reserved 111b Reserved Functions that do not generate Read Requests larger than 128 bytes and Functions that do not generate Read Requests on their own behalf are permitted to implement this field as Read Only (RO) with a value of 000b. Default value of this field is 010b.	RW
15	*PCI Express to PCI/PCI-X Bridges:* **Bridge Configuration Retry Enable** – When Set, this bit enables PCI Express to PCI/PCI-X bridges to return Configuration Request Retry Status (CRS) in response to Configuration Requests that target devices below the bridge. Refer to the *PCI Express to PCI/PCI-X Bridge Specification, Revision 1.0* for further details. Default value of this bit is 0b. *Endpoints with Function Level Reset Capability set to 1b:* **Initiate Function Level Reset** – A write of 1b initiates Function Level Reset to the Function. The value read by software from this bit is always 0b. *All others:* **Reserved** – Must hardwire the bit to 0b.	*PCI Express to PCI/PCI-X Bridges:* RW *FLR Capable Endpoints:* RW *All others:* RsvdP

IMPLEMENTATION NOTE

Software UR Reporting Compatibility with 1.0a Devices

With 1.0a device Functions,[125] if the Unsupported Request Reporting Enable bit is Set, the Function when operating as a Completer will send an uncorrectable error Message (if enabled) when a UR error is detected. On platforms where an uncorrectable error Message is handled as a System Error, this will break PC-compatible Configuration Space probing, so software/firmware on such platforms may need to avoid setting the Unsupported Request Reporting Enable bit.

With device Functions implementing Role-Based Error Reporting, setting the Unsupported Request Reporting Enable bit will not interfere with PC-compatible Configuration Space probing, assuming that the severity for UR is left at its default of non-fatal. However, setting the Unsupported Request Reporting Enable bit will enable the Function to report UR errors[126] detected with posted Requests, helping avoid this case for potential silent data corruption.

On platforms where robust error handling and PC-compatible Configuration Space probing is required, it is suggested that software or firmware have the Unsupported Request Reporting Enable bit Set for Role-Based Error Reporting Functions, but clear for 1.0a Functions. Software or firmware can distinguish the two classes of Functions by examining the Role-Based Error Reporting bit in the Device Capabilities register.

[125] In this context, “1.0a devices” are devices that do not implement Role-Based Error Reporting.

[126] With Role-Based Error Reporting devices, setting the SERR# Enable bit in the Command register also implicitly enables UR reporting.

IMPLEMENTATION NOTE

Use of Max_Payload_Size

The Max_Payload_Size mechanism allows software to control the maximum payload in packets sent by Endpoints to balance latency versus bandwidth trade-offs, particularly for isochronous traffic.

If software chooses to program the Max_Payload_Size of various System Elements to non-default values, it must take care to ensure that each packet does not exceed the Max_Payload_Size parameter of any System Element along the packet's path. Otherwise, the packet will be rejected by the System Element whose Max_Payload_Size parameter is too small.

Discussion of specific algorithms used to configure Max_Payload_Size to meet this requirement is beyond the scope of this specification, but software should base its algorithm upon factors such as the following:

- the Max_Payload_Size capability of each System Element within a hierarchy
- awareness of when System Elements are added or removed through Hot-Plug operations
- knowing which System Elements send packets to each other, what type of traffic is carried, what type of transactions are used, or if packet sizes are constrained by other mechanisms

For the case of firmware that configures System Elements in preparation for running legacy operating system environments, the firmware may need to avoid programming a Max_Payload_Size above the default of 128 bytes, which is the minimum supported by Endpoints.

For example, if the operating system environment does not comprehend PCI Express, firmware probably should not program a non-default Max_Payload_Size for a hierarchy that supports Hot-Plug operations. Otherwise, if no software is present to manage Max_Payload_Size settings when a new element is added, improper operation may result. Note that a newly added element may not even support a Max_Payload_Size setting as large as the rest of the hierarchy, in which case software may need to deny enabling the new element or reduce the Max_Payload_Size settings of other elements.

IMPLEMENTATION NOTE

Use of Max_Read_Request_Size

The Max_Read_Request_Size mechanism allows improved control of bandwidth allocation in systems where Quality of Service (QoS) is important for the target applications. For example, an arbitration scheme based on counting Requests (and not the sizes of those Requests) provides imprecise bandwidth allocation when some Requesters use much larger sizes than others. The Max_Read_Request_Size mechanism can be used to force more uniform allocation of bandwidth, by restricting the upper size of Read Requests.

7.5.3.5 *Device Status Register (Offset 0Ah)*

The Device Status register provides information about PCI Express device (Function) specific parameters. Figure 7-25 details allocation of register fields in the Device Status register; Table 7-20 provides the respective bit definitions.

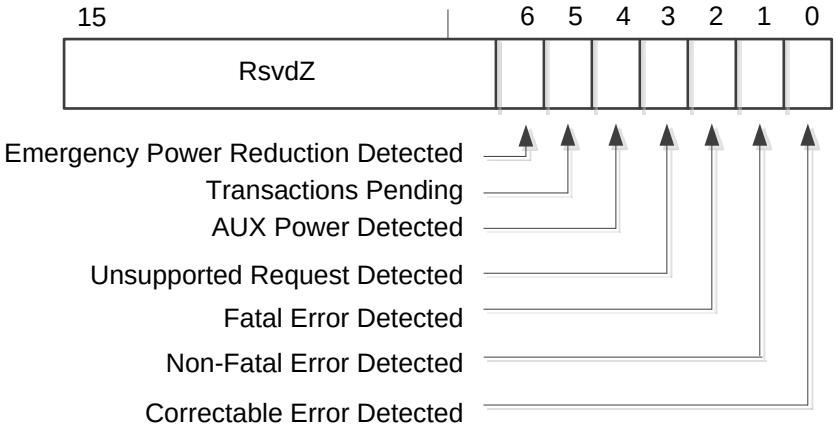


Figure 7-25: Device Status Register

Table 7-20: Device Status Register

Bit Location	Register Description	Attributes
0	**Correctable Error Detected** – This bit indicates status of correctable errors detected. Errors are logged in this register regardless of whether error reporting is enabled or not in the Device Control register. For a Multi-Function Device, each Function indicates status of errors as perceived by the respective Function. For Functions supporting Advanced Error Handling, errors are logged in this register regardless of the settings of the Correctable Error Mask register. Default value of this bit is 0b.	RW1C
1	**Non-Fatal Error Detected** – This bit indicates status of Non-fatal errors detected. Errors are logged in this register regardless of whether error reporting is enabled or not in the Device Control register. For a Multi-Function Device, each Function indicates status of errors as perceived by the respective Function. For Functions supporting Advanced Error Handling, errors are logged in this register regardless of the settings of the Uncorrectable Error Mask register. Default value of this bit is 0b.	RW1C

Bit Location	Register Description	Attributes
2	**Fatal Error Detected** – This bit indicates status of Fatal errors detected. Errors are logged in this register regardless of whether error reporting is enabled or not in the Device Control register. For a Multi-Function Device, each Function indicates status of errors as perceived by the respective Function. For Functions supporting Advanced Error Handling, errors are logged in this register regardless of the settings of the Uncorrectable Error Mask register. Default value of this bit is 0b.	RW1C
3	**Unsupported Request Detected** – This bit indicates that the Function received an Unsupported Request. Errors are logged in this register regardless of whether error reporting is enabled or not in the Device Control register. For a Multi-Function Device, each Function indicates status of errors as perceived by the respective Function. Default value of this bit is 0b.	RW1C
4	**AUX Power Detected** – Functions that require Aux power report this bit as Set if Aux power is detected by the Function.	RO
5	**Transactions Pending** – *Endpoints:* When Set, this bit indicates that the Function has issued Non-Posted Requests that have not been completed. A Function reports this bit cleared only when all outstanding Non-Posted Requests have completed or have been terminated by the Completion Timeout mechanism. This bit must also be cleared upon the completion of an FLR. *Root and Switch Ports:* When Set, this bit indicates that a Port has issued Non-Posted Requests on its own behalf (using the Port's own Requester ID) which have not been completed. The Port reports this bit cleared only when all such outstanding Non-Posted Requests have completed or have been terminated by the Completion Timeout mechanism. Note that Root and Switch Ports implementing only the functionality required by this document do not issue Non-Posted Requests on their own behalf, and therefore are not subject to this case. Root and Switch Ports that do not issue Non-Posted Requests on their own behalf hardwire this bit to 0b.	RO

Bit Location	Register Description	Attributes
6	**Emergency Power Reduction Detected –** This bit is Set when the Function is in the Emergency Power Reduction State. Whenever any condition is present that would cause the Emergency Power Reduction State to be entered, the Function remains in the Emergency Power Reduction State and writes to this bit have no effect. See Section 6.25 for additional details. Multi-Function Devices associated with an Upstream Port must Set this bit in all Functions that support Emergency Power Reduction State. This bit is RsvdZ if the Emergency Power Reduction Supported field is 00b (see Section 7.5.3.15). This bit is RsvdZ in Functions that are not associated with an Upstream Port. Default value is 0b.	RW1C

7.5.3.6 *Link Capabilities Register (Offset 0Ch)*

The Link Capabilities register identifies PCI Express Link specific capabilities. Figure 7-26 details allocation of register fields in the Link Capabilities register; Table 7-21 provides the respective bit definitions.

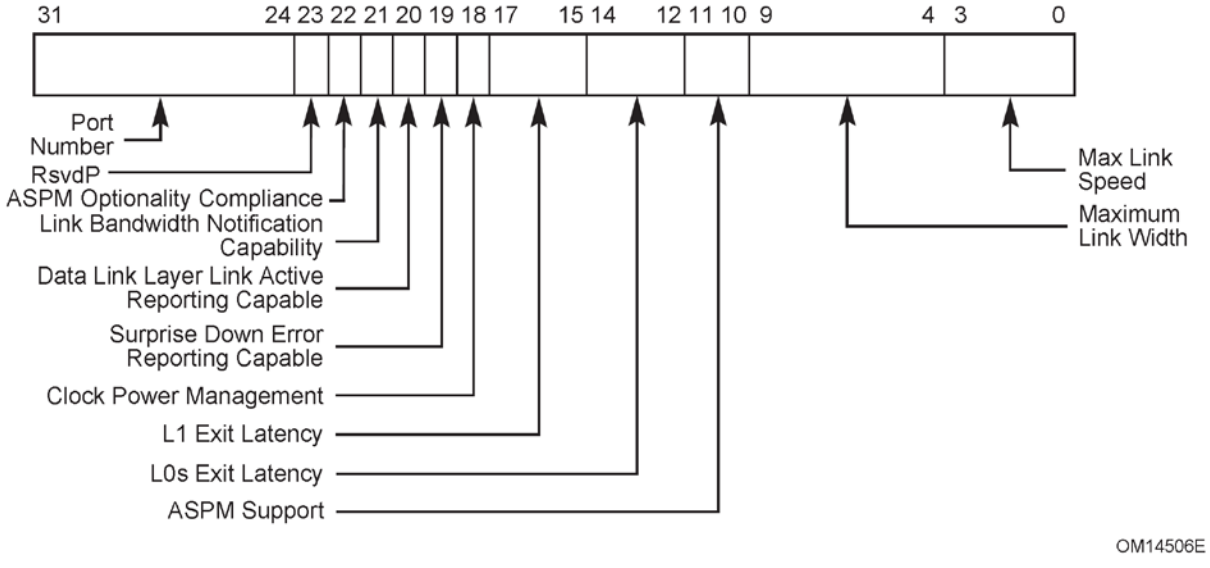


Figure 7-26: Link Capabilities Register

Table 7-21: Link Capabilities Register

Bit Location	Register Description	Attributes
3:0	**Max Link Speed** – This field indicates the maximum Link speed of the associated Port. The encoded value specifies a bit location in the Supported Link Speeds Vector (in the Link Capabilities 2 register) that corresponds to the maximum Link speed. Defined encodings are: 0001b Supported Link Speeds Vector field bit 0 0010b Supported Link Speeds Vector field bit 1 0011b Supported Link Speeds Vector field bit 2 0100b Supported Link Speeds Vector field bit 3 0101b Supported Link Speeds Vector field bit 4 0110b Supported Link Speeds Vector field bit 5 0111b Supported Link Speeds Vector field bit 6 All other encodings are reserved. Multi-Function Devices associated with an Upstream Port must report the same value in this field for all Functions.	RO
9:4	**Maximum Link Width** – This field indicates the maximum Link width (xN – corresponding to N Lanes) implemented by the component. This value is permitted to exceed the number of Lanes routed to the slot (Downstream Port), adapter connector (Upstream Port), or in the case of component-to-component connections, the actual wired connection width. Defined encodings are: 00 0001b x1 00 0010b x2 00 0100b x4 00 1000b x8 00 1100b x12 01 0000b x16 10 0000b x32 All other encodings are Reserved. Multi-Function Devices associated with an Upstream Port must report the same value in this field for all Functions.	RO

Bit Location	Register Description	Attributes
11:10	**Active State Power Management (ASPM) Support** – This field indicates the level of ASPM supported on the given PCI Express Link. See Section 5.4.1 for ASPM support requirements. Defined encodings are: 00b No ASPM Support 01b L0s Supported 10b L1 Supported 11b L0s and L1 Supported Multi-Function Devices associated with an Upstream Port must report the same value in this field for all Functions.	RO
14:12	**L0s Exit Latency** – This field indicates the L0s exit latency for the given PCI Express Link. The value reported indicates the length of time this Port requires to complete transition from L0s to L0. If L0s is not supported, the value is undefined; however, see the Implementation Note “Potential Issues With Legacy Software When L0s is Not Supported” in Section 5.4.1.1 for the recommended value. Defined encodings are: 000b Less than 64 ns 001b 64 ns to less than 128 ns 010b 128 ns to less than 256 ns 011b 256 ns to less than 512 ns 100b 512 ns to less than 1 µs 101b 1 µs to less than 2 µs 110b 2 µs-4 µs 111b More than 4 µs Note that exit latencies may be influenced by PCI Express reference clock configuration depending upon whether a component uses a common or separate reference clock. Multi-Function Devices associated with an Upstream Port must report the same value in this field for all Functions.	RO

Bit Location	Register Description	Attributes
17:15	**L1 Exit Latency** – This field indicates the L1 exit latency for the given PCI Express Link. The value reported indicates the length of time this Port requires to complete transition from ASPM L1 to L0. If ASPM L1 is not supported, the value is undefined. Defined encodings are: 000b Less than 1µs 001b 1 µs to less than 2 µs 010b 2 µs to less than 4 µs 011b 4 µs to less than 8 µs 100b 8 µs to less than 16 µs 101b 16 µs to less than 32 µs 110b 32 µs-64 µs 111b More than 64 µs Note that exit latencies may be influenced by PCI Express reference clock configuration depending upon whether a component uses a common or separate reference clock. Multi-Function Devices associated with an Upstream Port must report the same value in this field for all Functions.	RO
18	**Clock Power Management** – For Upstream Ports, a value of 1b in this bit indicates that the component tolerates the removal of any reference clock(s) via the “clock request” (CLKREQ#) mechanism when the Link is in the L1 and L2/L3 Ready Link states. A value of 0b indicates the component does not have this capability and that reference clock(s) must not be removed in these Link states. L1 PM Substates defines other semantics for the CLKREQ# signal, which are managed independently of Clock Power Management. This Capability is applicable only in form factors that support “clock request” (CLKREQ#) capability. For a Multi-Function Device associated with an Upstream Port, each Function indicates its capability independently. Power Management configuration software must only permit reference clock removal if all Functions of the Multi-Function Device indicate a 1b in this bit. For ARI Devices, all Functions must indicate the same value in this bit. For Downstream Ports, this bit must be hardwired to 0b.	RO
19	**Surprise Down Error Reporting Capable** – For a Downstream Port, this bit must be Set if the component supports the optional capability of detecting and reporting a Surprise Down error condition. For Upstream Ports and components that do not support this optional capability, this bit must be hardwired to 0b.	RO

Bit Location	Register Description	Attributes
20	**Data Link Layer Link Active Reporting Capable** – For a Downstream Port, this bit must be hardwired to 1b if the component supports the optional capability of reporting the DL_Active state of the Data Link Control and Management State Machine. For a hot-plug capable Downstream Port (as indicated by the Hot-Plug Capable bit of the Slot Capabilities register) or a Downstream Port that supports Link speeds greater than 5.0 GT/s, this bit must be hardwired to 1b. For Upstream Ports and components that do not support this optional capability, this bit must be hardwired to 0b.	RO
21	**Link Bandwidth Notification Capability** – A value of 1b indicates support for the Link Bandwidth Notification status and interrupt mechanisms. This capability is required for all Root Ports and Switch Downstream Ports supporting Links wider than x1 and/or multiple Link speeds. This field is not applicable and is Reserved for Endpoints, PCI Express to PCI/PCI-X bridges, and Upstream Ports of Switches. Functions that do not implement the Link Bandwidth Notification Capability must hardwire this bit to 0b.	RO
22	**ASPM Optionality Compliance** – This bit must be set to 1b in all Functions. Components implemented against certain earlier versions of this specification will have this bit set to 0b. Software is permitted to use the value of this bit to help determine whether to enable ASPM or whether to run ASPM compliance tests.	HwInit
31:24	**Port Number** – This field indicates the PCI Express Port number for the given PCI Express Link. Multi-Function Devices associated with an Upstream Port must report the same value in this field for all Functions.	HwInit

IMPLEMENTATION NOTE

Use of the ASPM Optionality Compliance Bit

Correct implementation and utilization of ASPM can significantly reduce Link power. However, ASPM feature implementations can be complex, and historically, some implementations have not been compliant to the specification. To address this, some of the ASPM optionality and ASPM entry requirements from earlier revisions of this document have been loosened. However, clear pass/fail compliance testing for ASPM features is also supported and expected.

The ASPM Optionality Compliance bit was created as a tool to establish clear expectations for hardware and software. This bit is Set to indicate hardware that conforms to the current specification, and this bit must be Set in components compliant to this specification.

System software as well as compliance software can assume that if this bit is Set, that the associated hardware conforms to the current specification. Hardware should be fully capable of supporting ASPM configuration management without needing component-specific treatment by system software.

For older hardware that does not have this bit Set, it is strongly recommended for system software to provide mechanisms to enable ASPM on components that work correctly with ASPM, and to disable ASPM on components that don't.

7.5.3.7 *Link Control Register (Offset 10h)*

The Link Control register controls PCI Express Link specific parameters. Figure 7-27 details allocation of register fields in the Link Control register; Table 7-22 provides the respective bit definitions.

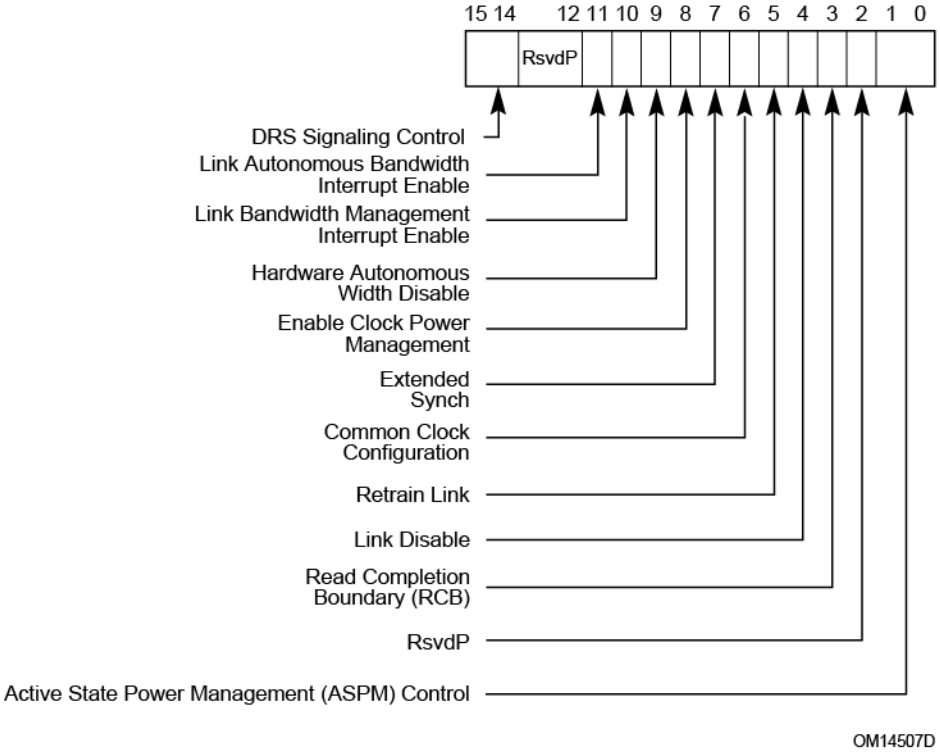


Figure 7-27: Link Control Register

Table 7-22: Link Control Register

Bit Location	Register Description	Attributes
1:0	**Active State Power Management (ASPM) Control** – This field controls the level of ASPM enabled on the given PCI Express Link. See Section 5.4.1.3 for requirements on when and how to enable ASPM. Defined encodings are: 00b Disabled 01bL0s Entry Enabled 10bL1 Entry Enabled 11bL0s and L1 Entry Enabled Note: "L0s Entry Enabled" enables the Transmitter to enter L0s. If L0s is supported, the Receiver must be capable of entering L0s even when the Transmitter is disabled from entering L0s (00b or 10b). ASPM L1 must be enabled by software in the Upstream component on a Link prior to enabling ASPM L1 in the Downstream component on that Link. When disabling ASPM L1, software must disable ASPM L1 in the Downstream component on a Link prior to disabling ASPM L1 in the Upstream component on that Link. ASPM L1 must only be enabled on the Downstream component if both components on a Link support ASPM L1. For Multi-Function Devices (including ARI Devices), it is recommended that software program the same value for this field in all Functions. For non-ARI Multi-Function Devices, only capabilities enabled in all Functions are enabled for the component as a whole. For ARI Devices, ASPM Control is determined solely by the setting in Function 0, regardless of Function 0's D-state. The settings in the other Functions always return whatever value software programmed for each, but otherwise are ignored by the component. Default value of this field is 00b unless otherwise required by a particular form factor.	RW

Bit Location	Register Description	Attributes
3	Root Ports: Read Completion Boundary (RCB) – Indicates the RCB value for the Root Port. Refer to Section 2.3.1.1 for the definition of the parameter RCB. Defined encodings are: 0b 64 byte 1b 128 byte This bit is hardwired for a Root Port and returns its RCB support capabilities. Endpoints and Bridges: Read Completion Boundary (RCB) – Optionally Set by configuration software to indicate the RCB value of the Root Port Upstream from the Endpoint or Bridge. Refer to Section 2.3.1.1 for the definition of the parameter RCB. Defined encodings are: 0b 64 byte 1b 128 byte Configuration software must only Set this bit if the Root Port Upstream from the Endpoint or Bridge reports an RCB value of 128 bytes (a value of 1b in the Read Completion Boundary bit). Default value of this bit is 0b. Functions that do not implement this feature must hardwire the bit to 0b. Switch Ports: Not applicable – must hardwire the bit to 0b	Root Ports: RO Endpoints and Bridges: RW Switch Ports: RO
4	**Link Disable** – This bit disables the Link by directing the LTSSM to the Disabled state when Set; this bit is Reserved on Endpoints, PCI Express to PCI/PCI-X bridges, and Upstream Ports of Switches. Writes to this bit are immediately reflected in the value read from the bit, regardless of actual Link state. After clearing this bit, software must honor timing requirements defined in Section 6.6.1 with respect to the first Configuration Read following a Conventional Reset. Default value of this bit is 0b.	RW
5	**Retrain Link** – A write of 1b to this bit initiates Link retraining by directing the Physical Layer LTSSM to the Recovery state. If the LTSSM is already in Recovery or Configuration, re-entering Recovery is permitted but not required. If the Port is in DPC when a write of 1b to this bit occurs, the result is undefined. Reads of this bit always return 0b. It is permitted to write 1b to this bit while simultaneously writing modified values to other fields in this register. If the LTSSM is not already in Recovery or Configuration, the resulting Link training must use the modified values. If the LTSSM is already in Recovery or Configuration, the modified values are not required to affect the Link training that's already in progress. This bit is not applicable and is Reserved for Endpoints, PCI Express to PCI/PCI-X bridges, and Upstream Ports of Switches. This bit always returns 0b when read.	RW

Bit Location	Register Description	Attributes
6	**Common Clock Configuration** – When Set, this bit indicates that this component and the component at the opposite end of this Link are operating with a distributed common reference clock. A value of 0b indicates that this component and the component at the opposite end of this Link are operating with asynchronous reference clock. For non-ARI Multi-Function Devices, software must program the same value for this bit in all Functions. If not all Functions are Set, then the component must as a whole assume that its reference clock is not common with the Upstream component. For ARI Devices, Common Clock Configuration is determined solely by the setting in Function 0. The settings in the other Functions always return whatever value software programmed for each, but otherwise are ignored by the component. Components utilize this common clock configuration information to report the correct L0s and L1 Exit Latencies. After changing the value in this bit in both components on a Link, software must trigger the Link to retrain by writing a 1b to the Retrain Link bit of the Downstream Port. Default value of this bit is 0b.	RW
7	**Extended Synch** – When Set, this bit forces the transmission of additional Ordered Sets when exiting the L0s state (see Section 4.2.4.5) and when in the Recovery state (see Section 4.2.6.4.1). This mode provides external devices (e.g., logic analyzers) monitoring the Link time to achieve bit and Symbol lock before the Link enters the L0 state and resumes communication. For Multi-Function Devices if any Function has this bit Set, then the component must transmit the additional Ordered Sets when exiting L0s or when in Recovery. Default value for this bit is 0b.	RW

Bit Location	Register Description	Attributes
8	**Enable Clock Power Management** – Applicable only for Upstream Ports and with form factors that support a “Clock Request” (CLKREQ#) mechanism, this bit operates as follows: 0bClock power management is disabled and device must hold CLKREQ# signal low. 1bWhen this bit is Set, the device is permitted to use CLKREQ# signal to power manage Link clock according to protocol defined in appropriate form factor specification. For a non-ARI Multi-Function Device, power-management-configuration software must only Set this bit if all Functions of the Multi-Function Device indicate a 1b in the Clock Power Management bit of the Link Capabilities register. The component is permitted to use the CLKREQ# signal to power manage Link clock only if this bit is Set for all Functions. For ARI Devices, Clock Power Management is enabled solely by the setting in Function 0. The settings in the other Functions always return whatever value software programmed for each, but otherwise are ignored by the component. The CLKREQ# signal may also be controlled via the L1 PM Substates mechanism. Such control is not affected by the setting of this bit. Downstream Ports and components that do not support Clock Power Management (as indicated by a 0b value in the Clock Power Management bit of the Link Capabilities register) must hardwire this bit to 0b. Default value of this bit is 0b, unless specified otherwise by the form factor specification.	RW
9	**Hardware Autonomous Width Disable** – When Set, this bit disables hardware from changing the Link width for reasons other than attempting to correct unreliable Link operation by reducing Link width. For a Multi-Function Device associated with an Upstream Port, the bit in Function 0 is of type RW, and only Function 0 controls the component’s Link behavior. In all other Functions of that device, this bit is of type RsvdP. Components that do not implement the ability autonomously to change Link width are permitted to hardwire this bit to 0b. Default value of this bit is 0b.	RW/RsvdP (see description)
10	**Link Bandwidth Management Interrupt Enable** – When Set, this bit enables the generation of an interrupt to indicate that the Link Bandwidth Management Status bit has been Set. This bit is not applicable and is Reserved for Endpoints, PCI Express-to-PCI/PCI-X bridges, and Upstream Ports of Switches. Functions that do not implement the Link Bandwidth Notification Capability must hardwire this bit to 0b. Default value of this bit is 0b.	RW

Bit Location	Register Description	Attributes
11	**Link Autonomous Bandwidth Interrupt Enable** – When Set, this bit enables the generation of an interrupt to indicate that the Link Autonomous Bandwidth Status bit has been Set. This bit is not applicable and is Reserved for Endpoints, PCI Express-to-PCI/PCI-X bridges, and Upstream Ports of Switches. Functions that do not implement the Link Bandwidth Notification Capability must hardwire this bit to 0b. Default value of this bit is 0b.	RW
15:14	**DRS Signaling Control** – Indicates the mechanism used to report reception of a DRS message. Must be implemented for Downstream Ports with the DRS Supported bit Set in the Link Capabilities 2 Register. Encodings are: 00bDRS not Reported: If DRS Supported is Set, receiving a DRS Message will set DRS Message Received in the Link Status 2 Register but will otherwise have no effect 01bDRS Interrupt Enabled: If the DRS Message Received bit in the Link Status 2 Register transitions from 0 to 1, and either MSI or MSI-X is enabled, an MSI or MSI-X interrupt is generated using the vector in Interrupt Message Number (section 7.5.3.2) 10bDRS to FRS Signaling Enabled: If the DRS Message Received bit in the Link Status 2 Register transitions from 0 to 1, the Port must send an FRS Message Upstream with the FRS Reason field set to DRS Message Received. Behavior is undefined if this field is set to 10b and the FRS Supported bit in the Device Capabilities 2 Register is Clear. Behavior is undefined if this field is set to 11b. Downstream Ports with the DRS Supported bit Clear in the Link Capabilities 2 register must hardwire this field to 00b. This field is Reserved for Upstream Ports. Default value of this field is 00b.	RW/RsvdP

IMPLEMENTATION NOTE

Software Compatibility with ARI Devices

With the ASPM Control field, Common Clock Configuration bit, and Enable Clock Power Management bit in the Link Control register, there are potential software compatibility issues with ARI Devices since these controls operate strictly off the settings in Function 0 instead of the settings in all Functions.

With compliant software, there should be no issues with the Common Clock Configuration bit, since software is required to set this bit the same in all Functions.

With the Enable Clock Power Management bit, there should be no compatibility issues with software that sets this bit the same in all Functions. However, if software does not set this bit the same in all Functions, and relies on each Function having the ability to prevent Clock Power Management from being enabled, such software may have compatibility issues with ARI Devices.

With the ASPM Control field, there should be no compatibility issues with software that sets this bit the same in all Functions. However, if software does not set this bit the same in all Functions, and relies on each Function in D0 state having the ability to prevent ASPM from being enabled, such software may have compatibility issues with ARI Devices.

IMPLEMENTATION NOTE

Avoiding Race Conditions When Using the Retrain Link Bit

When software changes Link control parameters and writes a 1b to the Retrain Link bit in order to initiate Link training using the new parameter settings, special care is required in order to avoid certain race conditions. At any instant the LTSSM may transition to the Recovery or Configuration state due to normal Link activity, without software awareness. If the LTSSM is already in Recovery or Configuration when software writes updated parameters to the Link Control register, as well as a 1b, to the Retrain Link bit, the LTSSM might not use the updated parameter settings with the current Link training, and the current Link training might not achieve the results that software intended.

To avoid this potential race condition, it is highly recommended that software use the following algorithm or something similar:

1. Software sets the relevant Link control parameters to the desired settings without writing a 1b to the Retrain Link bit.
2. Software polls the Link Training bit in the Link Status register until the value returned is 0b.
3. Software writes a 1b to the Retrain Link bit without changing any other fields in the Link Control register.

The above algorithm guarantees that Link training will be based on the Link control parameter settings that software intends.

IMPLEMENTATION NOTE

Use of the Slot Clock Configuration and Common Clock Configuration Bits

In order to determine the common clocking configuration of components on opposite ends of a Link that crosses a connector, two pieces of information are required. The following description defines these requirements.

The first necessary piece of information is whether the Port that connects to the slot uses a clock that has a common source and therefore constant phase relationship to the clock signal provided on the slot. This information is provided by the system side component through a hardware initialized bit (Slot Clock Configuration) in its Link Status register. Note that some electromechanical form factor specifications may require the Port that connects to the slot use a clock that has a common source to the clock signal provided on the slot.

The second necessary piece of information is whether the component on the adapter uses the clock supplied on the slot or one generated locally on the adapter. The adapter design and layout will determine whether the component is connected to the clock source provided by the slot. A component going onto this adapter should have some hardware initialized method for the adapter design/designer to indicate the configuration used for this particular adapter design. This

information is reported by bit 12 (Slot Clock Configuration) in the Link Status register of each Function in the Upstream Port. Note that some electromechanical form factor specifications may require the Port on the adapter to use the clock signal provided on the connector.

System firmware or software will read this value from the components on both ends of a physical Link. If both components report the use of a common clock connection this firmware/software will program bit 6 (Common Clock Configuration) of the Link Control register to 1b on both components connected to the Link. Each component uses this bit to determine the length of time required to re-synch its Receiver to the opposing component's Transmitter when exiting L0s.

This value is reported as a time value in bits 12-14 of the Link Capabilities register (offset 0Ch) and is sent to the opposing Transmitter as part of the initialization process as N_FTS. Components would be expected to require much longer synch times without common clocking and would therefore report a longer L0s exit latency in bits 12-14 of the Link Capabilities register and would send a larger number for N_FTS during training. This forces a requirement that whatever software changes this bit should force a Link retrain in order to get the correct N-FTS set for the Receivers at both ends of the Link.

7.5.3.8 *Link Status Register (Offset 12h)*

The Link Status register provides information about PCI Express Link specific parameters. Figure 7-28 details allocation of register fields in the Link Status register; Table 7-23 provides the respective bit definitions.

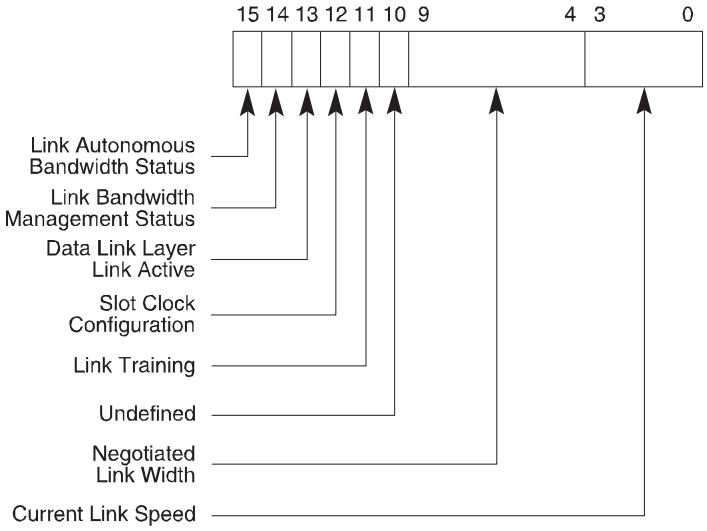


Figure 7-28: Link Status Register

Table 7-23: Link Status Register

Bit Location	Register Description	Attributes
3:0	**Current Link Speed** – This field indicates the negotiated Link speed of the given PCI Express Link. The encoded value specifies a bit location in the Supported Link Speeds Vector (in the Link Capabilities 2 register) that corresponds to the current Link speed. Defined encodings are: 0001b Supported Link Speeds Vector field bit 0 0010b Supported Link Speeds Vector field bit 1 0011b Supported Link Speeds Vector field bit 2 0100b Supported Link Speeds Vector field bit 3 0101b Supported Link Speeds Vector field bit 4 0110b Supported Link Speeds Vector field bit 5 0111b Supported Link Speeds Vector field bit 6 All other encodings are Reserved. The value in this field is undefined when the Link is not up.	RO
9:4	**Negotiated Link Width** – This field indicates the negotiated width of the given PCI Express Link. Defined encodings are: 00 0001b x1 00 0010b x2 00 0100b x4 00 1000b x8 00 1100b x12 01 0000b x16 10 0000b x32 All other encodings are Reserved. The value in this field is undefined when the Link is not up.	RO
10	**Undefined** – The value read from this bit is undefined. In previous versions of this specification, this bit was used to indicate a Link Training Error. System software must ignore the value read from this bit. System software is permitted to write any value to this bit.	RO

Bit Location	Register Description	Attributes
11	**Link Training** – This read-only bit indicates that the Physical Layer LTSSM is in the Configuration or Recovery state, or that 1b was written to the Retrain Link bit but Link training has not yet begun. Hardware clears this bit when the LTSSM exits the Configuration/Recovery state. This bit is not applicable and Reserved for Endpoints, PCI Express to PCI/PCI-X bridges, and Upstream Ports of Switches, and must be hardwired to 0b.	RO
12	**Slot Clock Configuration** – This bit indicates that the component uses the same physical reference clock that the platform provides on the connector. If the device uses an independent clock irrespective of the presence of a reference clock on the connector, this bit must be clear. For a Multi-Function Device, each Function must report the same value for this bit.	HwInit
13	**Data Link Layer Link Active** – This bit indicates the status of the Data Link Control and Management State Machine. It returns a 1b to indicate the DL_Active state, 0b otherwise. This bit must be implemented if the Data Link Layer Link Active Reporting Capable bit is 1b. Otherwise, this bit must be hardwired to 0b.	RO
14	**Link Bandwidth Management Status** – This bit is Set by hardware to indicate that either of the following has occurred without the Port transitioning through DL_Down status: • A Link retraining has completed following a write of 1b to the Retrain Link bit. Note: This bit is Set following any write of 1b to the Retrain Link bit, including when the Link is in the process of retraining for some other reason. • Hardware has changed Link speed or width to attempt to correct unreliable Link operation, either through an LTSSM timeout or a higher level process. This bit must be set if the Physical Layer reports a speed or width change was initiated by the Downstream component that was not indicated as an autonomous change. This bit is not applicable and is Reserved for Endpoints, PCI Express-to-PCI/PCI-X bridges, and Upstream Ports of Switches. Functions that do not implement the Link Bandwidth Notification Capability must hardwire this bit to 0b. The default value of this bit is 0b.	RW1C

Bit Location	Register Description	Attributes
15	**Link Autonomous Bandwidth Status** – This bit is Set by hardware to indicate that hardware has autonomously changed Link speed or width, without the Port transitioning through DL_Down status, for reasons other than to attempt to correct unreliable Link operation. This bit must be set if the Physical Layer reports a speed or width change was initiated by the Downstream component that was indicated as an autonomous change. This bit is not applicable and is Reserved for Endpoints, PCI Express-to-PCI/PCI-X bridges, and Upstream Ports of Switches. Functions that do not implement the Link Bandwidth Notification Capability must hardwire this bit to 0b. The default value of this bit is 0b.	RW1C

7.5.3.9 *Slot Capabilities Register (Offset 14h)*

The Slot Capabilities register identifies PCI Express slot specific capabilities. Figure 7-29 details allocation of register fields in the Slot Capabilities register; Table 7-24 provides the respective bit definitions.

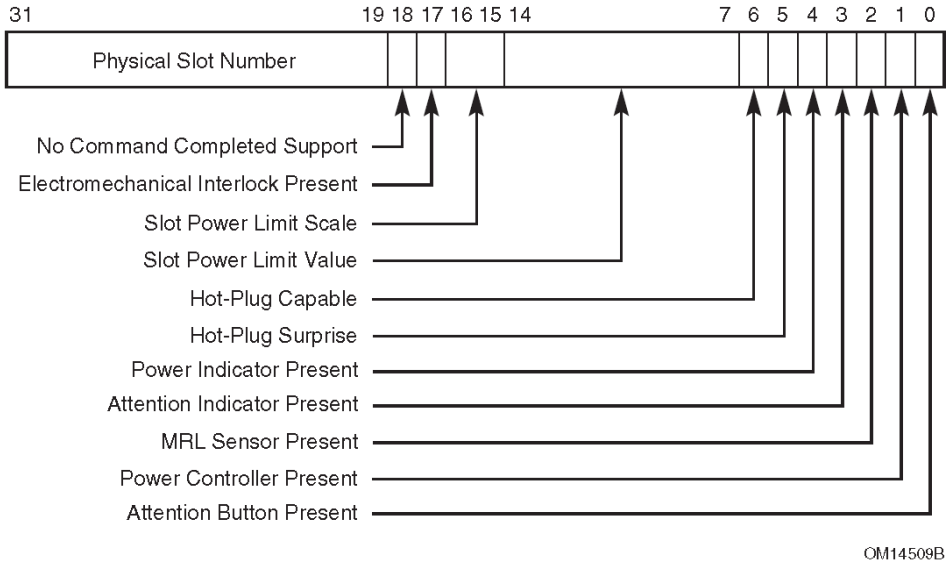


Figure 7-29: Slot Capabilities Register

Table 7-24: Slot Capabilities Register

Bit Location	Register Description	Attributes
0	**Attention Button Present** – When Set, this bit indicates that an Attention Button for this slot is electrically controlled by the chassis.	HwInit
1	**Power Controller Present** – When Set, this bit indicates that a software programmable Power Controller is implemented for this slot/adapter (depending on form factor).	HwInit
2	**MRL Sensor Present** – When Set, this bit indicates that an MRL Sensor is implemented on the chassis for this slot.	HwInit
3	**Attention Indicator Present** – When Set, this bit indicates that an Attention Indicator is electrically controlled by the chassis.	HwInit
4	**Power Indicator Present** – When Set, this bit indicates that a Power Indicator is electrically controlled by the chassis for this slot.	HwInit
5	**Hot-Plug Surprise** – When Set, this bit indicates that an adapter present in this slot might be removed from the system without any prior notification. This is a form factor specific capability. This bit is an indication to the operating system to allow for such removal without impacting continued software operation.	HwInit
6	**Hot-Plug Capable** – When Set, this bit indicates that this slot is capable of supporting hot-plug operations.	HwInit
14:7	**Slot Power Limit Value** – In combination with the Slot Power Limit Scale value, specifies the upper limit on power supplied by the slot (see Section 6.9) or by other means to the adapter. Power limit (in Watts) is calculated by multiplying the value in this field by the value in the Slot Power Limit Scale field except when the Slot Power Limit Scale field equals 00b (1.0x) and Slot Power Limit Value exceeds EFh, the following alternative encodings are used: F0h = 250 W Slot Power Limit F1h = 275 W Slot Power Limit F2h = 300 W Slot Power Limit F3h to FFh = Reserved for Slot Power Limit values above 300 W This register must be implemented if the Slot Implemented bit is Set. Writes to this register also cause the Port to send the Set_Slot_Power_Limit Message. The default value prior to hardware/firmware initialization is 0000 0000b.	HwInit

Bit Location	Register Description	Attributes
16:15	**Slot Power Limit Scale** – Specifies the scale used for the Slot Power Limit Value (see Section 6.9). Range of Values: 00b = 1.0x 01b = 0.1x 10b = 0.01x 11b = 0.001x This register must be implemented if the Slot Implemented bit is Set. Writes to this register also cause the Port to send the Set_Slot_Power_Limit Message. The default value prior to hardware/firmware initialization is 00b.	HwInit
17	**Electromechanical Interlock Present** – When Set, this bit indicates that an Electromechanical Interlock is implemented on the chassis for this slot.	HwInit
18	**No Command Completed Support** – When Set, this bit indicates that this slot does not generate software notification when an issued command is completed by the Hot-Plug Controller. This bit is only permitted to be Set if the hot-plug capable Port is able to accept writes to all fields of the Slot Control register without delay between successive writes.	HwInit
31:19	**Physical Slot Number** – This field indicates the physical slot number attached to this Port. This field must be hardware initialized to a value that assigns a slot number that is unique within the chassis, regardless of the form factor associated with the slot. This field must be initialized to zero for Ports connected to devices that are either integrated on the system board or integrated within the same silicon as the Switch device or Root Port.	HwInit

7.5.3.10 *Slot Control Register (Offset 18h)*

The Slot Control register controls PCI Express Slot specific parameters. Figure 7-30 details allocation of register fields in the Slot Control register; Table 7-25 provides the respective bit definitions.

Attention Indicator Control, Power Indicator Control, and Power Controller Control fields of the Slot Control register do not have a defined default value. If these fields are implemented, it is the responsibility of either system firmware or operating system software to (re)initialize these fields after a reset of the Link.

In hot-plug capable Downstream Ports, a write to the Slot Control register must cause a hot-plug command to be generated (see Section 6.7.3.2 for details on hot-plug commands). A write to the Slot Control register in a Downstream Port that is not hot-plug capable must not cause a hot-plug command to be executed.

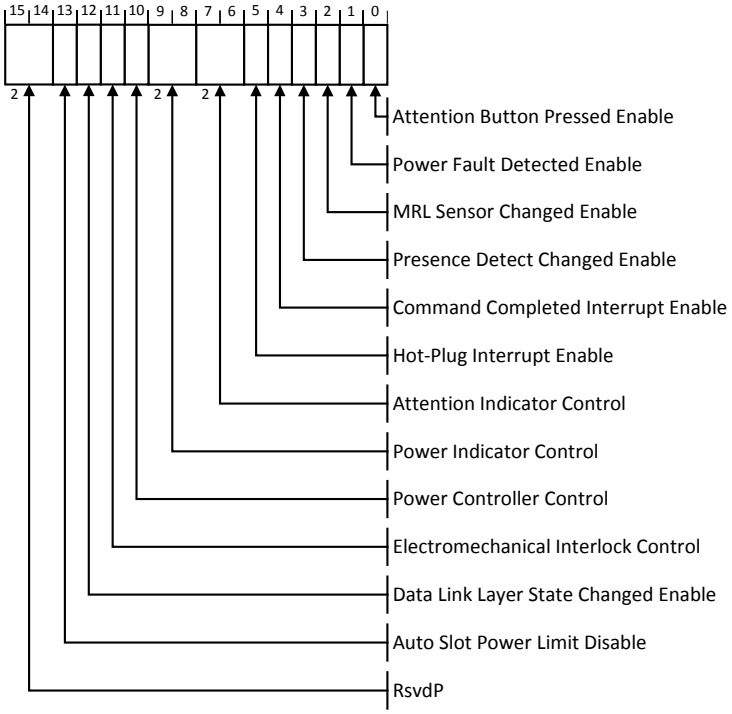


Figure 7-30: Slot Control Register

Table 7-25: Slot Control Register

Bit Location	Register Description	Attributes
0	**Attention Button Pressed Enable** – When Set to 1b, this bit enables software notification on an attention button pressed event (see Section 6.7.3). If the Attention Button Present bit in the Slot Capabilities register is 0b, this bit is permitted to be read-only with a value of 0b. Default value of this bit is 0b.	RW
1	**Power Fault Detected Enable** – When Set, this bit enables software notification on a power fault event (see Section 6.7.3). If a Power Controller that supports power fault detection is not implemented, this bit is permitted to be read-only with a value of 0b. Default value of this bit is 0b.	RW
2	**MRL Sensor Changed Enable** – When Set, this bit enables software notification on a MRL sensor changed event (see Section 6.7.3). If the MRL Sensor Present bit in the Slot Capabilities register is Clear, this bit is permitted to be read-only with a value of 0b. Default value of this bit is 0b.	RW

Bit Location	Register Description	Attributes
3	**Presence Detect Changed Enable** – When Set, this bit enables software notification on a presence detect changed event (see Section 6.7.3). If the Hot-Plug Capable bit in the Slot Capabilities register is 0b, this bit is permitted to be read-only with a value of 0b. Default value of this bit is 0b.	RW
4	**Command Completed Interrupt Enable** – If Command Completed notification is supported (if the No Command Completed Support bit in the Slot Capabilities register is 0b), when Set, this bit enables software notification when a hot-plug command is completed by the Hot-Plug Controller. If Command Completed notification is not supported, this bit must be hardwired to 0b. Default value of this bit is 0b.	RW
5	**Hot-Plug Interrupt Enable** – When Set, this bit enables generation of an interrupt on enabled hot-plug events. If the Hot Plug Capable bit in the Slot Capabilities register is Clear, this bit is permitted to be read-only with a value of 0b. Default value of this bit is 0b.	RW
7:6	**Attention Indicator Control** – If an Attention Indicator is implemented, writes to this field set the Attention Indicator to the written state. Reads of this field must reflect the value from the latest write, even if the corresponding hot-plug command is not complete, unless software issues a write without waiting, if required to, for the previous command to complete in which case the read value is undefined. Defined encodings are: 00b Reserved 01b On 10b Blink 11b Off Note: The default value of this field must be one of the non-Reserved values. If the Attention Indicator Present bit in the Slot Capabilities register is 0b, this bit is permitted to be read-only with a value of 00b.	RW

Bit Location	Register Description	Attributes
9:8	**Power Indicator Control** – If a Power Indicator is implemented, writes to this field set the Power Indicator to the written state. Reads of this field must reflect the value from the latest write, even if the corresponding hot-plug command is not complete, unless software issues a write without waiting, if required to, for the previous command to complete in which case the read value is undefined. Defined encodings are: 00b Reserved 01b On 10b Blink 11b Off Note: The default value of this field must be one of the non-Reserved values. If the Power Indicator Present bit in the Slot Capabilities register is 0b, this bit is permitted to be read-only with a value of 00b.	RW
10	**Power Controller Control** – If a Power Controller is implemented, this bit when written sets the power state of the slot per the defined encodings. Reads of this bit must reflect the value from the latest write, even if the corresponding hot-plug command is not complete, unless software issues a write, if required to, without waiting for the previous command to complete in which case the read value is undefined. Note that in some cases the power controller may autonomously remove slot power or not respond to a power-up request based on a detected fault condition, independent of the Power Controller Control setting. The defined encodings are: 0b Power On 1b Power Off If the Power Controller Present bit in the Slot Capabilities register is Clear, then writes to this bit have no effect and the read value of this bit is undefined.	RW
11	**Electromechanical Interlock Control** – If an Electromechanical Interlock is implemented, a write of 1b to this bit causes the state of the interlock to toggle. A write of 0b to this bit has no effect. A read of this bit always returns a 0b.	RW
12	**Data Link Layer State Changed Enable** – If the Data Link Layer Link Active Reporting capability is 1b, this bit enables software notification when Data Link Layer Link Active bit is changed. If the Data Link Layer Link Active Reporting Capable bit is 0b, this bit is permitted to be read-only with a value of 0b. Default value of this bit is 0b.	RW

Bit Location	Register Description	Attributes
13	**Auto Slot Power Limit Disable** – When Set, this disables the automatic sending of a Set_Slot_Power_Limit Message when a Link transitions from a non-DL_Up status to a DL_Up status. Downstream ports that don't support DPC are permitted to hardwire this bit to 0. Default value of this bit is implementation specific.	RW

7.5.3.11 *Slot Status Register (Offset 1Ah)*

The Slot Status register provides information about PCI Express Slot specific parameters. Figure 7-31 details allocation of register fields in the Slot Status register; Table 7-26 provides the respective bit definitions. Register fields for status bits not implemented by the device have the RsvdZ attribute.

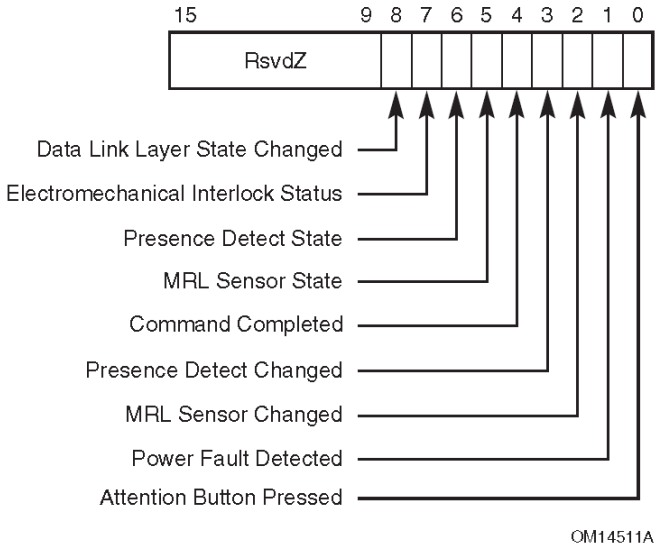


Figure 7-31: Slot Status Register

Table 7-26: Slot Status Register

Bit Location	Register Description	Attributes
0	**Attention Button Pressed** – If an Attention Button is implemented, this bit is Set when the attention button is pressed. If an Attention Button is not supported, this bit must not be Set.	RW1C
1	**Power Fault Detected** – If a Power Controller that supports power fault detection is implemented, this bit is Set when the Power Controller detects a power fault at this slot. Note that, depending on hardware capability, it is possible that a power fault can be detected at any time, independent of the Power Controller Control setting or the occupancy of the slot. If power fault detection is not supported, this bit must not be Set.	RW1C

Bit Location	Register Description	Attributes
2	**MRL Sensor Changed** – If an MRL sensor is implemented, this bit is Set when a MRL Sensor state change is detected. If an MRL sensor is not implemented, this bit must not be Set.	RW1C
3	**Presence Detect Changed** – This bit is Set when the value reported in the Presence Detect State bit is changed.	RW1C
4	**Command Completed** – If Command Completed notification is supported (if the No Command Completed Support bit in the Slot Capabilities register is 0b), this bit is Set when a hot-plug command has completed and the Hot-Plug Controller is ready to accept a subsequent command. The Command Completed status bit is Set as an indication to host software that the Hot-Plug Controller has processed the previous command and is ready to receive the next command; it provides no guarantee that the action corresponding to the command is complete. If Command Completed notification is not supported, this bit must be hardwired to 0b.	RW1C
5	**MRL Sensor State** – This bit reports the status of the MRL sensor if implemented. Defined encodings are: 0b MRL Closed 1b MRL Open	RO
6	**Presence Detect State** – This bit indicates the presence of an adapter in the slot, reflected by the logical “OR” of the Physical Layer in-band presence detect mechanism and, if present, any out-of-band presence detect mechanism defined for the slot’s corresponding form factor. Note that the in-band presence detect mechanism requires that power be applied to an adapter for its presence to be detected. Consequently, form factors that require a power controller for hot-plug must implement a physical pin presence detect mechanism. Defined encodings are: 0b Slot Empty 1b Adapter Present in slot This bit must be implemented on all Downstream Ports that implement slots. For Downstream Ports not connected to slots (where the Slot Implemented bit of the PCI Express Capabilities register is 0b), this bit must be hardwired to 1b.	RO
7	**Electromechanical Interlock Status** – If an Electromechanical Interlock is implemented, this bit indicates the status of the Electromechanical Interlock. Defined encodings are: 0b Electromechanical Interlock Disengaged 1b Electromechanical Interlock Engaged	RO

Bit Location	Register Description	Attributes
8	**Data Link Layer State Changed** – This bit is Set when the value reported in the Data Link Layer Link Active bit of the Link Status register is changed. In response to a Data Link Layer State Changed event, software must read the Data Link Layer Link Active bit of the Link Status register to determine if the Link is active before initiating configuration cycles to the hot plugged device.	RW1C

IMPLEMENTATION NOTE

No Slot Power Controller

For slots that do not implement a power controller, software must ensure that system power planes are enabled to provide power to slots prior to reading presence detect state.

7.5.3.12 *Root Control Register (Offset 1Ch)*

The Root Control register controls PCI Express Root Complex specific parameters. Figure 7-32 details allocation of register fields in the Root Control register; Table 7-27 provides the respective bit definitions.

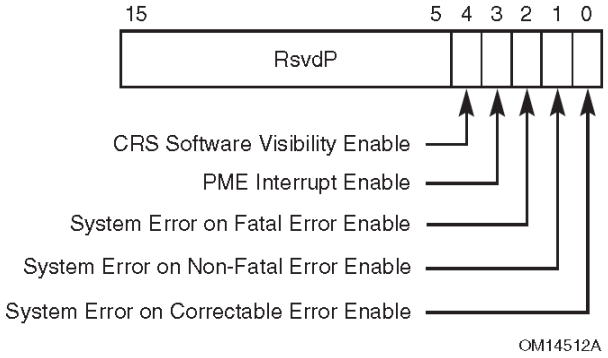


Figure 7-32: Root Control Register

Table 7-27: Root Control Register

Bit Location	Register Description	Attributes
0	**System Error on Correctable Error Enable** – If Set, this bit indicates that a System Error should be generated if a correctable error (ERR_COR) is reported by any of the devices in the Hierarchy Domain associated with this Root Port, or by the Root Port itself. The mechanism for signaling a System Error to the system is system specific. Root Complex Event Collectors provide support for the above-described functionality for RCiEPs. Default value of this bit is 0b.	RW

Bit Location	Register Description	Attributes
1	**System Error on Non-Fatal Error Enable** – If Set, this bit indicates that a System Error should be generated if a Non-fatal error (ERR_NONFATAL) is reported by any of the devices in the Hierarchy Domain associated with this Root Port, or by the Root Port itself. The mechanism for signaling a System Error to the system is system specific. Root Complex Event Collectors provide support for the above-described functionality for RCiEPs. Default value of this bit is 0b.	RW
2	**System Error on Fatal Error Enable** – If Set, this bit indicates that a System Error should be generated if a Fatal error (ERR_FATAL) is reported by any of the devices in the Hierarchy Domain associated with this Root Port, or by the Root Port itself. The mechanism for signaling a System Error to the system is system specific. Root Complex Event Collectors provide support for the above-described functionality for RCiEPs. Default value of this bit is 0b.	RW
3	**PME Interrupt Enable** – When Set, this bit enables PME interrupt generation upon receipt of a PME Message as reflected in the PME Status bit (see Table 7-29). A PME interrupt is also generated if the PME Status bit is Set when this bit is changed from Clear to Set (see Section 0). Default value of this bit is 0b.	RW
4	**CRS Software Visibility Enable** – When Set, this bit enables the Root Port to return Configuration Request Retry Status (CRS) Completion Status to software (see Section 2.3.1). Root Ports that do not implement this capability must hardwire this bit to 0b. Default value of this bit is 0b.	RW

7.5.3.13 *Root Capabilities Register (Offset 1Eh)*

The Root Capabilities register identifies PCI Express Root Port specific capabilities. Figure 7-33 details allocation of register fields in the Root Capabilities register; Table 7-28 provides the respective bit definitions.

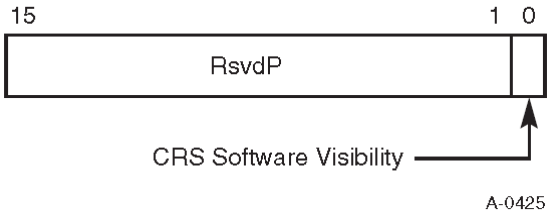


Figure 7-33: Root Capabilities Register

Table 7-28: Root Capabilities Register

Bit Location	Register Description	Attributes
0	**CRS Software Visibility** – When Set, this bit indicates that the Root Port is capable of returning Configuration Request Retry Status (CRS) Completion Status to software (see Section 2.3.1).	RO

7.5.3.14 *Root Status Register (Offset 20h)*

The Root Status register provides information about PCI Express device specific parameters. Figure 7-34 details allocation of register fields in the Root Status register; Table 7-29 provides the respective bit definitions.

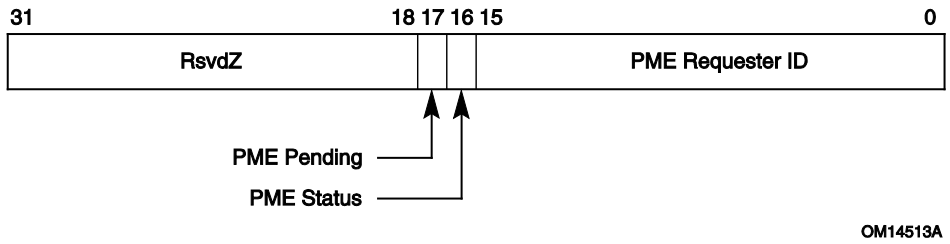


Figure 7-34: Root Status Register

Table 7-29: Root Status Register

Bit Location	Register Description	Attributes
15:0	**PME Requester ID** – This field indicates the PCI Requester ID of the last PME Requester. This field is only valid when the PME Status bit is Set.	RO
16	**PME Status** – This bit indicates that PME was asserted by the PME Requester indicated in the PME Requester ID field. Subsequent PMEs are kept pending until the status register is cleared by software by writing a 1b. Default value of this bit is 0b.	RW1C
17	**PME Pending** – This bit indicates that another PME is pending when the PME Status bit is Set. When the PME Status bit is cleared by software; the PME is delivered by hardware by setting the PME Status bit again and updating the PME Requester ID field appropriately. The PME Pending bit is cleared by hardware if no more PMEs are pending.	RO

7.5.3.15 *Device Capabilities 2 Register (Offset 24h)*

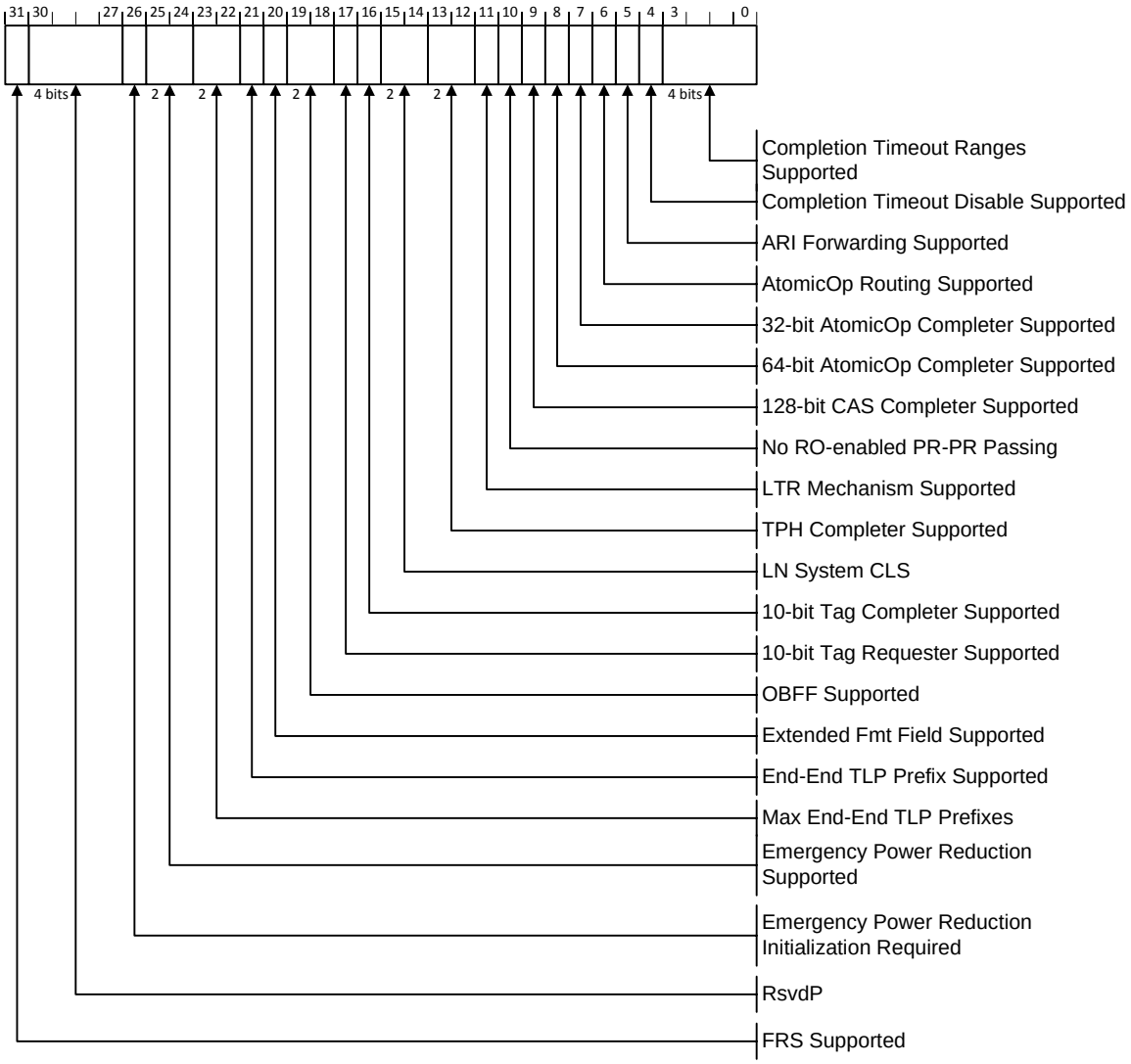


Figure 7-35: Device Capabilities 2 Register

Table 7-30: Device Capabilities 2 Register

Bit Location	Register Description	Attributes
3:0	**Completion Timeout Ranges Supported** – This field indicates device Function support for the optional Completion Timeout programmability mechanism. This mechanism allows system software to modify the Completion Timeout value. This field is applicable only to Root Ports, Endpoints that issue Requests on their own behalf, and PCI Express to PCI/PCI-X Bridges that take ownership of Requests issued on PCI Express. For all other Functions this field is Reserved and must be hardwired to 0000b. Four time value ranges are defined: Range A: 50 µs to 10 ms Range B: 10 ms to 250 ms Range C: 250 ms to 4 s Range D: 4 s to 64 s Bits are set according to the table below to show timeout value ranges supported. 0000b Completion Timeout programming not supported – the Function must implement a timeout value in the range 50 µs to 50 ms. 0001b Range A 0010b Range B 0011b Ranges A and B 0110b Ranges B and C 0111b Ranges A, B, and C 1110b Ranges B, C, and D 1111b Ranges A, B, C, and D All other values are Reserved. It is strongly recommended that the Completion Timeout mechanism not expire in less than 10 ms.	HwInit

Bit Location	Register Description	Attributes
4	**Completion Timeout Disable Supported** – A value of 1b indicates support for the Completion Timeout Disable mechanism. The Completion Timeout Disable mechanism is required for Endpoints that issue Requests on their own behalf and PCI Express to PCI/PCI-X Bridges that take ownership of Requests issued on PCI Express. This mechanism is optional for Root Ports. For all other Functions this field is Reserved and must be hardwired to 0b.	RO
5	**ARI Forwarding Supported** – Applicable only to Switch Downstream Ports and Root Ports; must be 0b for other Function types. This bit must be set to 1b if a Switch Downstream Port or Root Port supports this optional capability. See Section 6.13 for additional details.	RO
6	**AtomicOp Routing Supported** – Applicable only to Switch Upstream Ports, Switch Downstream Ports, and Root Ports; must be 0b for other Function types. This bit must be set to 1b if the Port supports this optional capability. See Section 6.15 for additional details.	RO
7	**32-bit AtomicOp Completer Supported** – Applicable to Functions with Memory Space BARs as well as all Root Ports; must be 0b otherwise. Includes FetchAdd, Swap, and CAS AtomicOps. This bit must be set to 1b if the Function supports this optional capability. See Section 6.15.3.1 for additional RC requirements.	RO
8	**64-bit AtomicOp Completer Supported** – Applicable to Functions with Memory Space BARs as well as all Root Ports; must be 0b otherwise. Includes FetchAdd, Swap, and CAS AtomicOps. This bit must be set to 1b if the Function supports this optional capability. See Section 6.15.3.1 for additional RC requirements.	RO
9	**128-bit CAS Completer Supported** – Applicable to Functions with Memory Space BARs as well as all Root Ports; must be 0b otherwise. This bit must be set to 1b if the Function supports this optional capability. See Section 6.15 for additional details.	RO
10	**No RO-enabled PR-PR Passing** – If this bit is Set, the routing element never carries out the passing permitted by Table 2-39 entry A2b that is associated with the Relaxed Ordering Attribute field being Set. This bit applies only for Switches and RCs that support peer-to-peer traffic between Root Ports. This bit applies only to Posted Requests being forwarded through the Switch or RC and does not apply to traffic originating or terminating within the Switch or RC itself. All Ports on a Switch or RC must report the same value for this bit. For all other functions, this bit must be 0b.	HwInit

Bit Location	Register Description	Attributes
11	**LTR Mechanism Supported** – A value of 1b indicates support for the optional Latency Tolerance Reporting (LTR) mechanism. Root Ports, Switches and Endpoints are permitted to implement this capability. For a Multi-Function Device associated with an Upstream Port, each Function must report the same value for this bit. For Bridges and other Functions that do not implement this capability, this bit must be hardwired to 0b.	RO
13:12	**TPH Completer Supported** – Value indicates Completer support for TPH or Extended TPH. Applicable only to Root Ports and Endpoints. For all other Functions, this field is Reserved. Defined Encodings are: 00b TPH and Extended TPH Completer not supported. 01b TPH Completer supported; Extended TPH Completer not supported. 10b Reserved. 11b Both TPH and Extended TPH Completer supported. See Section 6.17 for details.	RO
15:14	**LN System CLS** – Applicable only to Root Ports and RCRBs; must be 00b for all other Function types. This field indicates if the Root Port or RCRB supports LN protocol as an LN Completer, and if so, what cacheline size is in effect. Encodings are: 00b LN Completer either not supported or not in effect 01b LN Completer with 64-byte cachelines in effect 10b LN Completer with 128-byte cachelines in effect 11b Reserved	HwInit
16	10-Bit Tag Completer Supported – If this bit is Set, the Function supports 10-Bit Tag Completer capability; otherwise, the Function does not. See Section 2.2.6.2.	HwInit
17	10-Bit Tag Requester Supported – If this bit is Set, the Function supports 10-Bit Tag Requester capability; otherwise, the Function does not. This bit must not be Set if the 10-Bit Tag Completer Supported bit is Clear. If the Function is an RCiEP, this bit must be Clear if the RC does not support 10-Bit Tag Completer capability for Requests coming from this RCiEP. Note that 10-Bit Tag field generation must be enabled by the 10-Bit Tag Requester Enable bit in the Device Control 2 register of the Requester Function before 10-Bit Tags can be generated by the Requester. See Section 2.2.6.2.	HwInit

Bit Location	Register Description	Attributes
19:18	**OBFF Supported** – This field indicates if OBFF is supported and, if so, what signaling mechanism is used. 00b OBFF Not Supported 01b OBFF supported using Message signaling only 10b OBFF supported using WAKE# signaling only 11b OBFF supported using WAKE# and Message signaling The value reported in this field must indicate support for WAKE# signaling only if: - for a Downstream Port, driving the WAKE# signal for OBFF is supported and the connector or component connected Downstream is known to receive that same WAKE# signal - for an Upstream Port, receiving the WAKE# signal for OBFF is supported and, if the component is on an add-in-card, that the component is connected to the WAKE# signal on theconnector. Root Ports, Switch Ports, and Endpoints are permitted to implement this capability. For a Multi-Function Device associated with an Upstream Port, each Function must report the same value for this field. For Bridges and Ports that do not implement this capability, this field must be hardwired to 00b.	HwInit
20	**Extended Fmt Field Supported** – If Set, the Function supports the 3-bit definition of the Fmt field. If Clear, the Function supports a 2-bit definition of the Fmt field. See Section 2.2. Must be Set for Functions that support End-End TLP Prefixes. All Functions in an Upstream Port must have the same value for this bit. Each Downstream Port of a component may have a different value for this bit. It is strongly recommended that Functions support the 3-bit definition of the Fmt field.	RO
21	**End-End TLP Prefix Supported** – Indicates whether End-End TLP Prefix support is offered by a Function. Values are: 0b No Support 1b Support is provided to receive TLPs containing End-End TLP Prefixes. All Ports of a Switch must have the same value for this bit**.**	HwInit

Bit Location	Register Description	Attributes
23:22	**Max End-End TLP Prefixes –** Indicates the maximum number of End-End TLP Prefixes supported by this Function. See Section 2.2.10.2 for important details. Values are: 01b 1 End-End TLP Prefix 10b 2 End-End TLP Prefixes 11b 3 End-End TLP Prefixes 00b 4 End-End TLP Prefixes If End-End TLP Prefix Supported is Clear, this field is RsvdP. Different Root Ports that have the End-End TLP Prefix Supported bit Set are permitted to report different values for this field. For Switches where End-End TLP Prefix Supported is Set, this field must be 00b indicating support for up to four End-End TLP Prefixes.	HwInit
25:24	**Emergency Power Reduction Supported** – Indicates support level of the optional Emergency Power Reduction State feature. A Function can enter Emergency Power Reduction State autonomously, or based on one of two mechanisms defined by the associated Form Factor Specification. Functions that are in the Emergency Power Reduction State consume less power. The Emergency Power Reduction mechanism permits a chassis to request add-in cards to rapidly enter Emergency Power Reduction State without involving system software. See Section 6.25 for additional details. Values are: 00b Emergency Power Reduction State not supported 01b Emergency Power Reduction State is supported and is triggered by Device Specific mechanism(s) 10b Emergency Power Reduction State is supported and is triggered either by the mechanism defined in the corresponding Form Factor specification or by Device Specific mechanism(s) 11b Reserved This field is RsvdP in Functions that are not associated with an Upstream Port. For Multi-Function Devices associated with an Upstream Port, all Functions that report a non-zero value for this field, must report the same non-zero value for this field. Default value is 00b. After reset, once this field returns a non-zero value, it must continue to return the same non-zero value, until the next reset.	HwInit

Bit Location	Register Description	Attributes
26	**Emergency Power Reduction Initialization Required** – If Set, the Function requires complete or partial initialization upon exit from the Emergency Power Reduction State. If Clear, the Function requires no software intervention to return to normal operation upon exit from the Emergency Power Reduction State. See Section 6.25 for additional details. For Multi-Function Devices associated with an Upstream Port, all Functions must report the same value for this bit. This bit is RsvdP in Functions that are not associated with an Upstream Port. Default value is 0b. After reset, when this field returns a non-zero value, it must continue to return the same non-zero value.	HwInit
31	**FRS Supported** – When Set, indicates support for the optional Function Readiness Status (FRS) capability. Must be Set for all Functions that support generation or reception capabilities of FRS Messages. Must not be Set by Switch Functions that do not generate FRS Messages on their own behalf.	HwInit

IMPLEMENTATION NOTE

Use of the No RO-enabled PR-PR Passing Bit

The No RO-enabled PR-PR Passing bit allows platforms to utilize PCI Express switching elements on the path between a requester and completer for requesters that could benefit from a slightly less relaxed ordering model. An example is a device that cannot ensure that multiple overlapping posted writes to the same address are outstanding at the same time. The method by which such a device is enabled to utilize this mode is beyond the scope of this specification.

7.5.3.16 *Device Control 2 Register (Offset 28h)*

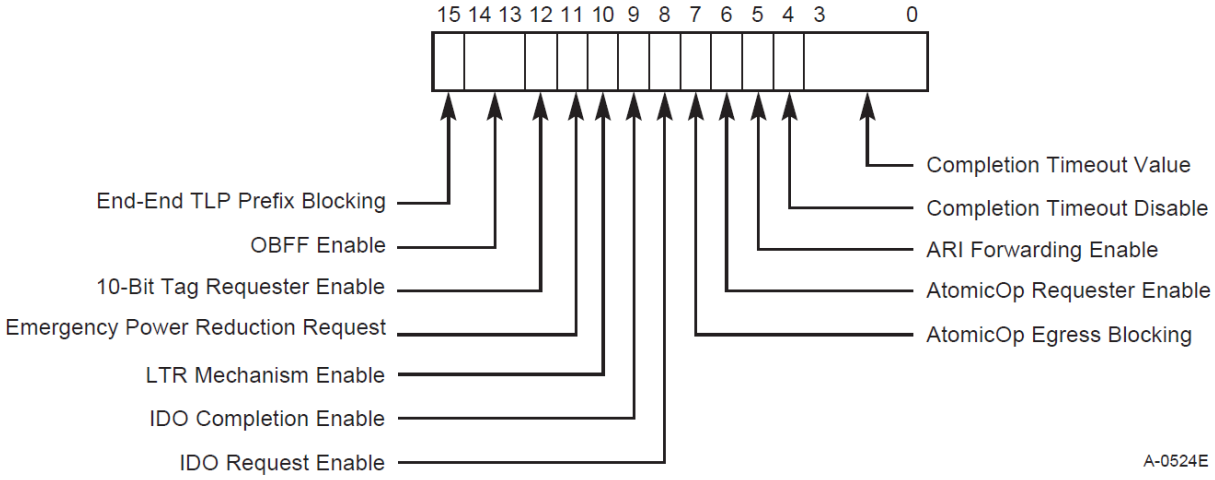


Figure 7-36: Device Control 2 Register

Table 7-31: Device Control 2 Register

Bit Location	Register Description	Attributes
3:0	**Completion Timeout Value** – In device Functions that support Completion Timeout programmability, this field allows system software to modify the Completion Timeout value. This field is applicable to Root Ports, Endpoints that issue Requests on their own behalf, and PCI Express to PCI/PCI-X Bridges that take ownership of Requests issued on PCI Express. For all other Functions this field is Reserved and must be hardwired to 0000b. A Function that does not support this optional capability must hardwire this field to 0000b and is required to implement a timeout value in the range 50 µs to 50 ms. Functions that support Completion Timeout programmability must support the values given below corresponding to the programmability ranges indicated in the Completion Timeout Ranges Supported field. Defined encodings: 0000b Default range: 50 µs to 50 ms It is strongly recommended that the Completion Timeout mechanism not expire in less than 10 ms. Values available if Range A (50 µs to 10 ms) programmability range is supported: 0001b 50 µs to 100 µs 0010b 1 ms to 10 ms Values available if Range B (10 ms to 250 ms) programmability range is supported: 0101b 16 ms to 55 ms 0110b 65 ms to 210 ms Values available if Range C (250 ms to 4 s) programmability range is supported: 1001b 260 ms to 900 ms 1010b 1 s to 3.5 s Values available if the Range D (4 s to 64 s) programmability range is supported: 1101b 4 s to 13 s 1110b 17 s to 64 s Values not defined above are Reserved. Software is permitted to change the value in this field at any time. For Requests already pending when the Completion Timeout Value is changed, hardware is permitted to use either the new or the old value for the outstanding Requests, and is permitted to base the start time for each Request either on when this value was changed or on when each request was issued. The default value for this field is 0000b.	RW

Bit Location	Register Description	Attributes
4	**Completion Timeout Disable** – When Set, this bit disables the Completion Timeout mechanism. This bit is required for all Functions that support the Completion Timeout Disable Capability. Functions that do not support this optional capability are permitted to hardwire this bit to 0b Software is permitted to Set or Clear this bit at any time. When Set, the Completion Timeout detection mechanism is disabled. If there are outstanding Requests when the bit is cleared, it is permitted but not required for hardware to apply the completion timeout mechanism to the outstanding Requests. If this is done, it is permitted to base the start time for each Request on either the time this bit was cleared or the time each Request was issued. The default value for this bit is 0b.	RW
5	**ARI Forwarding Enable** – When set, the Downstream Port disables its traditional Device Number field being 0 enforcement when turning a Type 1 Configuration Request into a Type 0 Configuration Request, permitting access to Extended Functions in an ARI Device immediately below the Port. See Section 6.13. Default value of this bit is 0b. Must be hardwired to 0b if the ARI Forwarding Supported bit is 0b.	RW
6	**AtomicOp Requester Enable** – Applicable only to Endpoints and Root Ports; must be hardwired to 0b for other Function types. The Function is allowed to initiate AtomicOp Requests only if this bit and the Bus Master Enable bit in the Command register are both Set. This bit is required to be RW if the Endpoint or Root Port is capable of initiating AtomicOp Requests, but otherwise is permitted to be hardwired to 0b. This bit does not serve as a capability bit. This bit is permitted to be RW even if no AtomicOp Requester capabilities are supported by the Endpoint or Root Port. Default value of this bit is 0b.	RW
7	**AtomicOp Egress Blocking** – Applicable and mandatory for Switch Upstream Ports, Switch Downstream Ports, and Root Ports that implement AtomicOp routing capability; otherwise must be hardwired to 0b. When this bit is Set, AtomicOp Requests that target going out this Egress Port must be blocked. See Section 6.15.2. Default value of this bit is 0b.	RW
8	**IDO Request Enable** – If this bit is Set, the Function is permitted to set the ID-Based Ordering (IDO) bit (Attr[2]) of Requests it initiates (see Section 2.2.6.3 and Section 2.4). Endpoints, including RC Integrated Endpoints, and Root Ports are permitted to implement this capability. A Function is permitted to hardwire this bit to 0b if it never sets the IDO attribute in Requests. Default value of this bit is 0b.	RW

Bit Location	Register Description	Attributes
9	**IDO Completion Enable** – If this bit is Set, the Function is permitted to set the ID-Based Ordering (IDO) bit (Attr[2]) of Completions it returns (see Section 2.2.6.3 and Section 2.4). Endpoints, including RC Integrated Endpoints, and Root Ports are permitted to implement this capability. A Function is permitted to hardwire this bit to 0b if it never sets the IDO attribute in Completions. Default value of this bit is 0b.	RW
10	**LTR Mechanism Enable** – When Set to 1b, this bit enables Upstream Ports to send LTR messages and Downstream Ports to process LTR Messages. For a Multi-Function Device associated with an Upstream Port of a device that implements LTR, the bit in Function 0 is RW, and only Function 0 controls the component's Link behavior. In all other Functions of that device, this bit is RsvdP. Functions that do not implement the LTR mechanism are permitted to hardwire this bit to 0b. Default value of this bit is 0b. For Downstream Ports, this bit must be reset to the default value if the Port goes to DL_Down status.	RW/RsvdP
11	**Emergency Power Reduction Request** – If Set, all Functions in the component that support Emergency Power Reduction State must enter the Emergency Power Reduction State. If Clear these Functions must exit the Emergency Power Reduction State if no other reasons exist to preclude exiting this state. See Section 6.25 for additional details. This bit is implemented in the lowest numbered Function associated with an Upstream Port that has a non-zero value in the Emergency Power Reduction Supported field. This bit is RsvdP in all other Functions. Default is 0b.	RW/RsvdP
12	**10-Bit Tag Requester Enable** – This bit, in combination with the Extended Tag Field Enable bit in the Device Control register, determines how many Tag field bits a Requester is permitted to use. When the 10-Bit Tag Requester Enable bit is Set, the Requester is permitted to use 10-Bit Tags. See Section 2.2.6.2 for complete details. If software changes the value of this bit while the Function has outstanding Non-Posted Requests, the result is undefined. Functions that do not implement 10-Bit Tag Requester capability are permitted to hardwire this bit to 0b. Default value of this bit is 0b.	RW

Bit Location	Register Description	Attributes
14:13	**OBFF Enable** – This field enables the OBFF mechanism and selects the signaling method. 00b Disabled 01b Enabled using Message signaling [Variation A] 10b Enabled using Message signaling [Variation B] 11b Enabled using WAKE# signaling See Section 6.19 for an explanation of the above encodings. This field is required for all Ports that support the OBFF Capability. For a Multi-Function Device associated with an Upstream Port of a Device that implements OBFF, the field in Function 0 is of type RW, and only Function 0 controls the Component's behavior. In all other Functions of that Device, this field is of type RsvdP. Ports that do not implement OBFF are permitted to hardwire this field to 00b. Default value of this field is 00b.	RW/RsvdP (see description)
15	**End-End TLP Prefix Blocking** – Controls whether the routing function is permitted to forward TLPs containing an End-End TLP Prefix. Values are: 0b Forwarding Enabled – Function is permitted to send TLPs with End-End TLP Prefixes. 1b Forwarding Blocked – Function is not permitted to send TLPs with End-End TLP Prefixes. This bit affects TLPs that exit the Switch/Root Complex using the associated Port. It does not affect TLPs forwarded internally within the Switch/Root Complex. It does not affect TLPs that enter through the associated Port, that originate in the associated Port or originate in a Root Complex Integrated Device integrated with the associated Port. As described in Section 2.2.10.2, blocked TLPs are reported by the TLP Prefix Blocked Error. The default value of this bit is 0b. This bit is hardwired to 1b in Root Ports that support End-End TLP Prefixes but do not support forwarding of End-End TLP Prefixes. This bit is applicable to Root Ports and Switch Ports where the End-End TLP Prefix Supported bit is Set. This bit is not applicable and is RsvdP in all other cases.	RW (see description)

7.5.3.17 *Device Status 2 Register (Offset 2Ah)*

This section is a placeholder. There are no capabilities that require this register.

This register must be treated by software as RsvdZ.

7.5.3.18 *Link Capabilities 2 Register (Offset 2Ch)*

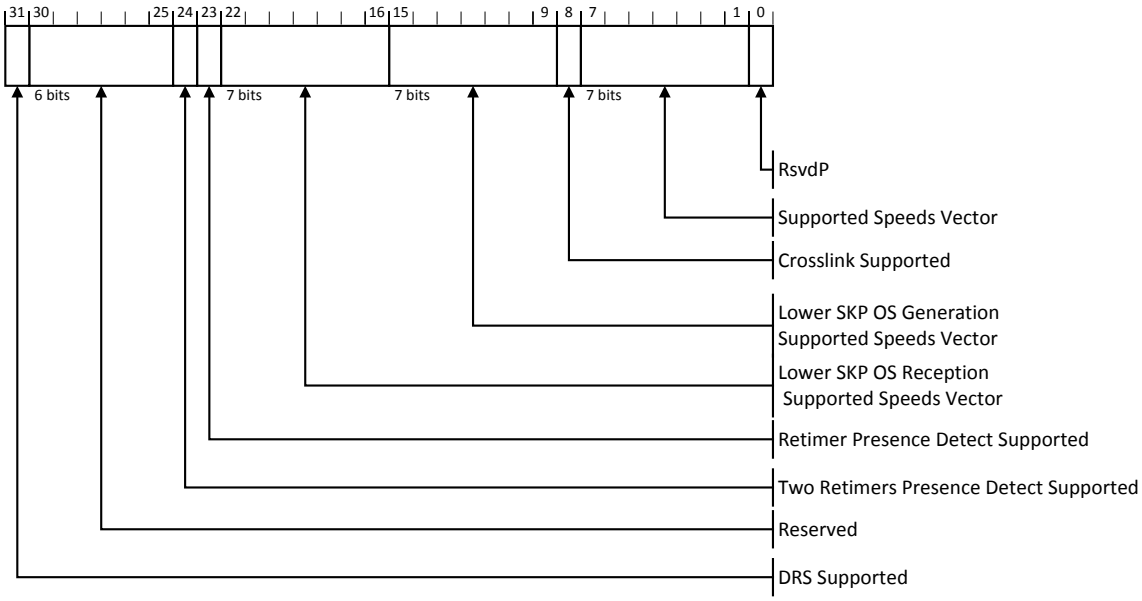


Figure 7-37: Link Capabilities 2 Register

Table 7-32: Link Capabilities 2 Register

Bit Location	Register Description	Attributes
7:1	**Supported Link Speeds Vector** – This field indicates the supported Link speed(s) of the associated Port. For each bit, a value of 1b indicates that the corresponding Link speed is supported; otherwise, the Link speed is not supported. See Section 8.2.1 for further requirements. Bit definitions within this field are: Bit 0 2.5 GT/s Bit 1 5.0 GT/s Bit 2 8.0 GT/s Bit 3 16.0 GT/s Bits 6:4 RsvdP Multi-Function Devices associated with an Upstream Port must report the same value in this field for all Functions.	HwInit/RsvdP
8	**Crosslink Supported** – When set to 1b, this bit indicates that the associated Port supports crosslinks (see Section 4.2.6.3.1). When set to 0b on a Port that supports Link speeds of 8.0 GT/s or higher, this bit indicates that the associated Port does not support crosslinks. When set to 0b on a Port that only supports Link speeds of 2.5 GT/s or 5.0 GT/s, this bit provides no information regarding the Port's level of crosslink support. It is recommended that this bit be Set in any Port that supports crosslinks even though doing so is only required for Ports that also support operating at 8.0 GT/s or higher Link speeds. Note: Software should use this bit when referencing fields whose definition depends on whether or not the Port supports crosslinks (see Section 7.7.3.4). Multi-Function Devices associated with an Upstream Port must report the same value in this field for all Functions.	RO

Bit Location	Register Description	Attributes
15:9	**Lower SKP OS Generation Supported Speeds Vector** – If this field is non-zero, it indicates that the Port, when operating at the indicated speed(s) supports SRIS and also supports software control of the SKP Ordered Set transmission scheduling rate. Bit definitions within this field are: Bit 0 2.5 GT/s Bit 1 5.0 GT/s Bit 2 8.0 GT/s Bit 3 16.0 GT/s Bits 6:4 RsvdP Multi-Function Devices associated with an Upstream Port must report the same value in this field for all Functions. Behavior is undefined if a bit is Set in this field and the corresponding bit is not Set in the Supported Link Speeds Vector.	HwInit/RsvdP
22:16	**Lower SKP OS Reception Supported Speeds Vector** – If this field is non-zero, it indicates that the Port, when operating at the indicated speed(s) supports SRIS and also supports receiving SKP OS at the rate defined for SRNS while running in SRIS. Bit definitions within this field are: Bit 0 2.5 GT/s Bit 1 5.0 GT/s Bit 2 8.0 GT/s Bit 3 16.0 GT/s Bits 6:4 RsvdP Multi-Function Devices associated with an Upstream Port must report the same value in this field for all Functions. Behavior is undefined if a bit is Set in this field and the corresponding bit is not Set in the Supported Link Speeds Vector.	HwInit/RsvdP
23	**Retimer Presence Detect Supported –** When set to 1b, this bit indicates that the associated Port supports detection and reporting of Retimer presence. This bit must be set to 1b in a Port when the Supported Link Speeds Vector of the Link Capabilities 2 register indicates support for a Link speed of 16.0 GT/s or higher. It is permitted to be set to 1b regardless of the supported Link speeds. Multi-Function Devices associated with an Upstream Port must report the same value in this field for all Functions.	HwInit/RsvdP

Bit Location	Register Description	Attributes
24	**Two Retimers Presence Detect Supported –** When set to 1b, this bit indicates that the associated Port supports detection and reporting of two Retimers presence. This bit must be set to 1b in a Port when the Supported Link Speeds Vector of the Link Capabilities 2 register indicates support for a Link speed of 16.0 GT/s or higher. It is permitted to be set to 1b regardless of the supported Link speeds if the Retimer Presence Detect Supported bit is also set to 1b. Multi-Function Devices associated with an Upstream Port must report the same value in this field for all Functions.	HwInit/RsvdP
31	**DRS Supported** – When Set, indicates support for the optional Device Readiness Status (DRS) capability. Must be Set in Downstream Ports that support DRS. Must be Set in Downstream Ports that support FRS. For Upstream Ports that support DRS, it is strongly recommended that this bit be Set in Function 0. For all other Functions associated with an Upstream Port, this bit must be Clear.[127] Must be Clear in Functions that are not associated with a Port. RsvdP in all other Functions.	HwInit/RsvdP

IMPLEMENTATION NOTE

Software Management of Link Speeds With Earlier Hardware

Hardware components compliant to versions prior to the *PCI Express Base Specification, Revision 3.0* either did not implement the Link Capabilities 2 register, or the register was Reserved.

For software to determine the supported Link speeds for components where the Link Capabilities 2 register is either not implemented, or the value of its Supported Link Speeds Vector is 0000000b, software can read bits 3:0 of the Link Capabilities register (now defined to be the Max Link Speed field), and interpret the value as follows:

0001b2.5 GT/s Link speed supported

0010b5.0 GT/s and 2.5 GT/s Link speeds supported

For such components, the encoding of the values for the Current Link Speed field (in the Link Status register) and Target Link Speed field (in the Link Control 2 register) is the same as above.

[127] It is expressly permitted for Upstream Ports to send DRS Messages even when the DRS Supported bit is Clear.

IMPLEMENTATION NOTE

Software Management of Link Speeds With Future Hardware

It is strongly encouraged that software primarily utilize the Supported Link Speeds Vector instead of the Max Link Speed field, so that software can determine the exact set of supported speeds on current and future hardware. This can avoid software being confused if a future specification defines Links that do not require support for all slower speeds.

7.5.3.19 *Link Control 2 Register (Offset 30h)*

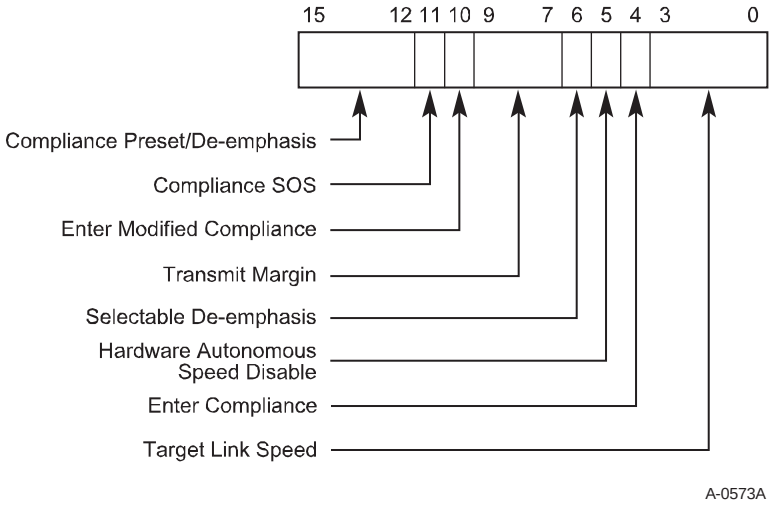


Figure 7-38: Link Control 2 Register

Table 7-33: Link Control 2 Register

Bit Location	Register Description	Attributes
3:0	**Target Link Speed** – For Downstream Ports, this field sets an upper limit on Link operational speed by restricting the values advertised by the Upstream component in its training sequences. The encoded value specifies a bit location in the Supported Link Speeds Vector (in the Link Capabilities 2 register) that corresponds to the desired target Link speed. Defined encodings are: 0001b Supported Link Speeds Vector field bit 0 0010b Supported Link Speeds Vector field bit 1 0011b Supported Link Speeds Vector field bit 2 0100b Supported Link Speeds Vector field bit 3 0101b Supported Link Speeds Vector field bit 4 0110b Supported Link Speeds Vector field bit 5 0111b Supported Link Speeds Vector field bit 6 All other encodings are Reserved. If a value is written to this field that does not correspond to a supported speed (as indicated by the Supported Link Speeds Vector), the result is undefined. If either of the Enter Compliance or Enter Modified Compliance bits are implemented, then this field must also be implemented. The default value of this field is the highest Link speed supported by the component (as reported in the Max Link Speed field of the Link Capabilities register) unless the corresponding platform/form factor requires a different default value. For both Upstream and Downstream Ports, this field is used to set the target compliance mode speed when software is using the Enter Compliance bit to force a Link into compliance mode. For Upstream Ports, if the Enter Compliance bit is Clear, this field is permitted to have no effect. For a Multi-Function Device associated with an Upstream Port, the field in Function 0 is of type RWS, and only Function 0 controls the component's Link behavior. In all other Functions of that device, this field is of type RsvdP. Components that support only the 2.5 GT/s speed are permitted to hardwire this field to 0000b.	RWS/RsvdP (see description)

Bit Location	Register Description	Attributes
4	**Enter Compliance** – Software is permitted to force a Link to enter Compliance mode (at the speed indicated in the Target Link Speed field and the de-emphasis/preset level indicated by the Compliance Preset/De-emphasis field) by setting this bit to 1b in both components on a Link and then initiating a hot reset on the Link. Default value of this bit following Fundamental Reset is 0b. For a Multi-Function Device associated with an Upstream Port, the bit in Function 0 is of type RWS, and only Function 0 controls the component's Link behavior. In all other Functions of that device, this bit is of type RsvdP. Components that support only the 2.5 GT/s speed are permitted to hardwire this bit to 0b. This bit is intended for debug, compliance testing purposes only. System firmware and software is allowed to modify this bit only during debug or compliance testing. In all other cases, the system must ensure that this bit is set to the default value.	RWS/RsvdP (see description)
5	**Hardware Autonomous Speed Disable** – When Set, this bit disables hardware from changing the Link speed for device-specific reasons other than attempting to correct unreliable Link operation by reducing Link speed. Initial transition to the highest supported common link speed is not blocked by this bit. For a Multi-Function Device associated with an Upstream Port, the bit in Function 0 is of type RWS, and only Function 0 controls the component's Link behavior. In all other Functions of that device, this bit is of type RsvdP. Functions that do not implement the associated mechanism are permitted to hardwire this bit to 0b. Default value of this bit is 0b.	RWS/RsvdP (see description)
6	**Selectable De-emphasis** – When the Link is operating at 5.0 GT/s speed, this bit is used to control the transmit de-emphasis of the link in specific situations. See Section 4.2.6 for detailed usage information. Encodings: 1b -3.5 dB 0b -6 dB When the Link is not operating at 5.0 GT/s speed, the setting of this bit has no effect. Components that support only the 2.5 GT/s speed are permitted to hardwire this bit to 0b. This bit is not applicable and Reserved for Endpoints, PCI Express to PCI/PCI-X bridges, and Upstream Ports of Switches.	HwInit

Bit Location	Register Description	Attributes
9:7	**Transmit Margin** – This field controls the value of the non-deemphasized voltage level at the Transmitter pins. This field is reset to 000b on entry to the LTSSM Polling.Configuration substate (see Chapter 4 for details of how the Transmitter voltage level is determined in various states). Encodings: 000b Normal operating range 001b-111b As defined in Section 8.3.4 not all encodings are required to be implemented. For a Multi-Function Device associated with an Upstream Port, the field in Function 0 is of type RWS, and only Function 0 controls the component's Link behavior. In all other Functions of that device, this field is of type RsvdP. Default value of this field is 000b. Components that support only the 2.5 GT/s speed are permitted to hardwire this bit to 000b. This field is intended for debug, compliance testing purposes only. System firmware and software is allowed to modify this field only during debug or compliance testing. In all other cases, the system must ensure that this field is set to the default value.	RWS/RsvdP (see description)
10	**Enter Modified Compliance** – When this bit is set to 1b, the device transmits Modified Compliance Pattern if the LTSSM enters Polling.Compliance substate. Components that support only the 2.5 GT/s speed are permitted to hardwire this bit to 0b. For a Multi-Function Device associated with an Upstream Port, the bit in Function 0 is of type RWS, and only Function 0 controls the component's Link behavior. In all other Functions of that device, this bit is of type RsvdP. Default value of this bit is 0b. This bit is intended for debug, compliance testing purposes only. System firmware and software is allowed to modify this bit only during debug or compliance testing. In all other cases, the system must ensure that this bit is set to the default value.	RWS/RsvdP (see description)

Bit Location	Register Description	Attributes
11	**Compliance SOS** – When set to 1b, the LTSSM is required to send SKP Ordered Sets between sequences when sending the Compliance Pattern or Modified Compliance Pattern. For a Multi-Function Device associated with an Upstream Port, the bit in Function 0 is of type RWS, and only Function 0 controls the component's Link behavior. In all other Functions of that device, this bit is of type RsvdP. The default value of this bit is 0b. This bit is applicable when the Link is operating at 2.5 GT/s or 5.0 GT/s data rates only. Components that support only the 2.5 GT/s speed are permitted to hardwire this bit to 0b.	RWS/RsvdP (see description)
15:12	**Compliance Preset/De-emphasis** – For 8.0 GT/s and higher Data Rate: This field sets the Transmitter Preset in Polling.Compliance state if the entry occurred due to the Enter Compliance bit being 1b. The encodings are defined in Section 4.2.3.2. Results are undefined if a reserved preset encoding is used when entering Polling.Compliance in this way. For 5.0 GT/s Data Rate: This field sets the de-emphasis level in Polling.Compliance state if the entry occurred due to the Enter Compliance bit being 1b. Defined Encodings are: 0001b -3.5 dB 0000b -6 dB When the Link is operating at 2.5 GT/s, the setting of this field has no effect. Components that support only 2.5 GT/s speed are permitted to hardwire this field to 0000b. For a Multi-Function Device associated with an Upstream Port, the field in Function 0 is of type RWS, and only Function 0 controls the component's Link behavior. In all other Functions of that device, this field is of type RsvdP. The default value of this field is 0000b. This field is intended for debug and compliance testing purposes. System firmware and software is allowed to modify this field only during debug or compliance testing. In all other cases, the system must ensure that this field is set to the default value.	RWS/RsvdP (see description)

IMPLEMENTATION NOTE

Selectable De-emphasis Usage

Selectable De-emphasis setting is applicable only to Root Ports and Switch Downstream Ports. The De-emphasis setting is implementation specific and depends on the platform or enclosure in which the Root Port or the Switch Downstream Port is located. System firmware or hardware strapping is used to configure the selectable de-emphasis value. In cases where system firmware cannot be used to set the de-emphasis value (for example, a hot plugged Switch), hardware strapping must be used to set the de-emphasis value.

7.5.3.20 *Link Status 2 Register (Offset 32h)*

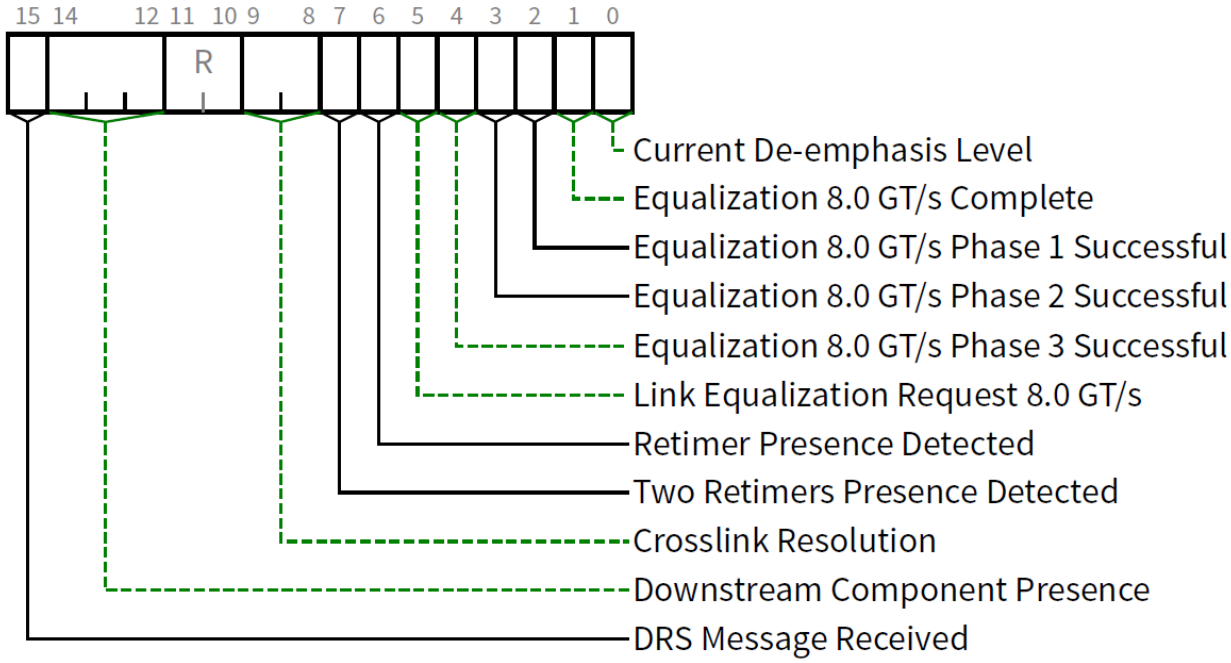


Figure 7-39: Link Status 2 Register

Table 7-34: Link Status 2 Register

Bit Location	Register Description	Attributes
0	**Current De-emphasis Level** – When the Link is operating at 5.0 GT/s speed, this bit reflects the level of de-emphasis. Encodings: 1b -3.5 dB 0b -6 dB The value in this bit is undefined when the Link is not operating at 5.0 GT/s speed. Components that support only the 2.5 GT/s speed are permitted to hardwire this bit to 0b. For components that support speeds greater than 2.5 GT/s, Multi-Function Devices associated with an Upstream Port must report the same value in this field for all Functions of the Port.	RO
1	**Equalization 8.0 GT/s Complete** – When set to 1b, this bit indicates that the Transmitter Equalization procedure at the 8.0 GT/s data rate has completed. Details of the Transmitter Equalization process and when this bit needs to be set to 1b is provided in Section 4.2.6.4.2. The default value of this bit is 0b. For Multi-Function Upstream Port, this bit must be implemented in Function 0 and RsvdZ in other Functions. Components that only support speeds below 8.0 GT/s are permitted to hardwire this bit to 0b.	ROS
2	**Equalization 8.0 GT/s Phase 1 Successful** – When set to 1b, this bit indicates that Phase 1 of the 8.0 GT/s Transmitter Equalization procedure has successfully completed. Details of the Transmitter Equalization process and when this bit needs to be set to 1b is provided in Section 4.2.6.4.2. The default value of this bit is 0b. For Multi-Function Upstream Port, this bit must be implemented in Function 0 and RsvdZ in other Functions. Components that only support speeds below 8.0 GT/s are permitted to hardwire this bit to 0b.	ROS
3	**Equalization 8.0 GT/s Phase 2 Successful** – When set to 1b, this bit indicates that Phase 2 of the 8.0 GT/s Transmitter Equalization procedure has successfully completed. Details of the Transmitter Equalization process and when this bit needs to be set to 1b is provided in Section 4.2.6.4.2. The default value of this bit is 0b. For Multi-Function Upstream Port, this bit must be implemented in Function 0 and RsvdZ in other Functions. Components that only support speeds below 8.0 GT/s are permitted to hardwire this bit to 0b.	ROS

Bit Location	Register Description	Attributes
4	**Equalization 8.0 GT/s Phase 3 Successful** – When set to 1b, this bit indicates that Phase 3 of the 8.0 GT/s Transmitter Equalization procedure has successfully completed. Details of the Transmitter Equalization process and when this bit needs to be set to 1b is provided in Section 4.2.6.4.2. The default value of this bit is 0b. For Multi-Function Upstream Port, this bit must be implemented in Function 0 and RsvdZ in other Functions. Components that only support speeds below 8.0 GT/s are permitted to hardwire this bit to 0b.	ROS
5	**Link Equalization Request** 8.0 GT/s – This bit is Set by hardware to request the 8.0 GT/s Link equalization process to be performed on the Link. Refer to Sections 4.2.3 and 4.2.6.4.2 for details. The default value of this bit is 0b. For Multi-Function Upstream Port, this bit must be implemented in Function 0 and RsvdZ in other Functions. Components that only support speeds below 8.0 GT/s are permitted to hardwire this bit to 0b.	RW1CS
6	**Retimer Presence Detected –** When set to 1b, this bit indicates that a Retimer was present during the most recent Link negotiation. Refer to Section 4.2.6.3.5.1 for details. The default value of this bit is 0b. This bit is required for Ports that have the Retimer Presence Detect Supported bit of the Link Capabilities 2 register set to 1b. Ports that have the Retimer Presence Detect Supported bit set to 0b are permitted to hardwire this bit to 0b. For Multi-Function Devices associated with an Upstream Port, this bit must be implemented in Function 0 and is RsvdZ in all other Functions.	ROS/RsvdZ
7	**Two Retimers Presence Detected** – When set to 1b, this bit indicates that two Retimers were present during the most recent Link negotiation. Refer to Section 4.2.6.3.5.1 for details. The default value of this bit is 0b. This bit is required for Ports that have the Two Retimers Presence Detect Supported bit of the Link Capabilities 2 register set to 1b. Ports that have the Two Retimers Presence Detect Supported bit set to 0b are permitted to hardwire this bit to 0b**.** For Multi-Function Devices associated with an Upstream Port, this bit must be implemented in Function 0 and RsvdZ in all other Functions.	ROS/RsvdZ

Bit Location	Register Description	Attributes
9:8	**Crosslink Resolution –** This field indicates the state of the Crosslink negotiation. It must be implemented if Crosslink Supported is Set and the Port supports 16.0 GT/s or higher data rate. It is permitted to be implemented in all other Ports. If Crosslink Supported is Clear, this field may be hardwired to 01b or 10b. Encoding is: 00bCrosslink Resolution is not supported. No information is provided regarding the status of the Crosslink negotiation. 01bCrosslink negotiation resolved as an Upstream Port. 10bCrosslink negotiation resolved as a Downstream Port. 11bCrosslink negotiation is not completed. Once a value of 01b or 10b is returned in this field, that value must continue to be returned while the Link is Up.	RO
14:12	**Downstream Component Presence** – This field indicates the presence and DRS status for the Downstream Component, if any, connected to the Link; defined values are: 000bLink Down – Presence Not Determined 001bLink Down – Component Not Present indicates the Downstream Port (DP) has determined that a Downstream Component is not present 010bLink Down – Component Present indicates the DP has determined that a Downstream Component is present, but the Data Link Layer is not active 011b Reserved 100bLink Up – Component Presentindicates the DP has determined that a Downstream Component is present, but no DRS Message has been received since the Data Link Layer became active 101bLink Up – Component Present and DRS Received indicates the DP has received a DRS Message since the Data Link Layer became active 110b Reserved 111b Reserved Component Presence state must be determined by the logical “OR” of the Physical Layer in-band presence detect mechanism and, if present, any out-of-band presence detect mechanism implemented for the Link. If no out-of-band presence detect mechanism is implemented, then Component Presence state must be determined solely by the Physical Layer in-band presence detect mechanism. This field must be implemented in any Downstream Port where the DRS Supported bit is Set in the Link Capabilities 2 register. This field is RsvdZ for all other Functions. Default value of this field is 000b.	RO/RsvdZ

Bit Location	Register Description	Attributes
15	**DRS Message Received** – This bit must be Set whenever the Port receives a DRS Message. This bit must be Cleared in DL_Down. This bit must be implemented in any Downstream Port where the DRS Supported bit is Set in the Link Capabilities 2 register. This bit is RsvdZ for all other Functions. Default value of this bit is 0b.	RW1C / RsvdZ

7.5.3.21 *Slot Capabilities 2 Register (Offset 34h)*

This section is a placeholder. There are no capabilities that require this register.

This register must be treated by software as RsvdP.

7.5.3.22 *Slot Control 2 Register (Offset 38h)*

This section is a placeholder. There are no capabilities that require this register.

This register must be treated by software as RsvdP.

7.5.3.23 *Slot Status 2 Register (Offset 3Ah)*

This section is a placeholder. There are no capabilities that require this register.

This register must be treated by software as RsvdZ.

7.6 PCI Express Extended Capabilities

PCI Express Extended Capability registers are located in Configuration Space at offsets 256 or greater as shown in Figure 7-40 or in the Root Complex Register Block (RCRB). These registers when located in the Configuration Space are accessible using only the PCI Express Enhanced Configuration Access Mechanism (ECAM).

PCI Express Extended Capability structures are allocated using a linked list of optional or required PCI Express Extended Capabilities following a format resembling PCI Capability structures. The first DWORD of the Capability structure identifies the Capability and version and points to the next Capability as shown in Figure 7-40.

Each Capability structure must be DWORD aligned.

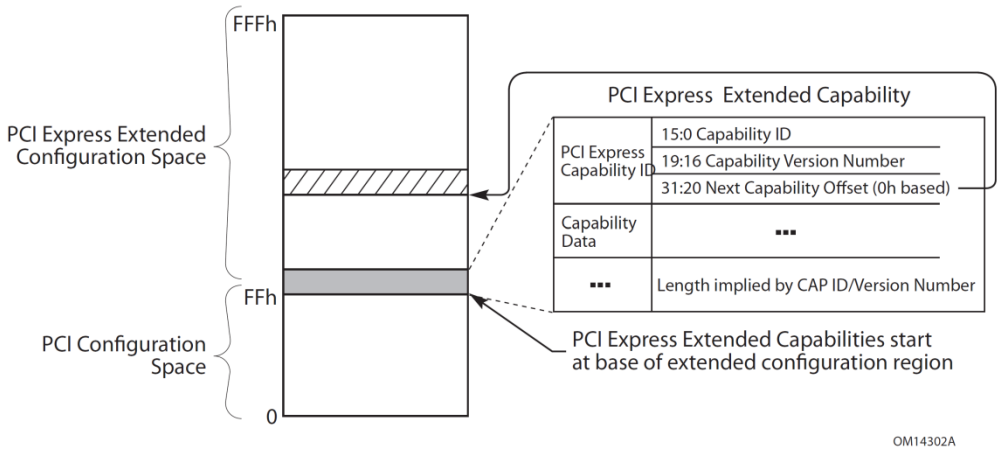


Figure 7-40: PCI Express Extended Configuration Space Layout

7.6.1 Extended Capabilities in Configuration Space

Extended Capabilities in Configuration Space always begin at offset 100h with a PCI Express Extended Capability header (Section 7.6.3). Absence of any Extended Capabilities is required to be indicated by an Extended Capability header with a Capability ID of 0000h, a Capability Version of 0h, and a Next Capability Offset of 000h.

7.6.2 Extended Capabilities in the Root Complex Register Block

Extended Capabilities in a Root Complex Register Block always begin at offset 000h with a PCI Express Extended Capability header (Section 7.6.3). Absence of any Extended Capabilities is required to be indicated by an Extended Capability header with a Capability ID of FFFFh and a Next Capability Offset of 000h.

7.6.3 PCI Express Extended Capability Header

All PCI Express Extended Capabilities must begin with a PCI Express Extended Capability header. Figure 7-41 details the allocation of register fields of a PCI Express Extended Capability header; Table 7-35 provides the respective bit definitions.

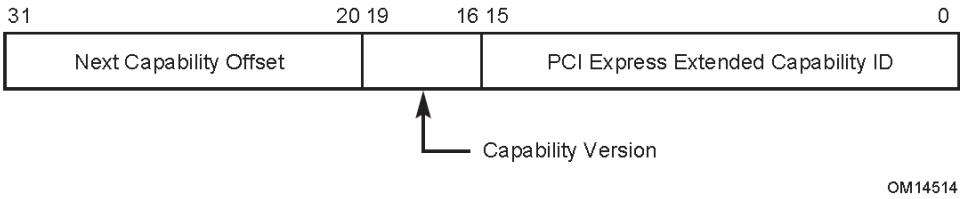


Figure 7-41: PCI Express Extended Capability Header

Table 7-35: PCI Express Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. A version of the specification that changes the Extended Capability in a way that is not otherwise identifiable (e.g., through a new Capability field) is permitted to increment this field. All such changes to the Capability structure must be software-compatible. Software must check for Capability Version numbers that are greater than or equal to the highest number defined when the software is written, as Functions reporting any such Capability Version numbers will contain a Capability structure that is compatible with that piece of software.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh. The bottom 2 bits of this offset are Reserved and must be implemented as 00b although software must mask them to allow for future uses of these bits.	RO

7.7 PCI and PCIe Capabilities Required by the Base Spec in Some Situations

The following capabilities are required by this specification for some Functions. For example, Functions that support specific data rates, functions that generate interrupts, etc.

7.7.1 MSI Capability Structures

All PCI Express device Functions that are capable of generating interrupts must implement MSI or MSI-X or both.

The MSI Capability structure is described in this section. The MSI-X Capability structure is described in Section 7.7.2.

The MSI Capability structure is illustrated in Figure 7-42 and Figure 7-43. Each device Function that supports MSI (in a Multi-Function Device) must implement its own MSI Capability structure. More than one MSI Capability structure per Function is prohibited, but a Function is permitted to have both an MSI and an MSI-X Capability structure.

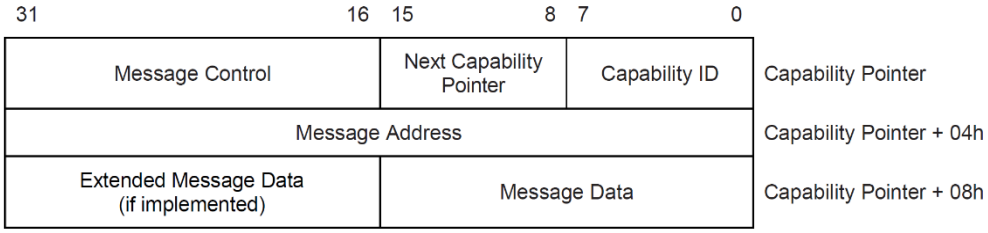


Figure 7-42: MSI Capability Structure for 32-bit Message Address

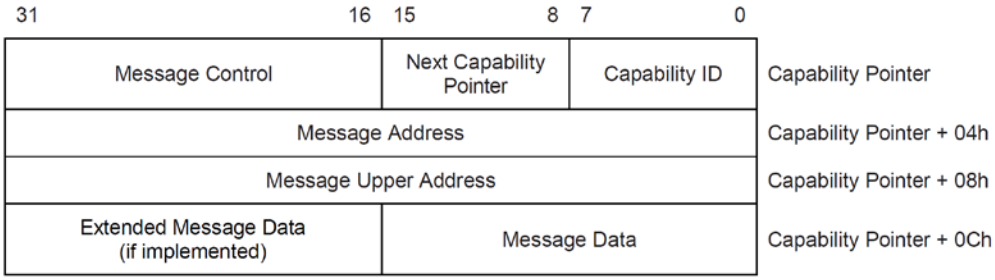


Figure 7-43: MSI Capability Structure for 64-bit Message Address

31 – 16	15 – 8	7 – 0	
Message Control	Next Capability Pointer	Capability ID	Capability Pointer
Message Address			Capability Pointer + 04h
Extended Message Data (or Reserved)	Message Data		Capability Pointer + 08h
Mask Bits			Capability Pointer + 0Ch
Pending Bits			Capability Pointer + 10h

Figure 7-44: MSI Capability Structure for 32-bit Message Address and PVM

31 – 16	15 – 8	7 – 0	
Message Control	Next Capability Pointer	Capability ID	Capability Pointer
Message Address			Capability Pointer + 04h
Message Upper Address			Capability Pointer + 08h
Extended Message Data (or Reserved)	Message Data		Capability Pointer + 0Ch
Mask Bits			Capability Pointer + 10h
Pending Bits			Capability Pointer + 14h

Figure 7-45: MSI Capability Structure for 64-bit Message Address and PVM

To request service, an MSI Function writes the contents of the Message Data register, and if enabled, the Extended Message Data register, to the address specified by the contents of the Message Address register (and, optionally, the Message Upper Address register for a 64-bit message address). A read of the address specified by the contents of the Message Address register produces undefined results.

A Function supporting MSI implements one of four MSI Capability structure layouts illustrated in Figure 7-42 to Figure 7-45, depending upon which optional features are supported. A Legacy Endpoint that implements MSI is required to support either the 32-bit or 64-bit Message Address version of the MSI Capability structure. A PCI Express Endpoint that implements MSI is required to support the 64-bit Message Address version of the MSI Capability structure. The Message Control register indicates the Function's capabilities and provides system software control over MSI.

Each field is further described in the following sections.

7.7.1.1 *MSI Capability Header (Offset 00h)*

The MSI Capability Header enumerates the MSI Capability structure in the PCI Configuration Space Capability list. Figure 7-46 details allocation of register fields in the MSI Capability Header; Table 7-36 provides the respective bit definitions.

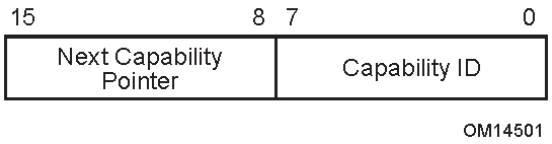


Figure 7-46: MSI Capability Header

Table 7-36: MSI Capability Header

Bit Location	Register Description	Attributes
7:0	**Capability ID** – Indicates the MSI Capability structure. This field must return a Capability ID of 05h indicating that this is an MSI Capability structure.	RO
15:8	**Next Capability Pointer** – This field contains the offset to the next PCI Capability structure or 00h if no other items exist in the linked list of Capabilities.	RO

7.7.1.2 *Message Control Register for MSI (Offset 02h)*

This register provides system software control over MSI. By default, MSI is disabled. If MSI and MSI-X are both disabled, the Function requests servicing using INTx interrupts (if supported). System software can enable MSI by Setting bit 0 of this register. System software is permitted to modify the Message Control register's read-write bits and fields. A device driver is not permitted to modify the Message Control register's read-write bits and fields.

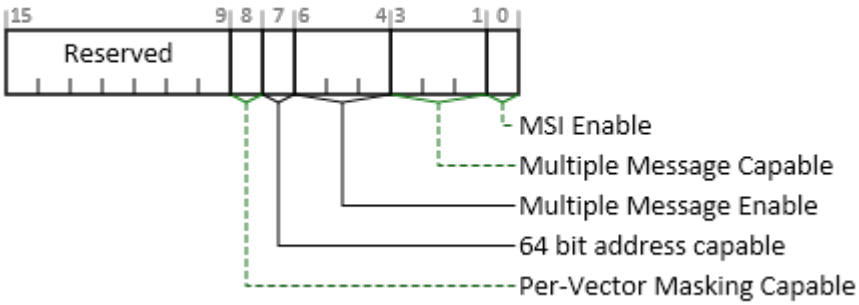


Figure 7-47: Message Control Register for MSI

Table 7-37: Message Control Register for MSI

Bit Location	Register Description	Attributes
0	**MSI Enable** – If Set and the MSI-X Enable bit in the MSI-X Message Control register (see Section 7.9.2) is Clear, the Function is permitted to use MSI to request service and is prohibited from using INTx interrupts. System configuration software Sets this bit to enable MSI. A device driver is prohibited from writing this bit to mask a Function's service request. Refer to Section 7.5.1.1 for control of INTx interrupts. If Clear, the Function is prohibited from using MSI to request service. Default value of this bit is 0b.	RW
3:1	**Multiple Message Capable** – System software reads this field to determine the number of requested vectors. The number of requested vectors must be aligned to a power of two (if a Function requires three vectors, it requests four by initializing this field to 010b). The encoding is defined as: **Encoding** — **# of vectors requested** 000b — 1 001b — 2 010b — 4 011b — 8 100b — 16 101b — 32 110b — Reserved 111b — Reserved	RO

Bit Location	Register Description	Attributes
6:4	**Multiple Message Enable** – software writes to this field to indicate the number of allocated vectors (equal to or less than the number of requested vectors). The number of allocated vectors is aligned to a power of two. If a Function requests four vectors (indicated by a Multiple Message Capable encoding of 010b), system software can allocate either four, two, or one vector by writing a 010b, 001b, or 000b to this field, respectively. When MSI is enabled, a Function will be allocated at least 1 vector. The encoding is defined as: **Encoding** — **# of vectors allocated** 000b — 1 001b — 2 010b — 4 011b — 8 100b — 16 101b — 32 110b — Reserved 111b — Reserved Default value of this field is 000b.	RW
7	**64 bit address capable** – If Set, the Function is capable of sending a 64-bit message address. If Clear, the Function is not capable of sending a 64-bit message address. This bit must be Set if the Function is a PCI Express Endpoint.	RO
8	**Per-Vector Masking Capable** – If Set, the Function supports MSI Per-Vector Masking. If Clear, the Function does not support MSI Per-Vector Masking. This bit must be Set if the Function is a PF or VF within an SR-IOV Device.	RO
9	**Extended Message Data Capable** – If Set, the Function is capable of providing Extended Message Data. If Clear, the Function does not support providing Extended Message Data.	RO
10	**Extended Message Data Enable** – If Set, the Function is enabled to provide Extended Message Data. If Clear, the Function is not enabled to provide Extended Message Data. Default value of this bit is 0b. This bit must be read-write if the Extended Message Data Capable bit is 1b; otherwise it must be hardwired to 0b.	RW/RO

7.7.1.3 *Message Address Register for MSI (Offset 04h)*

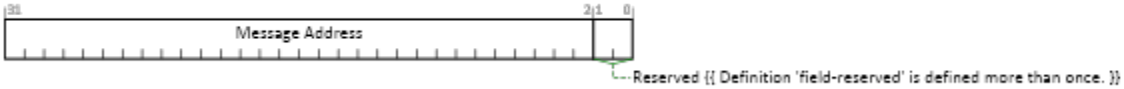


Figure 7-48: Message Address Register for MSI

Table 7-38: Message Address Register for MSI

Bit Location	Register Description	Attributes
1:0	**Reserved** – Always returns 0 on read. Write operations have no effect.	RsvdP
31:2	**Message Address** – System-specified message address. If the Message Enable bit (bit 0 of the Message Control register) is Set, the contents of this register specify the DWORD-aligned address (Address[31:02]) for the MSI transaction. Address[1:0] are set to 00b.	RW

7.7.1.4 *Message Upper Address Register for MSI (Offset 08h)*

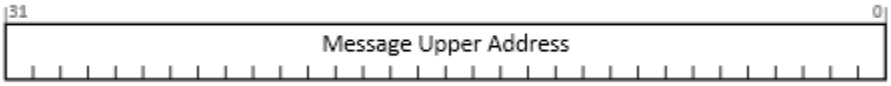


Figure 7-49: Message Upper Address Register for MSI

Table 7-39: Message Upper Address Register for MSI

Bit Location	Register Description	Attributes
31:0	**Message Upper Address** – System-specified message upper address. This register is implemented only if the Function supports a 64-bit message address (bit 7 in Message Control register Set). This register is required for PCI Express Endpoints and is optional for other Function types. If the Message Enable bit (bit 0 of the Message Control register) is Set, the contents of this register (if non-zero) specify the upper 32-bits of a 64-bit message address (Address[63:32]). If the contents of this register are zero, the Function uses the 32 bit address specified by the Message Address register.	RW

7.7.1.5 *Message Data Register for MSI (Offset 08h or 0Ch)*

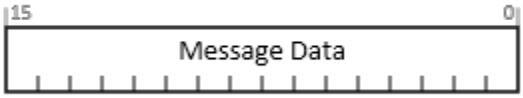


Figure 7-50: Message Data Register for MSI

Table 7-40: Message Data Register for MSI

Bit Location	Register Description	Attributes
15:0	**Message Data** – System-specified message data. If the Message Enable bit (bit 0 of the Message Control register) is Set, the Function sends a DWORD Memory Write transaction using Message Data for the lower 16 bits. All 4 Byte Enables are Set. The Multiple Message Enable field (bits 6-4 of the Message Control register) defines the number of low order message data bits the Function is permitted to modify to generate its system software allocated vectors. For example, a Multiple Message Enable encoding of 010b indicates the Function has been allocated four vectors and is permitted to modify message data bits 1 and 0 (a Function modifies the lower message data bits to generate the allocated number of vectors). If the Multiple Message Enable field is 000b, the Function is not permitted to modify the message data.	RW

7.7.1.6 *Extended Message Data Register for MSI (Optional)*

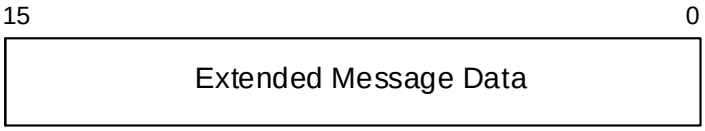


Figure 7-51: Extended Message Data Register for MSI (Optional)

Table 7-41: Extended Message Data Register for MSI

Bit Location	Register Description	Attributes
15:0	**Extended Message Data** – System-specified message data. This register is optional. For the MSI Capability structures without Per-vector Masking, it must be implemented if the Extended Message Data Capable bit is Set; otherwise, it is outside the MSI Capability structure and undefined. For the MSI Capability structures with Per-vector Masking, it must be implemented if the Extended Message Data Capable bit is Set; otherwise, it is RsvdP. If the Extended Message Data Enable bit (bit 10 of the Message Control register) is Set, the DWORD Memory Write transaction uses Extended Message Data for the upper 16 bits; otherwise, it uses 0000h for the upper 16 bits. Default value of this field is 0000h.	RW/undefined/ RsvdP

7.7.1.7 *Mask Bits Register for MSI (Offset 0Ch or 10h*

This register is optional. It is present if Per-Vector Masking Capable is Set (see Section 7.7.1.2). The offset of this register within the capability depends on the value of the 64 bit address capable bit (see Section 7.7.1.2).

The Mask Bits and Pending Bits registers enable software to disable or defer message sending on a per-vector basis.

MSI vectors are numbered 0 through N-1, where N is the number of vectors allocated by software. Each vector is associated with a correspondingly numbered bit in the Mask Bits and Pending Bits registers.

The Multiple Message Capable field indicates how many vectors (with associated Mask and Pending bits) are implemented. All unimplemented Mask and Pending bits are Reserved.

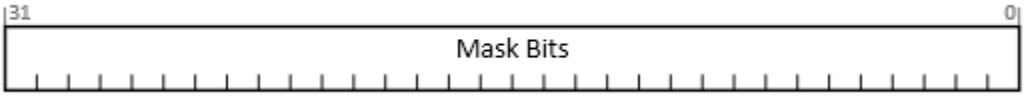


Figure 7-52: Mask Bits Register for MSI

Table 7-42: Mask Bits Register for MSI

Bit Location	Register Description	Attributes
31:0	**Mask Bits** – For each Mask bit that is Set, the Function is prohibited from sending the associated message. Default is 0.	RW

7.7.1.8 *Pending Bits Register for MSI (Offset 10h or 14h)*

This register is optional. It is present if Per-Vector Masking Capable is Set (see Section 7.7.1.2).

The offset of this register within the capability depends on the value of the 64 bit address capable bit (see Section 7.7.1.2)

See Section 7.7.1.7 for additional requirements on this register.

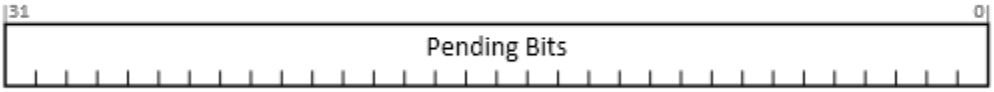


Figure 7-53: Pending Bits Register for MSI

Table 7-43: Pending Bits Register for MSI

Bit Location	Register Description	Attributes
31:0	**Pending Bits** – For each Pending bit that is Set, the Function has a pending associated message. Default is 0.	RO

7.7.2 MSI-X Capability and Table Structure

The MSI-X Capability structure is illustrated in Figure 7-54. More than one MSI-X Capability structure per Function is prohibited, but a Function is permitted to have both an MSI and an MSI-X Capability structure.

In contrast to the MSI Capability structure, which directly contains all of the control/status information for the Function's vectors, the MSI-X Capability structure instead points to an MSI-X Table structure and an MSI-X Pending Bit Array (PBA) structure, each residing in Memory Space.

Each structure is mapped by a Base Address register (BAR) belonging to the Function, located beginning at 10h in Configuration Space, or entry in the Enhanced Allocation capability. A BAR Indicator register (BIR) indicates which BAR(or BEI when using Enhanced Allocation), and a QWORD-aligned Offset indicates where the structure begins relative to the base address associated with the BAR. The BAR is permitted to be either 32-bit or 64-bit, but must map Memory Space. A Function is permitted to map both structures with the same BAR, or to map each structure with a different BAR.

The MSI-X Table structure, illustrated in Figure 7-55, typically contains multiple entries, each consisting of several fields: Message Address, Message Upper Address, Message Data, and Vector Control. Each entry is capable of specifying a unique vector.

The Pending Bit Array (PBA) structure, illustrated in Figure 7-56, contains the Function's Pending Bits, one per Table entry, organized as a packed array of bits within QWORDs. The last QWORD will not necessarily be fully populated.

31 – 16	15 – 8	7 – 3	2 – 0	
Message Control	Next Capability Pointer	Capability ID		CP + 00h
Table Offset			Table BIR	CP + 04h
PBA Offset			PBA BIR	CP + 08h

Figure 7-54: MSI-X Capability Structure

DWORD 3	DWORD 2	DWORD 1	DWORD 0		
Vector Control	Msg Data	Msg Upper Addr	Msg Addr	entry 0	Base
Vector Control	Msg Data	Msg Upper Addr	Msg Addr	entry 1	Base + 1*16
Vector Control	Msg Data	Msg Upper Addr	Msg Addr	entry 2	Base + 2*16
...	...	...	...	...	...
Vector Control	Msg Data	Msg Upper Addr	Msg Addr	entry (N-1)	Base + (N-1)*16

Figure 7-55: MSI-X Table Structure

63 – 0		
Pending Bits 0 through 63	QWORD 0	Base
Pending Bits 64 through 127	QWORD 1	Base + 1*8
...	...	...
Pending Bits ((N-1) div 64)*64 through N-1	QWORD ((N-1) div 64)	Base + ((N-1) div 64)*8

Figure 7-56: MSI-X PBA Structure

To request service using a given MSI-X Table entry, a Function performs a DWORD Memory Write transaction using the contents of the Message Data field entry for data, the contents of the Message Upper Address field for the upper 32 bits of address, and the contents of the Message Address field entry for the lower 32 bits of address. A memory read transaction from the address targeted by the MSI-X message produces undefined results.

If a Base Address register or entry in the Enhanced Allocation capability that maps address space for the MSI-X Table or MSI-X PBA also maps other usable address space that is not associated with MSI-X structures, locations (e.g., for CSRs) used in the other address space must not share any naturally aligned 4-KB address range with one where either MSI-X structure resides. This allows system software where applicable to use different processor attributes for MSI-X structures and the other address space. (Some processor architectures do not support having different processor attributes associated with the same naturally aligned 4-KB physical address range.) The MSI-X Table and MSI-X PBA are permitted to co-reside within a naturally aligned 4-KB address range, though they must not overlap with each other.

IMPLEMENTATION NOTE

Dedicated BARs and Address Range Isolation

To enable system software to map MSI-X structures onto different processor pages for improved access control, it is recommended that a Function dedicate separate Base Address registers for the MSI-X Table and MSI-X PBA, or else provide more than the minimum required isolation with address ranges.

If dedicated separate Base Address registers is not feasible, it is recommended that a Function dedicate a single Base Address register for the MSI-X Table and MSI-X PBA.

If a dedicated Base Address register is not feasible, it is recommended that a Function isolate the MSI-X structures from the non-MSI-X structures with aligned 8 KB ranges rather than the mandatory aligned 4 KB ranges.

For example, if a Base Address register needs to map 2 KB for an MSI-X Table containing 128 entries, 16 bytes for an MSI-X PBA containing 128 bits, and 64 bytes for registers not related to MSI-X, the following is an acceptable implementation. The Base Address register requests 8 KB of total address space, maps the first 64 bytes for the non MSI-X registers, maps the MSI-X Table beginning at an offset of 4 KB, and maps the MSI-X PBA beginning at an offset of 6 KB.

A preferable implementation for a shared Base Address register is for it to request 16 KB of total address space, map the first 64 bytes for the non MSI-X registers, map the MSI-X Table beginning at an offset of 8 KB, and map the MSI-X PBA beginning at an offset of 12 KB.

IMPLEMENTATION NOTE

MSI-X Memory Space Structures in Read/Write Memory

The MSI-X Table and MSI-X PBA structures are defined such that they can reside in general purpose read/write memory on a device, for ease of implementation and added flexibility. To achieve this, none of the contained fields are required to be read-only, and there are also restrictions on transaction alignment and sizes.

For all accesses to MSI-X Table and MSI-X PBA fields, software must use aligned full DWORD or aligned full QWORD transactions; otherwise, the result is undefined.

MSI-X Table entries and Pending bits are each numbered 0 through N-1, where N-1 is indicated by the Table Size field in the MSI-X Message Control register. For a given arbitrary MSI-X Table entry K, its starting address can be calculated with the formula:

entry starting address = Table base + K*16

For the associated Pending bit K, its address for QWORD access and bit number within that QWORD can be calculated with the formulas:

QWORD address = PBA base + (K div[128] 64)*8

QWORD bit# = K mod[129] 64

Software that chooses to read Pending bit K with DWORD accesses can use these formulas:

DWORD address = PBA base + (K div 32)*4

DWORD bit# = K mod 32

Each field in the MSI-X Capability, Table, and PBA structures is further described in the following sections. Within the MSI-X Capability structure, Reserved registers and bits always return 0 when read, and write operations have no effect. Within the MSI-X Table and PBA structures, Reserved fields have special rules.

7.7.2.1 *MSI-X Capability Header (Offset 00h)*

The MSI-X Capability Header enumerates the MSI-X Capability structure in the PCI Configuration Space Capability list. Figure 7-57 details allocation of register fields in the MSI-X Capability Header; Table 7-44 provides the respective bit definitions.

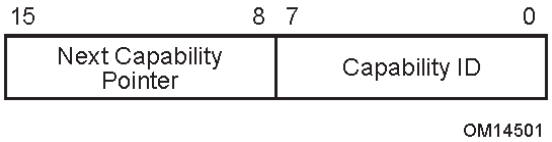


Figure 7-57: MSI-X CapabFPBility Header

Table 7-44: MSI-X Capability Header

Bit Location	Register Description	Attributes
7:0	**Capability ID** – Indicates the MSI-X Capability structure. This field must return a Capability ID of 11h indicating that this is an MSI-X Capability structure.	RO
15:8	**Next Capability Pointer** – This field contains the offset to the next PCI Capability structure or 00h if no other items exist in the linked list of Capabilities.	RO

7.7.2.2 *Message Control Register for MSI-X (Offset 04h)*

By default, MSI-X is disabled. If MSI and MSI-X are both disabled, the Function requests servicing via INTx interrupts (if supported). System software can enable MSI-X by Setting bit 15 of this register. System software is permitted to modify the Message Control register's read-write bits and fields. A device driver is not permitted to modify the Message Control register's read-write bits and fields.

[128] Div is an integer divide with truncation.

[129] Mod is the remainder from an integer divide.

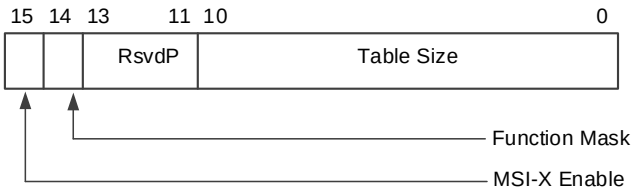


Figure 7-58: Message Control Register for MSI-X

Table 7-45: Message Control Register for MSI-X

Bit Location	Register Description	Attributes
10:0	**Table Size** – System software reads this field to determine the MSI-X Table Size N, which is encoded as N-1. For example, a returned value of 000 0000 0011b indicates a table size of 4.	RO
13:11	**Reserved** – Always returns 0 on a read, and a write operation has no effect.	RsvdP
14	**Function Mask** – If Set, all of the vectors associated with the Function are masked, regardless of their per-vector Mask bit values. If Clear, each vector's Mask bit determines whether the vector is masked or not. Setting or Clearing the MSI-X Function Mask bit has no effect on the value of the per-vector Mask bits. Default value of this bit is 0b.	RW
15	**MSI-X Enable** – If Set and the MSI Enable bit in the MSI Message Control register (see Section 6.8.1.3) is Clear, the Function is permitted to use MSI-X to request service and is prohibited from using INTx interrupts (if implemented). System configuration software Sets this bit to enable MSI-X. A device driver is prohibited from writing this bit to mask a Function's service request. If Clear, the Function is prohibited from using MSI-X to request service. Default value of this bit is 0b.	RW

7.7.2.3 *Table Offset/Table BIR Register for MSI-X (Offset 04h)*

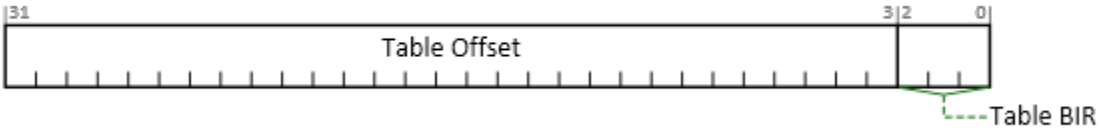


Figure 7-59: Table Offset/Table BIR Register for MSI-X

Table 7-46: Table Offset/Table BIR Register for MSI-X

Bit Location	Register Description	Attributes
2:0	**Table BIR** – Indicates which one of a Function's Base Address registers, located beginning at 10h in Configuration Space, or entry in the Enhanced Allocation capability with a matching BEI, is used to map the Function's MSI-X Table into Memory Space. **BIR Value** — **Base Address register** 0 — 10h 1 — 14h 2 — 18h 3 — 1Ch 4 — 20h 5 — 24h 6 — Reserved 7 — Reserved For a 64-bit Base Address register, the Table BIR indicates the lower DWORD. For Functions with Type 1 Configuration Space headers, BIR values 2 through 5 are also Reserved.	RO
31:3	**Table Offset** – Used as an offset from the address contained by one of the Function's Base Address registers to point to the base of the MSI-X Table. The lower 3 Table BIR bits are masked off (set to zero) by software to form a 32-bit QWORD-aligned offset.	RO

7.7.2.4 *PBA Offset/PBA BIR Register for MSI-X (Offset 08h)*

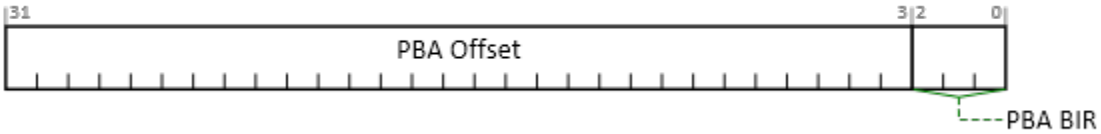


Figure 7-60: PBA Offset/PBA BIR Register for MSI-X

Table 7-47: PBA Offset/PBA BIR Register for MSI-X

Bit Location	Register Description	Attributes
2:0	**PBA BIR** – Indicates which one of a Function's Base Address registers, located beginning at 10h in Configuration Space, or entry in the Enhanced Allocation capability with a matching BEI, is used to map the Function's MSI-X PBA into Memory Space. The PBA BIR value definitions are identical to those for the MSI-X Table BIR.	RO
31:3	**PBA Offset** – Used as an offset from the address contained by one of the Function's Base Address registers to point to the base of the MSI-X PBA. The lower 3 PBA BIR bits are masked off (set to zero) by software to form a 32-bit QWORD-aligned offset.	RO

7.7.2.5 *Message Address Register for MSI-X Table Entries*

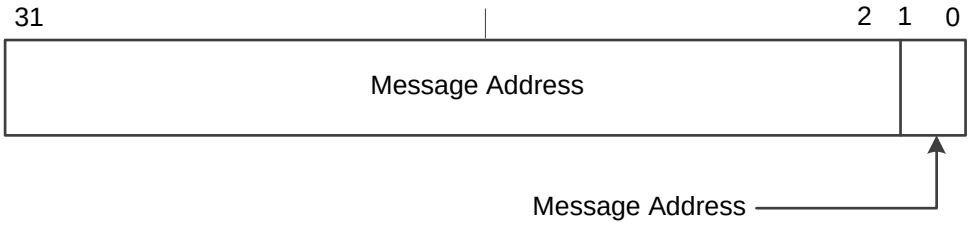


Figure 7-61: Message Address Register for MSI-X Table Entries

Table 7-48: Message Address Register for MSI-X Table Entries

Bit Location	Register Description	Attributes
1:0	**Message Address** – For proper DWORD alignment, software must always write zeroes to these two bits; otherwise the result is undefined. Default value of this field is 00b. These bits are permitted to be read-only or read-write.	RO or RW
31:2	**Message Address** – System-specified message lower address. For MSI-X messages, the contents of this field from an MSI-X Table entry specifies the lower portion of the DWORD-aligned address for the Memory Write transaction.	RW

7.7.2.6 Message Upper Address Register for MSI-X Table Entries

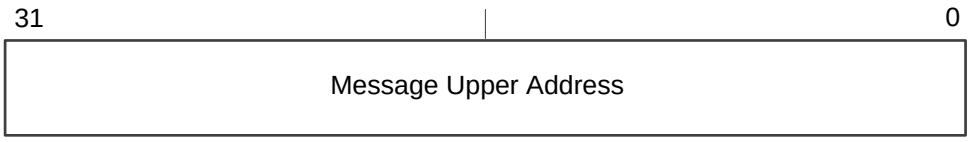


Figure 7-62: Message Upper Address Register for MSI-X Table Entries

Table 7-49: Message Upper Address Register for MSI-X Table Entries

Bit Location	Register Description	Attributes
31:0	**Message Upper Address** – System-specified message upper address bits. If this field is zero, 32-bit address messages are used. If this field is non-zero, 64-bit address messages are used.	RW

7.7.2.7 Message Data Register for MSI-X Table Entries

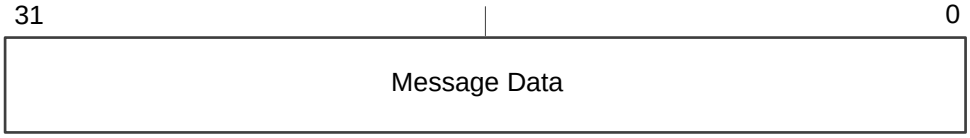


Figure 7-63: Message Data Register for MSI-X Table Entries

Table 7-50: Message Data Register for MSI-X Table Entries

Bit Location	Register Description	Attributes
31:0	**Message Data** – System-specified message data. For MSI-X messages, the contents of this field from an MSI-X Table entry specifies the 32-bit data payload of the DWORD Memory Write transaction. All 4 Byte Enables are Set. In contrast to message data used for MSI messages, the low-order message data bits in MSI-X messages are not modified by the Function. This field is read-write.	RW

7.7.2.8 Vector Control Register for MSI-X Table Entries

If a Function implements a TPH Requester Capability structure and an MSI-X Capability structure, the Function can optionally use the Vector Control register in each MSI-X Table Entry to store a Steering Tag. See Section 6.17.

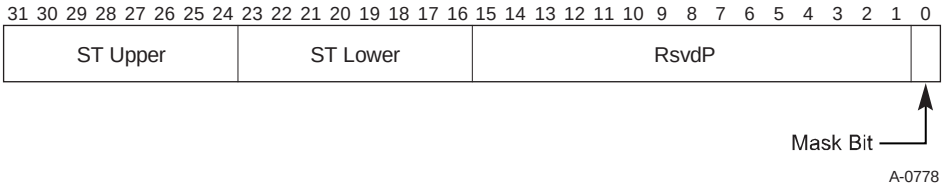


Figure 7-64: Vector Control Register for MSI-X Table Entries

Table 7-51: Vector Control Register for MSI-X Table Entries

Bit Location	Register Description	Attributes
0	**Mask Bit** – When this bit is Set, the Function is prohibited from sending a message using this MSI-X Table entry. However, any other MSI-X Table entries programmed with the same vector will still be capable of sending an equivalent message unless they are also masked. Default value of this bit is 1b (entry is masked)	RW
15:1	**Reserved** – By default, the value of these bits must be 0. However, for potential future use, software must preserve the value of these Reserved bits when modifying the value of other Vector Control bits. If software modifies the value of these Reserved bits, the result is undefined. These bits are permitted to be RsvdP or read-write.	RW or RsvdP
23:16	**ST Lower** – If the Function implements a TPH Requester Capability structure, and the ST Table Location indicates a value of 10b, then this field contains the lower 8 bits of a Steering Tag and must be read-write. Otherwise, this field is permitted to be read-write or RsvdP, and for potential future use, software must preserve the value of these Reserved bits when modifying the value of other Vector Control bits, or the result is undefined. Default value of this field is 00h.	RW/RsvdP
31:24	**ST Upper** – If the Function implements a TPH Requester Capability structure, and the ST Table Location indicates a value of 10b, and the Extended TPH Requester Supported bit is Set, then this field contains the upper 8 bits of a Steering Tag and must be read-write. Otherwise, this field is permitted to be read-write or RsvdP, and for potential future use, software must preserve the value of these Reserved bits when modifying the value of other Vector Control bits, or the result is undefined. Default value of this field is 00h.	RW/RsvdP

7.7.2.9 *Pending Bits Register for MSI-X PBA Entries*

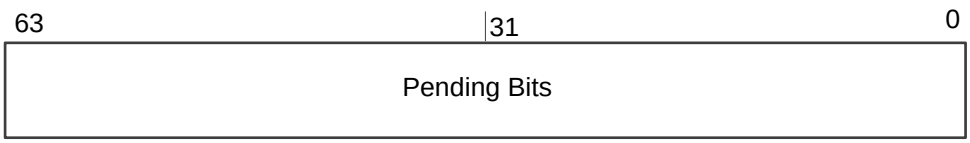


Figure 7-65: Pending Bits Register for MSI-X PBA Entries

Table 7-52: Pending Bits Register for MSI-X PBA Entries

Bit Location	Register Description	Attributes
63:0	**Pending Bits** – For each Pending Bit that is Set, the Function has a pending message for the associated MSI-X Table entry. Pending bits that have no associated MSI-X Table entry are Reserved. By default, the value of Reserved Pending bits must be 0b. Software should never write, and should only read Pending Bits. If software writes to Pending Bits, the result is undefined. Default value of each Pending Bit is 0b. These bits are permitted to be read-only or read-write.	RO or RW

7.7.3 Secondary PCI Express Extended Capability

The Secondary PCI Express Extended Capability structure must be implemented in any Function or RCRB where any of the following are true:

- The Supported Link Speeds Vector field indicates that the Link supports Link Speeds of 8.0 GT/s or higher (see Section 7.5.3.18 or 7.9.9.2).
- Any bit in the Lower SKP OS Generation Supported Speeds Vector field is Set (see Section 7.5.3.18).
- When Lane based errors are reported in the Lane Error Status register (discussed in Section 4.2.6).

To support future additions to this capability, this capability is permitted in any Function or RCRB associated with a Link. For a Multi-Function Device associated with an Upstream Port, this capability is permitted only in Function 0 of the Device.

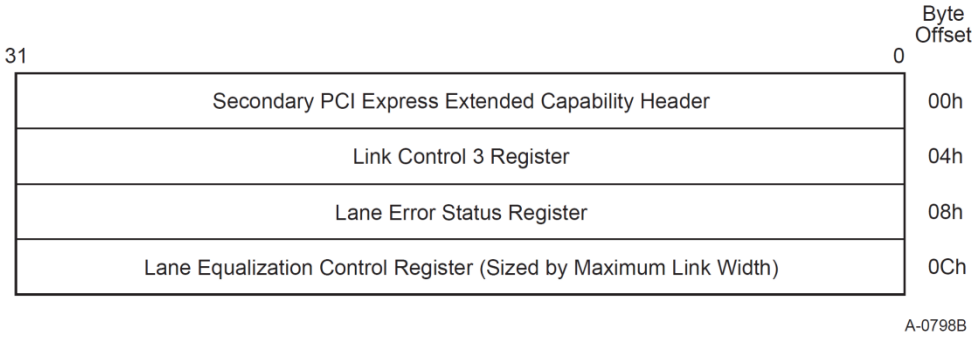


Figure 7-66: Secondary PCI Express Extended Capability Structure

7.7.3.1 Secondary PCI Express Extended Capability Header (Offset 00h)

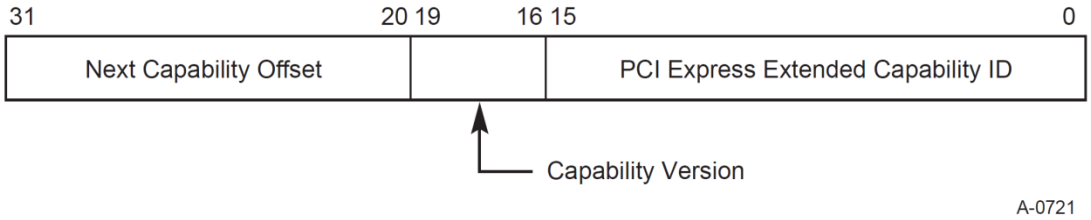


Figure 7-67: Secondary PCI Express Extended Capability Header

Table 7-53: Secondary PCI Express Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. PCI Express Extended Capability ID for the Secondary PCI Express Extended Capability is 0019h**.**	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express Extended Capability structure or 000h if no other items exist in the linked list of Capabilities.	RO

7.7.3.2 Link Control 3 Register (Offset 04h)

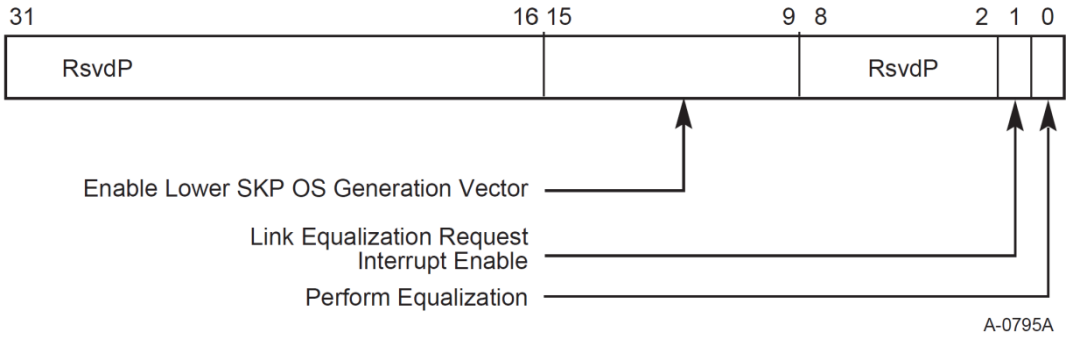


Figure 7-68: Link Control 3 Register

Table 7-54: Link Control 3 Register

Bit Location	Register Description	Attributes
0	**Perform Equalization** – When this bit is 1b and a 1b is written to the Retrain Link bit with the Target Link Speed field set to 8.0 GT/s or higher, the Downstream Port must perform Link Equalization. Refer to Sections 4.2.3 and 4.2.6.4.2 for details. This bit is RW for Downstream Ports and for Upstream Ports when Crosslink Supported is 1b (see Section 7.5.3.18). This bit is not applicable and is RsvdP for Upstream Ports when the Crosslink Supported bit is 0b. The default value is 0b. If the Port does not support 8.0 GT/s, this bit is permitted to be hardwired to 0b.	RW/RsvdP
1	**Link Equalization Request Interrupt Enable** – When Set, this bit enables the generation of an interrupt to indicate that the Link Equalization 8.0 GT/s Request bit or the Link Equalization Request 16.0 GT/s bit has been set. This bit is RW for Downstream Ports and for Upstream Ports when Crosslink Supported is 1b (see Section 7.5.3.18). This bit is not applicable and is RsvdP for Upstream Ports when the Crosslink Supported bit is 0b. The default value for this bit is 0b. If the Port does not support 8.0 GT/s, this bit is permitted to be hardwired to 0b.	RW/RsvdP
9:15	**Enable Lower SKP OS Generation Vector** – When the Link is in L0 and the bit in this field corresponding to the current Link speed is Set, SKP Ordered Sets are scheduled at the rate defined for SRNS, overriding the rate required based on the clock tolerance architecture. See Section 4.2.7 for additional requirements. Bit definitions within this field are: Bit 0 2.5 GT/s Bit 1 5.0 GT/s Bit 2 8.0 GT/s Bit 3 16.0 GT/s Bits 6:4 RsvdP Each unreserved bit in this field must be RW if the corresponding bit in the Lower SKP OS Generation Supported Speeds Vector is Set, otherwise the bit must be RW or hardwired to 0. Behavior is undefined if a bit is Set in this field and the corresponding bit in the Lower SKP OS Generation Supported Speeds Vector is not Set. The default value of this field is 000 0000b.	RW/RsvdP

7.7.3.3 *Lane Error Status Register (Offset 08h)*

The Lane Error Status register consists of a 32-bit vector, where each bit indicates if the Lane with the corresponding Lane number detected an error. This Lane number is the default Lane number which is invariant to Link width and Lane reversal negotiation that occurs during Link training.

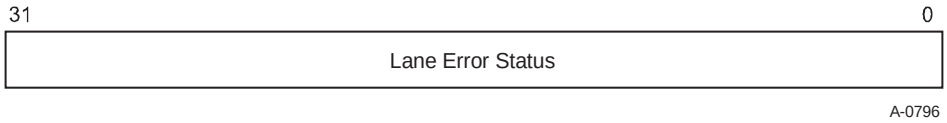


Figure 7-69: Lane Error Status Register

Table 7-55: Lane Error Status Register

Bit Location	Register Description	Attributes
(Maximum Link Width -1):0	**Lane Error Status Bits** – Each bit indicates if the corresponding Lane detected a Lane-based error. A value of 1b indicates that a Lane based-error was detected on the corresponding Lane Number (see Sections 4.2.2.3.3, 4.2.6, and 4.2.7.2 for details). The default value of each bit is 0b. For Ports that are narrower than 32 Lanes, the unused upper bits [31: Maximum Link Width] are RsvdZ. For Ports that do not support 8.0 GT/s and do not set these bits based on 8b/10b errors (optional, see Section 4.2.6), this field is permitted to be hardwired to 0.	RW1CS

7.7.3.4 *Lane Equalization Control Register (Offset 0Ch)*

The Lane Equalization Control register consists of control fields required for per-Lane 8.0 GT/s equalization and the number of entries in this register are sized by Maximum Link Width (see Section 7.5.3.6). Each entry contains the values for the Lane with the corresponding default Lane number which is invariant to Link width and Lane reversal negotiation that occurs during Link training.

If the Port does not support 8.0 GT/s, this register is permitted to be hardwired to 0.

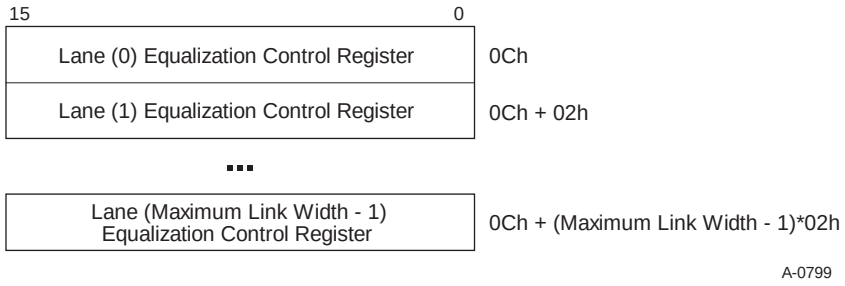


Figure 7-70: Lane Equalization Control Register

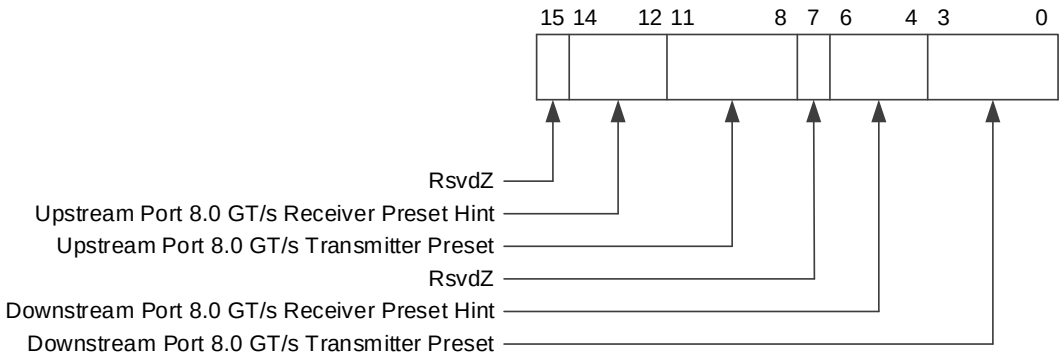


Figure 7-71: Lane Equalization Control Register Entry

Table 7-56: Lane Equalization Control Register Entry

Bit Location	Register Description	Attributes
3:0	**Downstream Port 8.0 GT/s Transmitter Preset** – Transmitter preset value used for 8.0 GT/s equalization by this Port when the Port is operating as a Downstream Port. This field is ignored when the Port is operating as an Upstream Port. See Chapter 8 for details. The field encodings are defined in Section 4.2.3.2. For an Upstream Port if Crosslink Supported is 0b, this field is RsvdP. Otherwise, this field is HwInit. See Section 7.5.3.18. The default value is 1111b.	HwInit/RsvdP (see description)
6:4	**Downstream Port 8.0 GT/s Receiver Preset Hint** – Receiver preset hint value that may be used as a suggested setting for 8.0 GT/s receiver equalization by this Port when the Port is operating as a Downstream Port. This field is ignored when the Port is operating as an Upstream Port. See Chapter 8 for details. The field encodings are defined in Section 4.2.3.2. For an Upstream Port if Crosslink Supported is 0b, this field is RsvdP. Otherwise, this field is HwInit. See Section 7.5.3.18. The default value is 111b.	HwInit/RsvdP (see description)

<table>
<tr><th>Bit Location</th><th>Register Description</th><th>Attributes</th></tr>
<tr><td>11:8</td><td>

Upstream Port 8.0 GT/s Transmitter Preset – Field contains the Transmitter preset value sent or received during 8.0 GT/s Link Equalization. Field usage varies as follows:

<table>
<tr><th></th><th>Operating Port Direction</th><th>Crosslink Supported</th><th>Usage</th></tr>
<tr><td>A</td><td>Downstream Port</td><td>Any</td><td>Field contains the value sent on the associated Lane during Link Equalization.

Field is HwInit.</td></tr>
<tr><td>B</td><td>Upstream Port</td><td>0b</td><td>Field is intended for debug and diagnostics. It contains the value captured from the associated Lane during Link Equalization.

Field is RO.

Note: When crosslinks are supported, case C (below) applies and this captured information is not visible to software. Vendors are encouraged to provide an alternate mechanism to obtain this information.</td></tr>
<tr><td>C</td><td>Upstream Port</td><td>1b</td><td>Field is not used or affected by the current Link Equalization.

Field value will be used if a future crosslink negotiation switches the Operating Port Direction so that case A (above) applies.

Field is HwInit.</td></tr>
</table>

See Sections 4.2.3 and Chapter 8 for details. The field encodings are defined in Section 4.2.3.2.

The default value is 1111b.

</td><td>HwInit/RO (see description)</td></tr>
</table>

<table>
<tr><th>Bit Location</th><th>Register Description</th><th>Attributes</th></tr>
<tr><td>14:12</td><td>

Upstream Port 8.0 GT/s Receiver Preset Hint – Field contains the Receiver preset hint value sent or received during 8.0 GT/s Link Equalization. Field usage varies as follows:

<table>
<tr><th></th><th>Operating Port Direction</th><th>Crosslink Supported</th><th>Usage</th></tr>
<tr><td>A</td><td>Downstream Port</td><td>Any</td><td>Field contains the value sent on the associated Lane during Link Equalization.

Field is HwInit.</td></tr>
<tr><td>B</td><td>Upstream Port</td><td>0b</td><td>Field is intended for debug and diagnostics. It contains the value captured from the associated Lane during Link Equalization.

Field is RO.

Note: When crosslinks are supported, case C (below) applies and this captured information is not visible to software. Vendors are encouraged to provide an alternate mechanism to obtain this information.</td></tr>
<tr><td>C</td><td>Upstream Port</td><td>1b</td><td>Field is not used or affected by the current Link Equalization.

Field value will be used if a future crosslink negotiation switches the Operating Port Direction so that case A (above) applies.

Field is HwInit.</td></tr>
</table>

See Sections 4.2.3 and Chapter 8 for details. The field encodings are defined in Section 4.2.3.2.

The default value is 111b.

</td><td>HwInit/RO (see description)</td></tr>
</table>

7.7.4 Data Link Feature Extended Capability

The Data Link Feature Capability is an optional Extended Capability that is required for Downstream Ports that support one or more of the associated features. Since the Scaled Flow

Control Feature is required for Ports that support 16.0 GT/s, this capability is required for Downstream Ports that support 16/0 GT/s (see Section 3.4.2). It is optional in other Downstream Ports. It is optional in Functions associated with an Upstream Port. In Multi-Function Devices associated with an Upstream Port, all instances of this capability must report identical information in all fields of this capability. It is not applicable in Functions that are not associated with a Port (e.g., RCiEPs, Root Complex Event Collectors). The Data Link Feature Extended Capability is shown in Figure 7-72.

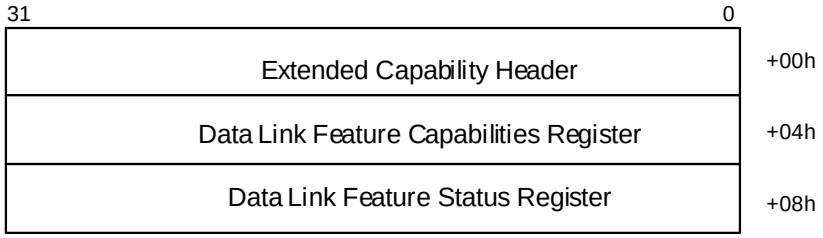


Figure 7-72: Data Link Feature Extended Capability

7.7.4.1 *Data Link Feature Extended Capability Header (Offset 00h)*

Figure 7-73 details allocation of register fields in the Data Link Feature Extended Capability header; Table 7-57 provides the respective bit definitions.

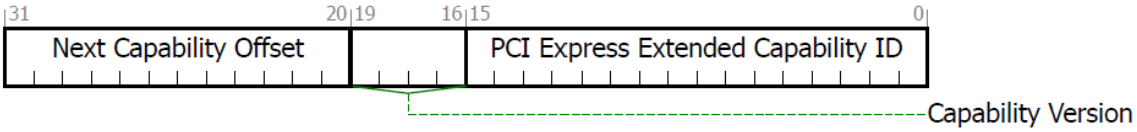


Figure 7-73: Data Link Feature Extended Capability Header

Table 7-57: Data Link Feature Extended Capability Header

Bit Location	Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. Extended Capability ID for Data Link Feature is 0025h	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.	RO

Bit Location	Description	Attributes
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh. The bottom 2 bits of this offset are Reserved and must be implemented as 00b although software must mask them to allow for future uses of these bits.	RO

7.7.4.2 *Data Link Feature Capabilities Register (Offset 04h)*

Figure 7-74details allocation of register fields in the Data Link Feature Capabilities register; Table 7-58 provides the respective bit definitions.

When this Port sends a Data Link Feature DLLP, the Feature Support field in Symbols 1, 2, and 3 of that DLLP contains bits [22:16], [15:8], and [7:0] of this register respectively (See Figure 3-12).

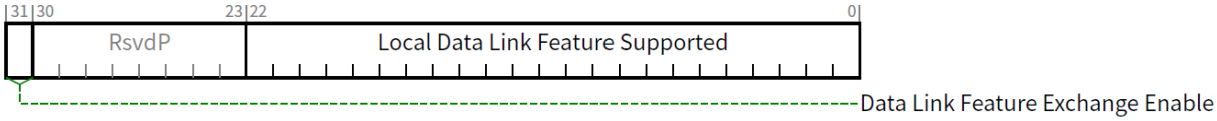


Figure 7-74: Data Link Feature Capabilities Register

Table 7-58: Data Link Feature Capabilities Register

Bit Location	Description	Attributes
22:0	**Local Data Link Feature Supported** – This field contains the Feature Supported value used when this Port sends a Data Link Feature DLLP (see Figure 3-12). Defined features are: Bit 0 – Local Scaled Flow Control Supported This bit indicates that this Port supports the Scaled Flow Control Feature (see Section 3.4.2). Bits 22:1 **RsvdP** Bits associated with features that this Port is capable of supporting are HwInit, defaulting to 1b. Other bits in this field are RsvdP.	HwInit/ RsvdP
31	**Data Link Feature Exchange Enable** – If Set, this bit indicates that this Port will enter the DL_Feature negotiation state (see Section 3.2.1). Default is 1b.	HwInit

7.7.4.3 *Data Link Feature Status Register (Offset 08h)*

Figure 7-75 details allocation of register fields in the Data Link Feature Status register; Table 7-59 provides the respective bit definitions.

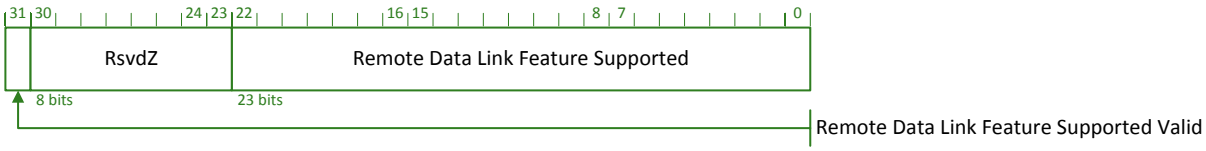


Figure 7-75: Data Link Feature Status Register

Table 7-59: Data Link Feature Status Register

Bit Location	Description	Attributes
22:0	**Remote Data Link Feature Supported** – These bits indicate that the Remote Port supports the corresponding Data Link Feature. These bits capture all information from the Feature Supported field of the Data Link Feature DLLP even when this Port doesn't support the corresponding feature. This field is Cleared on entry to state DL_Inactive (see Section 3.2.1). Features currently defined are: Bit 0 – Remote Scaled Flow Control Supported This bit indicates that the Remote Port supports the Scaled Flow Control Feature (see Section 3.4.2). Bits 22:1 – **Undefined** Default is 00 0000h	RO
31	**Remote Data Link Feature Supported Valid** – This bit indicates that the Port has received a Data Link Feature DLLP in state DL_Feature (see Section 3.2.1) and that the Remote Data Link Feature Supported field is meaningful. This bit is Cleared on entry to state DL_Inactive (see Section 3.2.1). Default is 0b.	RO

7.7.5 Physical Layer 16.0 GT/s Extended Capability

The Physical Layer 16.0 GT/s Extended Capability structure must be implemented in:

- A Function associated with a Downstream Port where the Supported Link Speeds Vector field indicates support for a Link speed of 16.0 GT/s.
- A Function of a single-Function Device associated with an Upstream Port where the Supported Link Speeds Vector field indicates support for a Link speed of 16.0 GT/s.
- Function 0 (and only Function 0) of a Multi-Function Device associated with an Upstream Port where the Supported Link Speeds Vector field indicates support for a Link speed of 16.0 GT/s.

This capability is permitted to be implemented in any of the Functions listed above even if the 16.0 GT/s Link speed is not supported. When the 16.0 GT/s Link speed is not supported, the behavior of registers other than the Capability Header is undefined.

Figure 7-76 details allocation of register fields in the Physical Layer 16.0 GT/s Extended Capability structure.

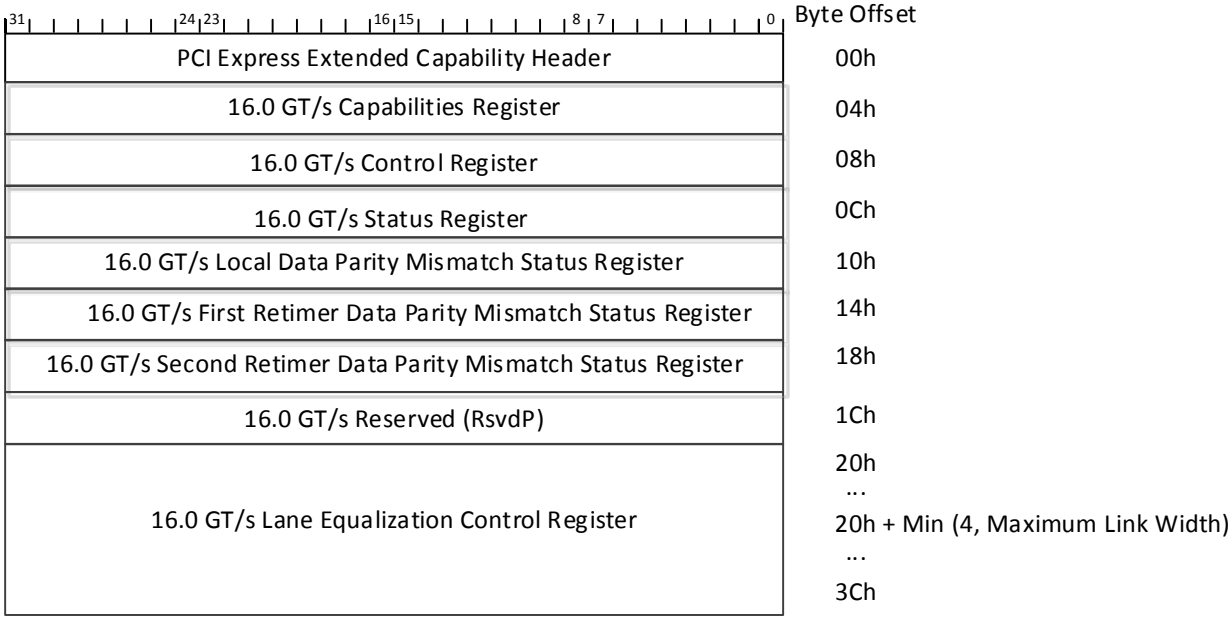


Figure 7-76: Physical Layer 16.0 GT/s Extended Capability

7.7.5.1 *Physical Layer 16.0 GT/s Extended Capability Header (Offset 00h)*

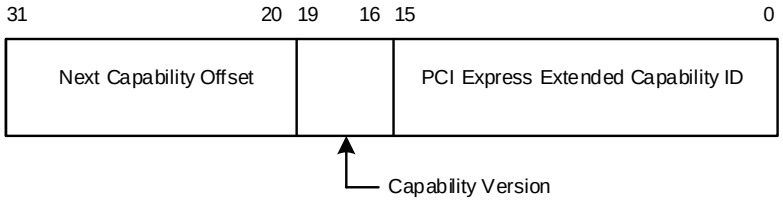


Table 7-60: Physical Layer 16.0 GT/s Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. The Extended Capability ID for the Physical Layer 16.0 GT/s Capability is 0026h.	RO

Bit Location	Register Description	Attributes
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh.	RO

7.7.5.2 *16.0 GT/s Capabilities Register (Offset 04h)*

Table 7-61: 16.0 GT/s Capabilities Register

Bit Location	Register Description	Attributes
31:0	**RsvdP**	RsvdP

7.7.5.3 *16.0 GT/s Control Register (Offset 08h)*

Table 7-62: 16.0 GT/s Control Register

Bit Location	Register Description	Attributes
31:0	**RsvdP**	RsvdP

7.7.5.4 *16.0 GT/s Status Register (Offset 0Ch)*

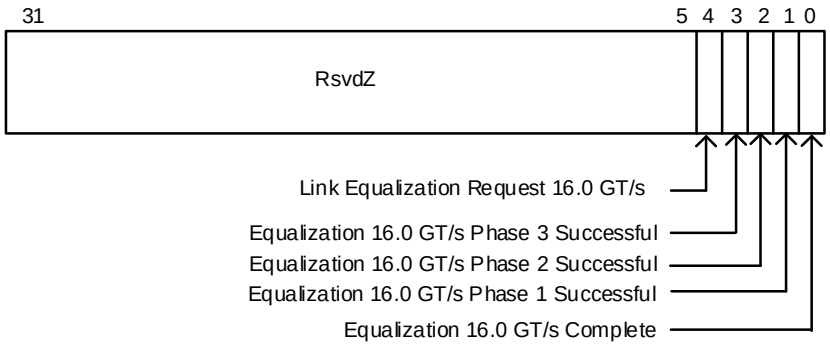


Figure 7-77: 16.0 GT/s Status Register

Table 7-63: 16.0 GT/s Status Register

Bit Location	Register Description	Attributes
0	**Equalization 16.0 GT/s Complete** – When Set, this bit indicates that the 16.0 GT/s Transmitter Equalization procedure has completed. Details of the Transmitter Equalization process and when this bit needs to be set to 1b is provided in Section 4.2.6.4.2. The default value of this bit is 0b. For a Multi-Function Upstream Port, this bit must be implemented in Function 0 and RsvdZ in other Functions.	ROS/ RsvdZ
1	**Equalization 16.0 GT/s Phase 1 Successful** – When set to 1b, this bit indicates that Phase 1 of the 16.0 GT/s Transmitter Equalization procedure has successfully completed. Details of the Transmitter Equalization process and when this bit needs to be set to 1b is provided in Section 4.2.6.4.2. The default value of this bit is 0b. For a Multi-Function Upstream Port, this bit must be implemented in Function 0 and RsvdZ in other Functions.	ROS/ RsvdZ
2	**Equalization 16.0 GT/s Phase 2 Successful** – When set to 1b, this bit indicates that Phase 2 of the 16.0 GT/s Transmitter Equalization procedure has successfully completed. Details of the Transmitter Equalization process and when this bit needs to be set to 1b is provided in Section 4.2.6.4.2. The default value of this bit is 0b. For a Multi-Function Upstream Port, this bit must be implemented in Function 0 and RsvdZ in other Functions.	ROS/ RsvdZ
3	**Equalization 16.0 GT/s Phase 3 Successful** – When set to 1b, this bit indicates that Phase 3 of the 16.0 GT/s Transmitter Equalization procedure has successfully completed. Details of the Transmitter Equalization process and when this bit needs to be set to 1b is provided in Section 4.2.6.4.2. The default value of this bit is 0b. For a Multi-Function Upstream Port, this bit must be implemented in Function 0 and RsvdZ in other Functions.	ROS/ RsvdZ
4	**Link Equalization Request 16.0 GT/s** – This bit is Set by hardware to request the 16.0 GT/s Link equalization process to be performed on the Link. Refer to Sections 4.2.3 and 4.2.6.4.2 for details. The default value of this bit is 0b. For a Multi-Function Upstream Port, this bit must be implemented in Function 0 and RsvdZ in other Functions.	RW1CS/ RsvdZ
31:5	**Reserved**	RsvdZ

7.7.5.5 *16.0 GT/s Local Data Parity Mismatch Status Register (Offset 10h)*

The Local Data Parity Mismatch Status register is a 32-bit vector where each bit indicates if the local receiver detected a Data Parity mismatch on the Lane with the corresponding Lane number. This Lane number is the default Lane number which is invariant to Link width and Lane reversal negotiation that occurs during Link training.

31	0
Local Data Parity Mismatch Status	

Figure 7-78: 16.0 GT/s Local Data Parity Mismatch Status Register

Table 7-64: 16.0 GT/s Local Data Parity Mismatch Status Register

Bit Location	Register Description	Attributes
(Maximum Link Width -1):0	**Local Data Parity Mismatch Status** – Each bit indicates if the corresponding Lane detected a Data Parity mismatch. A value of 1b indicates that a mismatch was detected on the corresponding Lane Number. See Section 4.2.7.2 for more information. The default value of each bit is 0b. For Ports that are narrower than 32 Lanes, the unused upper bits [31: Maximum Link Width] are RsvdZ.	RW1CS/ RsvdZ

7.7.5.6 *16.0 GT/s First Retimer Data Parity Mismatch Status Register (Offset 14h)*

The First Retimer Data Parity Status register is a 32-bit vector where each bit indicates if the first Retimer of a Path (see Figure 4-32 for more information) detected a Data Parity mismatch on the Lane with the corresponding Lane number. This Lane number is the default Lane number which is invariant to Link width and Lane reversal negotiation that occurs during Link training.

31	0
First Retimer Data Parity Mismatch Status	

Figure 7-79: 16.0 GT/s First Retimer Data Parity Mismatch Status Register

Table 7-65: 16.0 GT/s First Retimer Data Parity Mismatch Status Register

Bit Location	Register Description	Attributes
(Maximum Link Width -1):0	**First Retimer Data Parity Mismatch Status** – Each bit indicates if the corresponding Lane detected a Data Parity mismatch. A value of 1b indicates that a mismatch was detected on the corresponding Lane Number. See Section 4.2.7.2 for more information. The default value of each bit is 0b. The value of this field is undefined when no Retimers are present. For Ports that are narrower than 32 Lanes, the unused upper bits [31: Maximum Link Width] are RsvdZ.	RW1CS/ RsvdZ

7.7.5.7 *16.0 GT/s Second Retimer Data Parity Mismatch Status Register (Offset 18h)*

The Second Retimer Data Parity Mismatch Status register is a 32-bit vector where each bit indicates if the second Retimer of a Path (see Figure 4-32 for more information) detected a Data Parity mismatch on the Lane with the corresponding Lane number. This Lane number is the default Lane number which is invariant to Link width and Lane reversal negotiation that occurs during Link training.

31	0
Second Retimer Data Parity Mismatch Status	

Figure 7-80: 16.0 GT/s Second Retimer Data Parity Mismatch Status Register

Table 7-66: 16.0 GT/s Second Retimer Data Parity Mismatch Status Register

Bit Location	Register Description	Attributes
(Maximum Link Width -1):0	**Second Retimer Data Parity Mismatch Status** – Each bit indicates if the corresponding Lane detected a Data Parity mismatch. A value of 1b indicates that a mismatch was detected on the corresponding Lane Number. See Section 4.2.7.2 for more information. The default value of each bit is 0b. The value of this field is undefined when no Retimers are present or only one Retimer is present. For Ports that are narrower than 32 Lanes, the unused upper bits [31: Maximum Link Width] are RsvdZ.	RW1CS/ RsvdZ

7.7.5.8 *Physical Layer 16.0 GT/s Reserved (Offset 1Ch)*

This register is RsvdP.

7.7.5.9 *16.0 GT/s Lane Equalization Control Register (Offset 20h)*

The Equalization Control register consists of control fields required for per-Lane 16.0 GT/s equalization. It contains entries for at least the number of Lanes defined by the Maximum Link Width (see Section 7.5.3.6), must be implemented in whole DW size increments, and it is permitted to contain up to 32 entries regardless of the Maximum Link Width (see Figure 7-26). The value of entries beyond the Maximum Link Width is undefined.

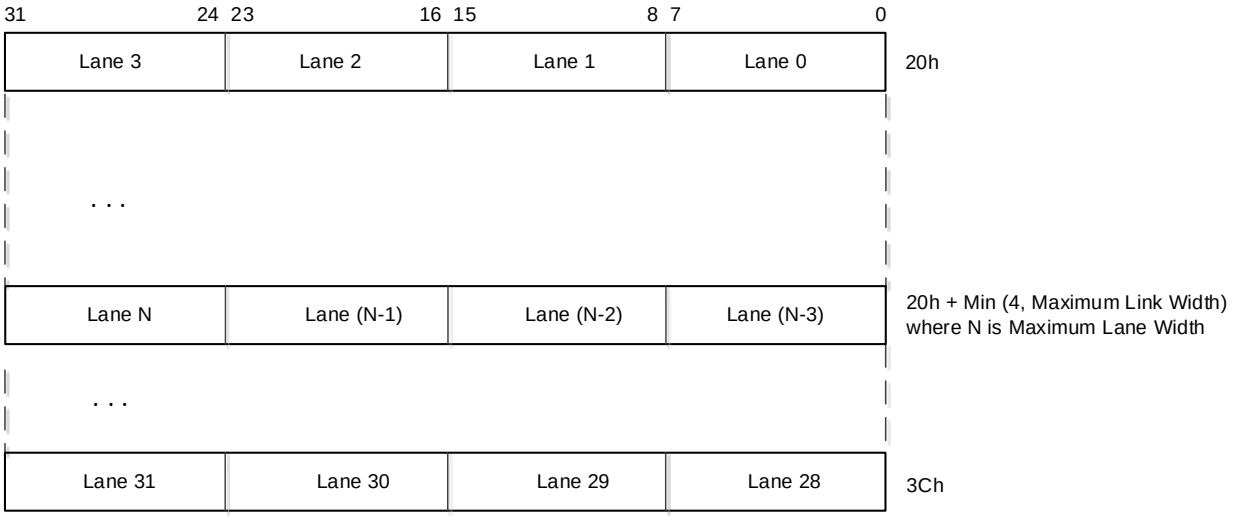


Figure 7-81: High Level Structure of 16.0 GT/s Lane Equalization Control Register

Each entry contains the values for the Lane with the corresponding default Lane number which is invariant to Link width and Lane reversal negotiation that occurs during Link training.

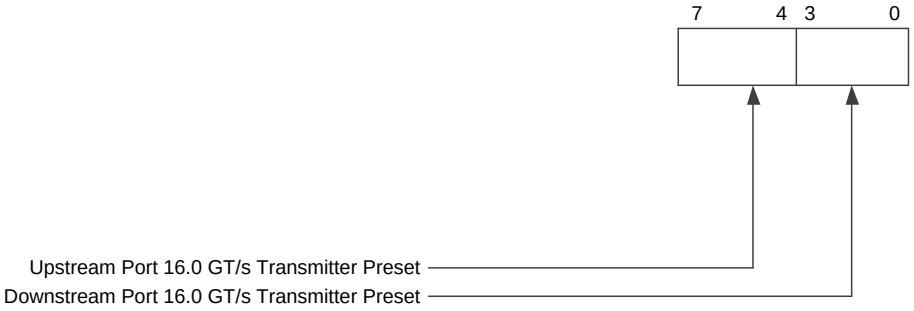


Figure 7-82: 16.0 GT/s Lane ((Maximum Link Width – 1):0) Equalization Control Register Entry

Table 7-67: 16.0 GT/s Lane ((Maximum Link Width – 1):0) Equalization Control Register Entry

Bit Location	Register Description	Attributes
3:0	**Downstream Port 16.0 GT/s Transmitter Preset** – Transmitter Preset used for 16.0 GT/s equalization by this Port when the Port is operating as a Downstream Port. This field is	HwInit/RsvdP (see description)

<table>
<tr><th>Bit Location</th><th>Register Description</th><th>Attributes</th></tr>
<tr><td></td><td>ignored when the Port is operating as an Upstream Port. See Chapter 8 for details. The field encodings are defined in Section 4.2.3.2.

For an Upstream Port if Crosslink Supported is 0b, this field is RsvdP. Otherwise, this field is HwInit. See Section 7.5.3.18.

The default value is 1111b.</td><td></td></tr>
<tr><td>7:4</td><td>Upstream Port 16.0 GT/s Transmitter Preset – Field contains the Transmit Preset value sent or received during 16.0 GT/s Link Equalization. Field usage varies as follows:
<table>
<tr><th></th><th>Operating Port Direction</th><th>Crosslink Supported</th><th>Usage</th></tr>
<tr><td>A</td><td>Downstream Port</td><td>Any</td><td>Field contains the value sent on the associated Lane during Link Equalization.

Field is HwInit.</td></tr>
<tr><td>B</td><td>Upstream Port</td><td>0b</td><td>Field is intended for debug and diagnostics. It contains the value captured from the associated Lane during Link Equalization.

Field is RO.

When crosslinks are supported, case C (below) applies and this captured information is not visible to software. Vendors are encouraged to provide an alternate mechanism to obtain this information.</td></tr>
<tr><td>C</td><td>Upstream Port</td><td>1b</td><td>Field is not used or affected by the current Link Equalization.

Field value will be used if a future crosslink negotiation switches the Operating Port Direction so that case A (above) applies.

Field is HwInit.</td></tr>
</table>
See Sections 4.2.3 and Chapter 8 for details. The field encodings are defined in Section 4.2.3.2.

The default value is 1111b.</td><td>HwInit/RO (see description)</td></tr>
</table>

7.7.6 Lane Margining at the Receiver Extended Capability

The Margining Extended Capability structure must be implemented in:

- A Function associated with a Downstream Port where the Supported Link Speeds Vector field indicates support for a Link speed of 16.0 GT/s or higher.
- A Function of a single-Function Device associated with an Upstream Port where the Supported Link Speeds Vector field indicates support for a Link speed of 16.0 GT/s or higher.
- Function 0 (and only Function 0) of a Multi-Function Device associated with an Upstream Port where the Supported Link Speeds Vector field indicates support for a Link speed of 16.0 GT/s or higher.

Figure 7-83 shows the layout of the Margining Extended Capability. This capability contains a pair of per-Port registers followed by a set of per-Lane registers.

The number of per-Lane entries in is sized by Maximum Link Width (see Section 7.5.3.6). Up to 32 entries are permitted regardless of the Maximum Link Width. The value of entries beyond the Maximum Link Width is undefined.

Each per-Lane entry contains the values for that Lane. Lane numbering uses the default Lane number and is thus invariant to Link width and Lane reversal negotiation that occurs during Link training.

		00h
Margining Port Status	Margining Port Capabilities	04h
Margining Lane Status: Lane 0	Margining Lane Control: Lane 0	08h
Margining Lane Status: Lane 1	Margining Lane Control: Lane 1	0Ch
• • •		
Margining Lane Status: Lane (Maximum Link Width – 1)	Margining Lane Control: Lane (Maximum Link Width – 1)	08h + (Maximum Link Width - 1) * 4
Undefined		
• • •		
Undefined		44h = 08h + (31-1)*4

Figure 7-83: Margining Extended Capability

7.7.6.1 *Margining Extended Capability Header (Offset 00h)*

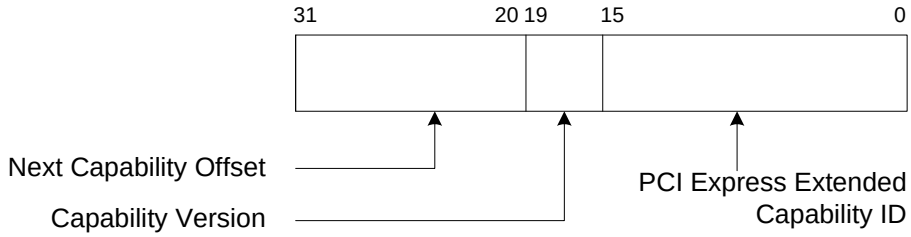


Figure 7-84: Physical Layer 16.0 GT/s Margining Extended Capability Header

Table 7-68: Physical Layer 16.0 GT/s Margining Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. The Extended Capability ID for the Physical Layer 16.0 GT/s Margining Extended Capability is 0027h.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh.	RO

7.7.6.2 *Margining Port Capabilities Register (Offset 04h)*

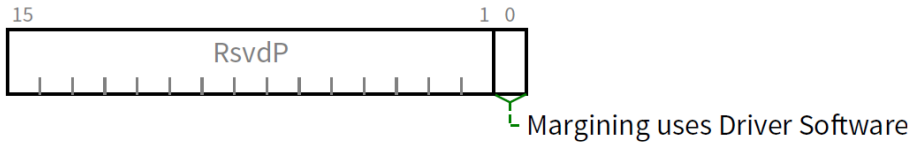


Figure 7-85: Margining Port Capabilities Register

Table 7-69: Margining Port Capabilities Register

Bit Location	Register Description	Attributes
0	**Margining uses Driver Software** – If Set, indicates that Margining is partially implemented using Device Driver software. Margining Software Ready indicates when this software is initialized. If Clear, Margining does not require device driver software. In this case the value read from Margining Software Ready is undefined.	HwInit

7.7.6.3 *Margining Port Status Register (Offset 06h)*

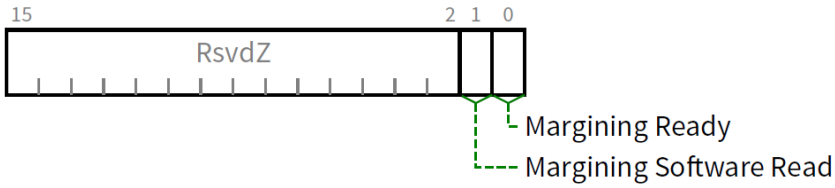


Figure 7-86: Margining Port Status Register

Table 7-70: Margining Port Status Register

Bit Location	Register Description	Attributes
0	**Margining Ready**. – Indicates when the Margining feature is ready to accept margining commands. Behavior is undefined if this bit is Clear and, for any Lane, any of the Receiver Number, Margin Type, Usage Model, or Margin Payload fields are written (see Section 7.7.6.5). If Margining uses Driver Software is Set, Margining Ready must be Set no later than 100 ms after the later of Margining Software Ready becoming Set or the link training to 16.0 GT/s. If Margining uses Driver Software is Clear, Margining Ready must be Set no later than 100 ms after the Link trains to 16.0 GT/s. Default value is implementation specific.	RO
1	**Margining Software Ready** – When Margining uses Driver Software is Set, then this bit, when Set, indicates that the required software has performed the required initialization. The value of this bit is Undefined if Margining users Driver Software is Clear. The default value of this bit is implementation specific.	RO

7.7.6.4 *Margining Lane Control Register (Offset 08h)*

The Margining Lane Control register consists of control fields required for per-Lane margining. The number of entries in this register are sized by Maximum Link Width (see Section 7.5.3.6). See Section 4.2.7.2 for details of this register.

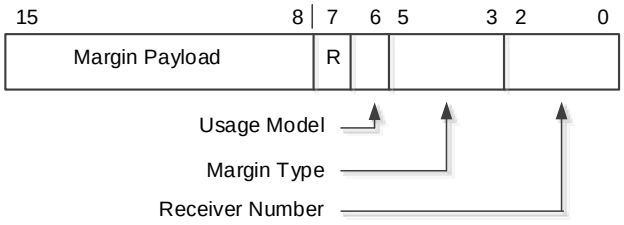


Figure 7-87: Lane N: Margining Lane Control Register Entry

Bit Location	Register Description	Attributes
2:0	**Receiver Number**. – See Section 8.4.4 for details. The default value is 000b. This field must be reset to the default value if the Port goes to DL_Down status.	RW (see description)
5:3	**Margin Type** –See Section 8.4.4 for details. The default value is 111b. This field must be reset to the default value if the Port goes to DL_Down status.	RW (see description)
6	**Usage Model** – See Section 8.4.4 for details. The default value is 0b. This field must be reset to the default value if the Port goes to DL_Down status.	RW (see description)
15:8	**Margin Payload** – See Section 8.4.4 for details. This field's value is used in conjunction with the Margin Type field, as described in Section 8.4.4. The default value is 9Ch. This field must be reset to the default value if the Port goes to DL_Down status.	RW (see description)

7.7.6.5 *Margining Lane Status Register (Offset 0Ah)*

The Margining Lane Status register consists of status fields required for per-Lane margining. The number of entries in this register are sized by Maximum Link Width (see Section 7.5.3.6). See Section 4.2.7.2 for details of this register.

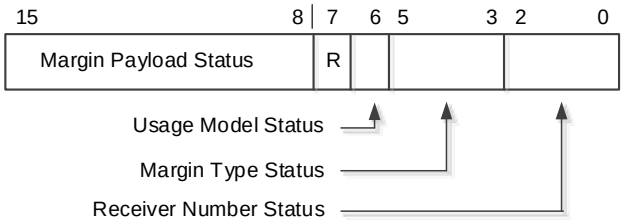


Figure 7-88: Lane N: Margining Lane Status Register Entry

Table 7-71: Lane N: Margining Lane Status Register Entry

Bit Location	Register Description	Attributes
	Control Fields	
2:0	**Receiver Number Status** – See Section 8.4.4 for details. The default value is 000b. For Downstream Ports, this field must be reset to the default value if the Port goes to DL_Down status.	RO (see description)
5:3	**Margin Type Status** – See Section 8.4.4 for details. The default value is 000b. This field must be reset to the default value if the Port goes to DL_Down status. RO (see description)	RO (see description)
6	**Usage Model Status** – See Section 8.4.4 for details. The default value is 0b. This field must be reset to the default value if the Port goes to DL_Down status.	RO (see description)
15:8	**MarginPayload Status** – See Section 8.4.4 for details. This field is only meaningful, when the Margin Type is a defined encoding other than 'No Command'. The default value is 00h. This field must be reset to the default value if the Port goes to DL_Down status.	RO (see description)

7.7.7 ACS Extended Capability

The ACS Extended Capability is an optional capability that provides enhanced access controls (see Section 6.12). This capability may be implemented by a Root Port, a Switch Downstream Port, or a Multi-Function Device Function. It is never applicable to a PCI Express to PCI Bridge or Root

Complex Event Collector. It is not applicable to a Switch Upstream Port unless that Switch Upstream Port is a Function in a Multi-Function Device.

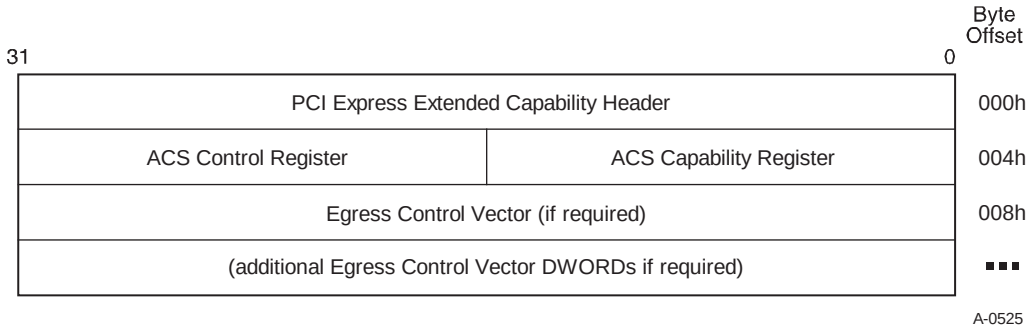


Figure 7-89: ACS Extended Capability

7.7.7.1 *ACS Extended Capability Header (Offset 00h)*

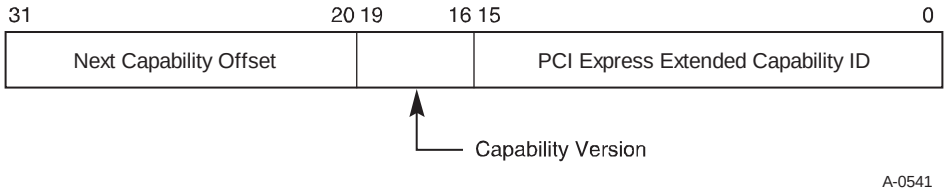


Figure 7-90: ACS Extended Capability Header

Table 7-72: ACS Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. PCI Express Extended Capability ID for the ACS Extended Capability is 000Dh.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express Extended Capability structure or 000h if no other items exist in the linked list of Capabilities.	RO

7.7.7.2 ACS Capability Register (Offset 04h)

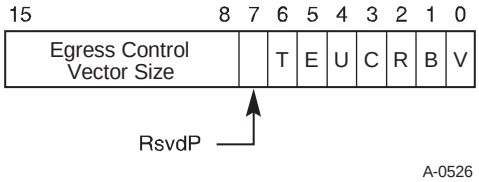


Figure 7-91: ACS Capability Register

Table 7-73: ACS Capability Register

Bit Location	Register Description	Attributes
0	**ACS Source Validation (V)** – Required for Root Ports and Switch Downstream Ports; must be hardwired to 0b otherwise. If 1b, indicates that the component implements ACS Source Validation.	RO
1	**ACS Translation Blocking (B)** – Required for Root Ports and Switch Downstream Ports; must be hardwired to 0b otherwise. If 1b, indicates that the component implements ACS Translation Blocking.	RO
2	**ACS P2P Request Redirect (R)** – Required for Root Ports that support peer-to-peer traffic with other Root Ports; required for Switch Downstream Ports; required for Multi-Function Device Functions that support peer-to-peer traffic with other Functions; must be hardwired to 0b otherwise. If 1b, indicates that the component implements ACS P2P Request Redirect.	RO
3	**ACS P2P Completion Redirect (C)** – Required for all Functions that support ACS P2P Request Redirect; must be hardwired to 0b otherwise. If 1b, indicates that the component implements ACS P2P Completion Redirect.	RO
4	**ACS Upstream Forwarding (U)** – Required for Root Ports if the RC supports Redirected Request Validation; required for Switch Downstream Ports; must be hardwired to 0b otherwise. If 1b, indicates that the component implements ACS Upstream Forwarding.	RO
5	**ACS P2P Egress Control (E)** – Optional for Root Ports, Switch Downstream Ports, and Multi-Function Device Functions; must be hardwired to 0b otherwise. If 1b, indicates that the component implements ACS P2P Egress Control.	RO
6	**ACS Direct Translated P2P (T)** – Required for Root Ports that support Address Translation Services (ATS) and also support peer-to-peer traffic with other Root Ports; required for Switch Downstream Ports; required for Multi-Function Device Functions that support Address Translation Services (ATS) and also support peer-to-peer traffic with other Functions; must be hardwired to 0b otherwise. If 1b, indicates that the component implements ACS Direct Translated P2P.	RO

Bit Location	Register Description	Attributes
15:8	**Egress Control Vector Size** – Encodings 01h-FFh directly indicate the number of applicable bits in the Egress Control Vector; the encoding 00h indicates 256 bits. If the ACS P2P Egress Control (E) bit is 0b, the value of the size field is undefined, and the Egress Control Vector register is not required to be present.	HwInit

7.7.7.3 *ACS Control Register (Offset 06h)*

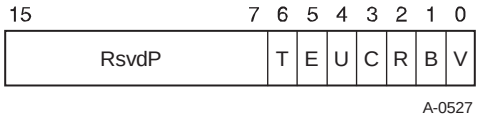


Figure 7-92: ACS Control Register

Table 7-74: ACS Control Register

Bit Location	Register Description	Attributes
0	**ACS Source Validation Enable (V)** – When Set, the component validates the Bus Number from the Requester ID of Upstream Requests against the secondary/subordinate Bus Numbers. Default value of this bit is 0b. Must be hardwired to 0b if the ACS Source Validation functionality is not implemented.	RW
1	**ACS Translation Blocking Enable (B)** – When Set, the component blocks all Upstream Memory Requests whose Address Translation (AT) field is not set to the default value. Default value of this bit is 0b. Must be hardwired to 0b if the ACS Translation Blocking functionality is not implemented.	RW
2	**ACS P2P Request Redirect Enable (R)** – In conjunction with ACS P2P Egress Control and ACS Direct Translated P2P mechanisms, determines when the component redirects peer-to-peer Requests Upstream (see Section 6.12.3). Note that with Downstream Ports, this bit only applies to Upstream Requests arriving at the Downstream Port, and whose normal routing targets a different Downstream Port. Default value of this bit is 0b. Must be hardwired to 0b if the ACS P2P Request Redirect functionality is not implemented.	RW
3	**ACS P2P Completion Redirect Enable (C)** – Determines when the component redirects peer-to-peer Completions Upstream; applicable only to Completions[130] whose Relaxed Ordering Attribute is clear. Default value of this bit is 0b. Must be hardwired to 0b if the ACS P2P Completion Redirect functionality is not implemented.	RW

[130] This includes Read Completions, AtomicOp Completions, and other Completions with or without Data.

Bit Location	Register Description	Attributes
4	**ACS Upstream Forwarding Enable (U)** – When Set, the component forwards Upstream any Request or Completion TLPs it receives that were redirected Upstream by a component lower in the hierarchy. Note that this bit only applies to Upstream TLPs arriving at a Downstream Port, and whose normal routing targets the same Downstream Port. Default value of this bit is 0b. Must be hardwired to 0b if the ACS Upstream Forwarding functionality is not implemented.	RW
5	**ACS P2P Egress Control Enable (E)** – In conjunction with the Egress Control Vector plus the ACS P2P Request Redirect and ACS Direct Translated P2P mechanisms, determines when to allow, disallow, or redirect peer-to-peer Requests (see Section 6.12.3). Default value of this bit is 0b. Must be hardwired to 0b if the ACS P2P Egress Control functionality is not implemented.	RW
6	**ACS Direct Translated P2P Enable (T)** – When Set, overrides the ACS P2P Request Redirect and ACS P2P Egress Control mechanisms with peer-to-peer Memory Requests whose Address Translation (AT) field indicates a Translated address (see Section 6.12.3). This bit is ignored if ACS Translation Blocking Enable (B) is 1b. Default value of this bit is 0b. Must be hardwired to 0b if the ACS Direct Translated P2P functionality is not implemented.	RW

7.7.7.4 *Egress Control Vector (Offset 08h)*

The Egress Control Vector is a read-write register that contains a bit-array. The number of bits in the register is specified by the Egress Control Vector Size field, and the register spans multiple DWORDs if required. If the ACS P2P Egress Control bit in the ACS Capability register is 0b, the Egress Control Vector Size field is undefined and the Egress Control Vector register is not required to be present.

For the general case of an Egress Control Vector spanning multiple DWORDs, the DWORD offset and bit number within that DWORD for a given arbitrary bit *K* are specified by the formulas[131]:

DWORD offset = 08h + (K div 32) * 4

DWORD bit# = K mod 32

Bits in a DWORD beyond those specified by the Egress Control Vector Size field are RsvdP.

For Root Ports and Switch Downstream Ports, each bit in the bit-array always corresponds to a Port Number. Otherwise, for Functions[132] within a Multi-Function Device, each bit in the bit-array corresponds to one or more Function Numbers, or a Function Group Number. For example, access to Function 2 is controlled by bit number 2 in the bit-array. For both Port Number cases and Function Number cases, the bit corresponding to the Function that implements this Extended Capability structure must be hardwired to 0b.[133]

[131] Div is an integer divide with truncation. Mod is the remainder from an integer divide.

[132] Including Switch Upstream Ports.

[133] For ARI Devices, the bit must be RW. See subsequent description.

If an ARI Device implements ACS Function Groups, its Egress Control Vector Size is required to be a power-of-2 from 8 to 256, and all of its implemented Egress Control Vector bits must be RW. With ARI Devices, multiple Functions can be associated with a single bit, so for each Function, its associated bit determines how Requests from it targeting other Functions (if any) associated with the same bit are handled.

If ACS Function Groups are enabled in an ARI Device, the first 8 Egress Control Vector bits in each Function are associated with Function Group Numbers instead of Function Numbers. In this case, access control is enforced between Function Groups instead of Functions, and any implemented Egress Control Vector bits beyond the first 8 are unused.

Independent of whether an ARI Device implements ACS Function Groups, its Egress Control Vector Size is not required to cover the entire Function Number range of all Functions implemented by the Device. If ACS Function Groups are not enabled, Function Numbers are mapped to implemented Egress Control Vector bits by taking the modulo of the Egress Control Vector Size, which is constrained to be a power-of-2.

With RCs, some Port Numbers may refer to internal Ports instead of Root Ports. For Root Ports in such RCs, each bit in the bit-array that corresponds to an internal Port must be hardwired to 0b.

Figure 7-93: Egress Control Vector Register

Table 7-75: Egress Control Vector

Bit Location	Register Description	Attributes
N-1:0	**Egress Control Vector** – An N-bit bit-array configured by software, where N is given by the value in the Egress Control Vector Size field. When a given bit is set, peer-to-peer Requests targeting the associated Port, Function, or Function Group are blocked or redirected (if enabled) (see Section 6.12.3). Default value of each bit is 0b.	RW

The following examples illustrate how the vector might be configured:

- For an 8-Port Switch, each Port will have a separate vector indicating which Downstream Egress Ports it may forward Requests to.

Port 1 being not allowed to communicate with any other Downstream Ports would be configured as: 1111 1100b with bit 0 corresponding to the Upstream Port (hardwired to 0b) and bit 1 corresponding to the Ingress Port (hardwired to 0b).

Port 2 being allowed to communicate with Ports 3, 5, and 7 would be configured as: 0101 0010b.

- For a 4-Function device, each Function will have a separate vector that indicates which Function it may forward Requests to.

Function 0 being not allowed to communicate with any other Functions would be configured as: 1110b with bit 0 corresponding to Function 0 (hardwired to 0b).

Function 1 being allowed to communicate with Functions 2 and 3 would be configured as: 0001b with bit 1 corresponding to Function 1 (hardwired to 0b).

7.8 Common PCI and PCIe Capabilities

This section, contains a description of common PCI and PCIe capabilities that are individually optional in the *PCI Express Base Specification* but may be required by other PCISIG specifications.

7.8.1 Power Budgeting Capability

The PCI Express Power Budgeting Capability allows the system to allocate power to devices that are added to the system at runtime. Through this Capability, a device can report the power it consumes on a variety of power rails, in a variety of device power-management states, in a variety of operating conditions. The system can use this information to ensure that the system is capable of providing the proper power and cooling levels to the device. Failure to indicate proper device power consumption may risk device or system failure.

Implementation of the Power Budgeting Capability is optional for PCI Express devices that are implemented either in a form factor which does not require Hot-Plug support, or that are integrated on the system board. PCI Express form factor specifications may require support for power budgeting. Figure 7-94 details allocation of register fields in the Power Budgeting Capability structure.

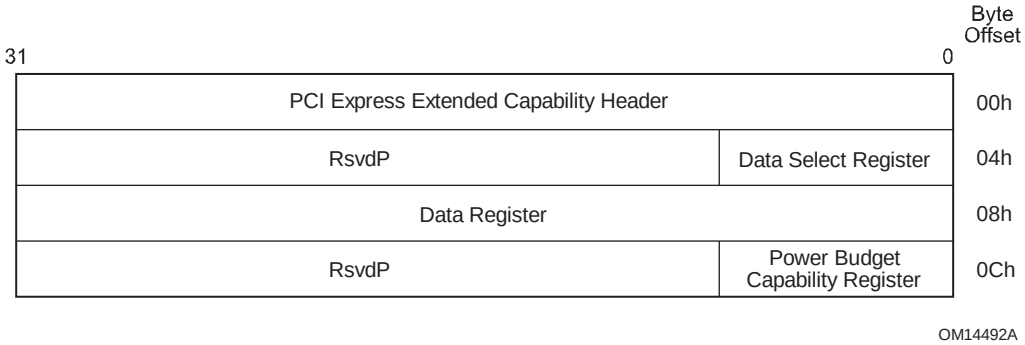


Figure 7-94: PCI Express Power Budgeting Capability Structure

7.8.1.1 *Power Budgeting Extended Capability Header (Offset 00h)*

Figure 7-95 details allocation of register fields in the Power Budgeting Extended Capability header; Table 7-76 provides the respective bit definitions. Refer to Section 7.6.3 for a description of the PCI Express Extended Capability header. The Extended Capability ID for the Power Budgeting Capability is 0004h.

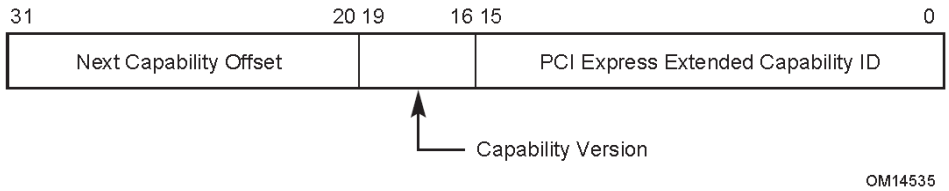


Figure 7-95: Power Budgeting Extended Capability Header

Table 7-76: Power Budgeting Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. Extended Capability ID for the Power Budgeting Capability is 0004h.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh.	RO

7.8.1.2 *Data Select Register (Offset 04h)*

The Data Select Register is an 8-bit read-write register that indexes the Power Budgeting Data reported through the Data register and selects the DWORD of Power Budgeting Data that is to appear in the Data register. Values for this register start at zero to select the first DWORD of Power Budgeting Data; subsequent DWORDs of Power Budgeting Data are selected by increasing index values. The default value of this register is undefined.

7.8.1.3 *Data Register (Offset 08h)*

This read-only register returns the DWORD of Power Budgeting Data selected by the Data Select register. Each DWORD of the Power Budgeting Data describes the power usage of the device in a particular operating condition. Power Budgeting Data for different operating conditions is not required to be returned in any particular order, as long as incrementing the Data Select register causes information for a different operating condition to be returned. If the Data Select register contains a value greater than or equal to the number of operating conditions for which the device provides power information, this register must return all zeros. The default value of this register is

undefined. Figure 7-96 details allocation of register fields in the Power Budgeting Data register; Table 7-77 provides the respective bit definitions.

The Base Power and Data Scale fields describe the power usage of the device; the Power Rail, Type, PM State, and PM Sub State fields describe the conditions under which the device has this power usage.

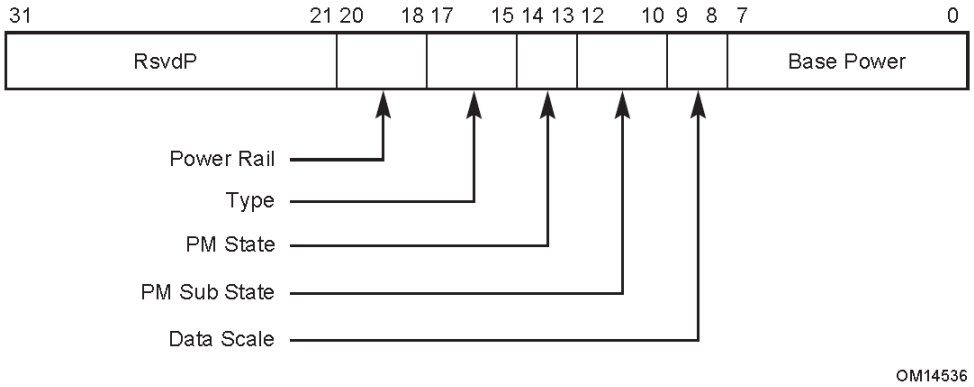


Figure 7-96: Power Budgeting Data Register

Table 7-77: Power Budgeting Data Register

Bit Location	Register Description	Attributes
7:0	**Base Power** – Specifies in watts the base power value in the given operating condition. This value must be multiplied by the data scale to produce the actual power consumption value except when the Data Scale field equals 00b (1.0x) and Base Power exceeds EFh, the following alternative encodings are used: F0h = greater than 239 W and less than or equal to 250 W Slot Power Limit F1h = greater than 250 W and less than or equal to 275 W Slot Power Limit F2h = greater than 275 W and less than or equal to 300 W Slot Power Limit F3h to FFh = Reserved for values greater than 300 W	RO

Bit Location	Register Description	Attributes
9:8	**Data Scale** – Specifies the scale to apply to the Base Power value. The power consumption of the device is determined by multiplying the contents of the Base Power field with the value corresponding to the encoding returned by this field, except as noted above. Defined encodings are: 00b 1.0x 01b 0.1x 10b 0.01x 11b 0.001x	RO
12:10	**PM Sub State** – Specifies the power management sub state of the operating condition being described. Defined encodings are: 000b Default Sub State 001b – 111b Device Specific Sub State	RO
14:13	**PM State** – Specifies the power management state of the operating condition being described. Defined encodings are: 00b D0 01b D1 10b D2 11b D3 A device returns 11b in this field and Aux or PME Aux in the Type field to specify the D3-Cold PM State. An encoding of 11b along with any other Type field value specifies the D3-Hot state.	RO
17:15	**Type** – Specifies the type of the operating condition being described. Defined encodings are: 000b PME Aux 001b Auxiliary 010b Idle 011b Sustained 100b Sustained – Emergency Power Reduction State (see Section 6.25) 101b Maximum – Emergency Power Reduction State (see Section 6.25) 111b Maximum All other encodings are Reserved.	RO

Bit Location	Register Description	Attributes
20:18	**Power Rail** – Specifies the thermal load or power rail of the operating condition being described. Defined encodings are: 000b Power (12V) 001b Power (3.3V) 010b Power (1.5V or 1.8V) 111b Thermal All other encodings are Reserved.	RO

A device that implements the Power Budgeting Capability is required to provide data values for the D0 Maximum and D0 Sustained PM State and Type combinations for every power rail from which it consumes power; data for the D0 Maximum and D0 Sustained for Thermal must also be provided if these values are different from the sum of the values for an operating condition reported for D0 Maximum and D0 Sustained on the power rails.

Devices that support auxiliary power or PME from auxiliary power must provide data for the appropriate power type (Auxiliary or PME Aux).

If a device implements Emergency Power Reduction State, it must report Power Budgeting values for the following:

- Maximum Emergency Power Reduction State, PM State D0, all power rails used by the device
- Maximum Emergency Power Reduction State, PM State D0, Thermal (if different from the sum of the preceding values)
- Sustained Emergency Power Reduction State, PM State D0, all power rails used by the device
- Sustained Emergency Power Reduction State, PM State: D0, Thermal (if different from the sum of the preceding values)

7.8.1.4 *Power Budget Capability Register (Offset 0Ch)*

This register indicates the power budgeting capabilities of a device. Figure 7-97 details allocation of register fields in the Power Budget Capability register; Table 7-78 provides the respective bit definitions.

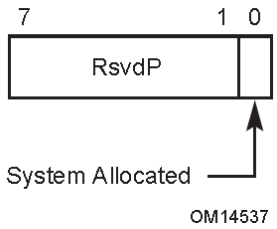


Figure 7-97: Power Budget Capability Register

Table 7-78: Power Budget Capability Register

Bit Location	Register Description	Attributes
0	**System Allocated** – When Set, this bit indicates that the power budget for the device is included within the system power budget. Reported Power Budgeting Data for this device must be ignored by software for power budgeting decisions if this bit is Set.	HwInit

7.8.2 Latency Tolerance Reporting (LTR) Capability

The PCI Express Latency Tolerance Reporting (LTR) Capability is an optional Extended Capability that allows software to provide platform latency information to components with Upstream Ports (Endpoints and Switches), and is required for Switch Upstream Ports and Endpoints if the Function supports the LTR mechanism. It is not applicable to Root Ports, Bridges, or Switch Downstream Ports.

For a Multi-Function Device associated with the Upstream Port of a component that implements the LTR mechanism, this Capability structure must be implemented only in Function 0, and must control the component's Link behavior on behalf of all the Functions of the Device.

RCiEPs implemented as Multi-Function Devices are permitted to implement this Capability structure in more than one Function of the Multi-Function Device.

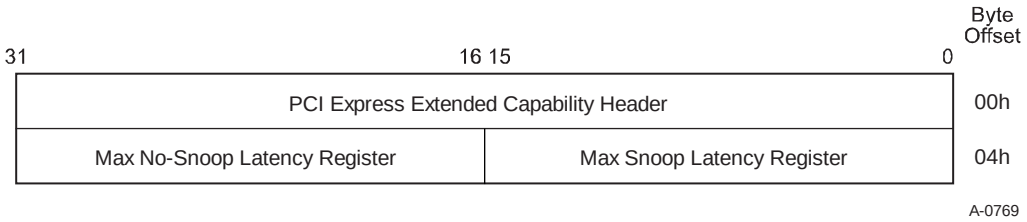


Figure 7-98: LTR Extended Capability Structure

7.8.2.1 *LTR Extended Capability Header (Offset 00h)*

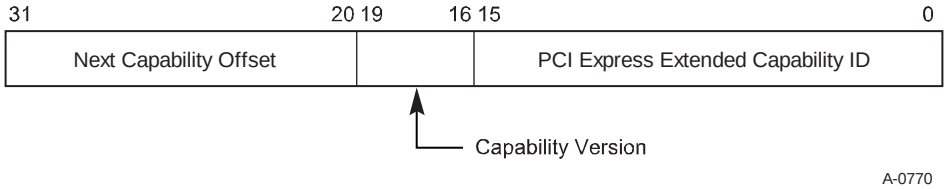


Figure 7-99: LTR Extended Capability Header

Table 7-79: LTR Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. PCI Express Extended Capability for the LTR Extended Capability is 0018h.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express Extended Capability structure or 000h if no other items exist in the linked list of Capabilities.	RO

7.8.2.2 *Max Snoop Latency Register (Offset 04h)*

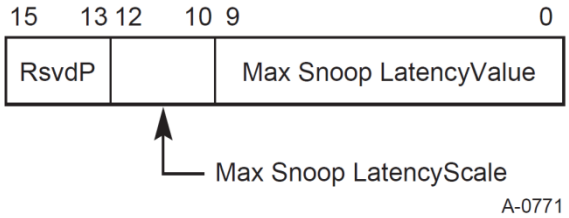


Figure 7-100: Max Snoop Latency Register

Table 7-80: Max Snoop Latency Register

Bit Location	Register Description	Attributes
9:0	**Max Snoop LatencyValue** – Along with the Max Snoop LatencyScale field, this register specifies the maximum snoop latency that a device is permitted to request. Software should set this to the platform's maximum supported latency or less. It is strongly recommended that any updates to this field are reflected in LTR Message(s) sent by the device within 1 ms. The default value for this field is 00 0000 0000b.	RW
12:10	**Max Snoop LatencyScale** – This register provides a scale for the value contained within the Maximum Snoop LatencyValue field. Encoding is the same as the LatencyScale fields in the LTR Message. See Section 6.18. It is strongly recommended that any updates to this field are reflected in LTR Message(s) sent by the device within 1 ms. The default value for this field is 000b. Hardware operation is undefined if software writes a Not Permitted value to this field.	RW

7.8.2.3 *Max No-Snoop Latency Register (Offset 06h)*

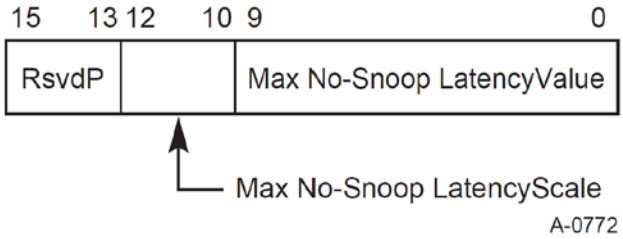


Figure 7-101: Max No-Snoop Latency Register

Table 7-81: Max No-Snoop Latency Register

Bit Location	Register Description	Attributes
9:0	**Max No-Snoop LatencyValue** – Along with the Max No-Snoop LatencyScale field, this register specifies the maximum no-snoop latency that a device is permitted to request. Software should set this to the platform's maximum supported latency or less. It is strongly recommended that any updates to this field are reflected in LTR Message(s) sent by the device within 1 ms. The default value for this field is 00 0000 0000b.	RW

Bit Location	Register Description	Attributes
12:10	**Max No-Snoop LatencyScale** – This register provides a scale for the value contained within the Max No-Snoop LatencyValue field. Encoding is the same as the LatencyScale fields in the LTR Message. See Section 6.18. It is strongly recommended that any updates to this field are reflected in LTR Message(s) sent by the device within 1 ms. The default value for this field is 000b. Hardware operation is undefined if software writes a Not Permitted value to this field.	RW

7.8.3 L1 PM Substates Extended Capability

The PCI Express L1 PM Substates Capability is an optional Extended Capability, that is required if L1 PM Substates is implemented at a Port. The L1 PM Substates Extended Capability structure is defined as shown in Figure 7-102.

For a Multi-Function Device associated with an Upstream Port implementing L1 PM Substates, this Extended Capability Structure must be implemented only in Function 0, and must control the Upstream Port's Link behavior on behalf of all the Functions of the device.

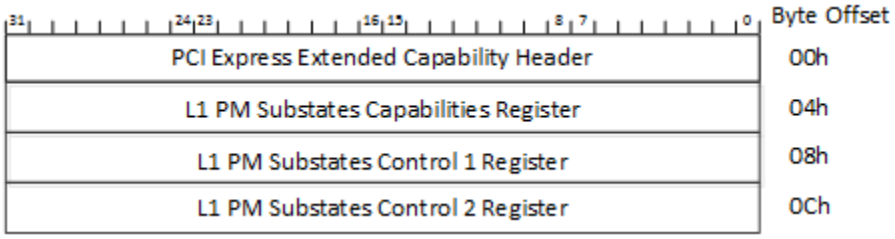


Figure 7-102: L1 PM Substates Capability

7.8.3.1 *L1 PM Substates Extended Capability Header (Offset 00h)*

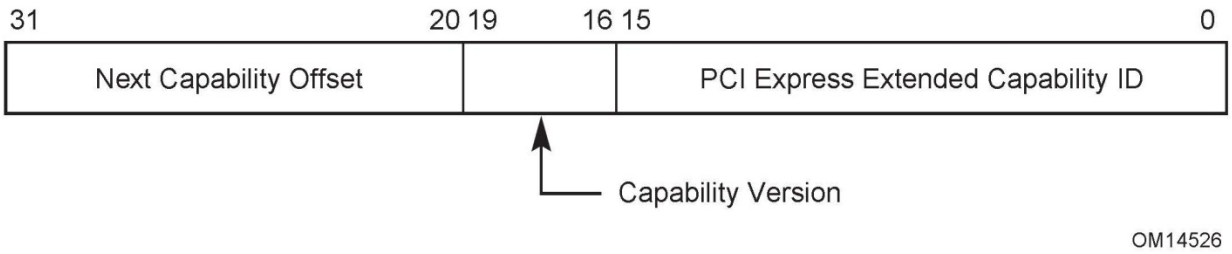


Figure 7-103: L1 PM Substates Extended Capability Header

Table 7-82: L1 PM Substates Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. Extended Capability ID for L1 PM Substates is 001Eh.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh. The bottom 2 bits of this offset are Reserved and must be implemented as 00b although software must mask them to allow for future uses of these bits.	RO

7.8.3.2 *L1 PM Substates Capabilities Register (Offset 04h)*

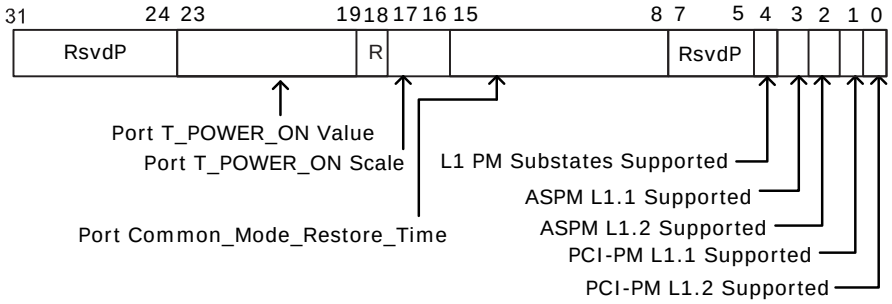


Figure 7-104: L1 PM Substates Capabilities Register

Table 7-83: L1 PM Substates Capabilities Register

Bit Location	Register Description	Attributes
0	**PCI-PM L1.2 Supported** – When Set this bit indicates that PCI-PM L1.2 is supported.	HwInit
1	**PCI-PM L1.1 Supported -** When Set this bit indicates that PCI-PM L1.1 is supported, and must be Set by all Ports implementing L1 PM Substates.	HwInit
2	**ASPM L1.2 Supported** – When Set this bit indicates that ASPM L1.2 is supported.	HwInit

Bit Location	Register Description	Attributes
3	**ASPM L1.1 Supported -** When Set this bit indicates that ASPM L1.1 is supported.	HwInit
4	**L1 PM Substates Supported** – When Set this bit indicates that this Port supports L1 PM Substates.	HwInit
7:5	**Reserved**	RsvdP
15:8	**Port Common_Mode_Restore_Time –** Time (in µs) required for this Port to re-establish common mode as described in Table 5-11. Required for all Ports for which either the PCI-PM L1.2 Supported bit is Set, ASPM L1.2 Supported bit is Set, or both are Set, otherwise this field is of type RsvdP.	HwInit/RsvdP (See description)
17:16	**Port T_POWER_ON Scale –** Specifies the scale used for the Port T_POWER_ON Value field in the L1 PM Substates Capabilities register. Range of Values 00b = 2 µs 01b = 10 µs 10b = 100 µs 11b = Reserved Required for all Ports for which either the PCI-PM L1.2 Supported bit is Set, ASPM L1.2 Supported bit is Set, or both are Set, otherwise this field is of type RsvdP. Default value is 00b	HwInit / RsvdP
18	**Reserved**	RsvdP
23:19	**Port T_POWER_ON Value –** Along with the Port T_POWER_ON Scale field in the L1 PM Substates Capabilities register sets the time (in µs) that this Port requires the port on the opposite side of Link to wait in L1.2.Exit after sampling CLKREQ# asserted before actively driving the interface. The value of Port T_POWER_ON is calculated by multiplying the value in this field by the scale value in the Port T_POWER_ON Scale field in the L1 PM Substates Capabilities register. Default value is 00101b Required for all Ports for which either the PCI-PM L1.2 Supported bit is Set, ASPM L1.2 Supported bit is Set, or both are Set, otherwise this field is of type RsvdP.	HwInit / RsvdP
31:24	**Reserved**	RsvdP

7.8.3.3 *L1 PM Substates Control 1 Register (Offset 08h)*

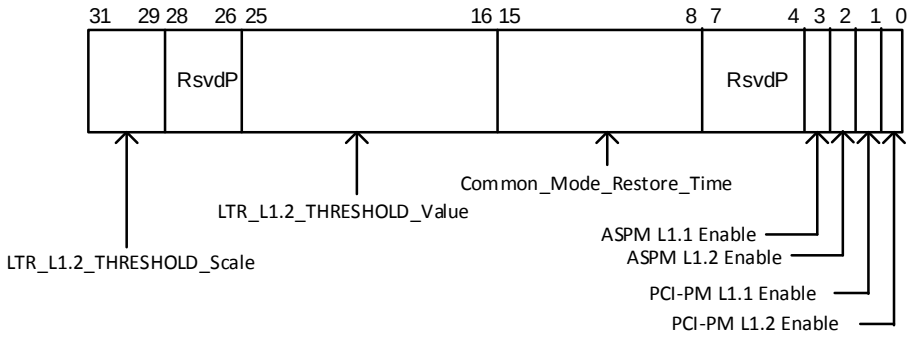


Figure 7-105: L1 PM Substates Control 1 Register

Table 7-84: L1 PM Substates Control 1 Register

Bit Location	Register Description	Attributes
0	**PCI-PM L1.2 Enable** – When Set this bit enables PCI-PM L1.2. Required for both Upstream and Downstream Ports. For Ports for which the PCI-PM L1.2 Supported bit is Clear this bit is permitted to be hardwired to 0. For compatibility with possible future extensions, software must not enable L1 PM Substates unless the L1 PM Substates Supported bit in the L1 PM Substates Capabilities Register is Set. Default value is 0b.	RW
1	**PCI-PM L1.1 Enable -** When Set this bit enables PCI-PM L1.1. Required for both Upstream and Downstream Ports. For compatibility with possible future extensions, software must not enable L1 PM Substates unless the L1 PM Substates Supported bit in the L1 PM Substates Capabilities Register is Set. Default value is 0b.	RW
2	**ASPM L1.2 Enable** – When Set this bit enables ASPM L1.2. Required for both Upstream and Downstream Ports. For Ports for which the ASPM L1.2 Supported bit is Clear this bit is permitted to be hardwired to 0. For compatibility with possible future extensions, software must not enable L1 PM Substates unless the L1 PM Substates Supported bit in the L1 PM Substates Capabilities Register is Set. Default value is 0b.	RW

Bit Location	Register Description	Attributes
3	**ASPM L1.1 Enable -** When Set this bit enables ASPM L1.1. Required for both Upstream and Downstream Ports. For Ports for which the ASPM L1.1 Supported bit is Clear this bit is permitted to be hardwired to 0. For compatibility with possible future extensions, software must not enable L1 PM Substates unless the L1 PM Substates Supported bit in the L1 PM Substates Capabilities Register is Set. Default value is 0b.	RW
7:4	**Reserved**	RsvdP
15:8	**Common_Mode_Restore_Time –** Sets value of $T_{COMMONMODE}$ (in µs), which must be used by the Downstream Port for timing the re-establishment of common mode, as described in Table 5-11. This field must only be modified when the ASPM L1.2 Enable and PCI-PM L1.2 Enable bits are both Clear. The Port behavior is undefined if this field is modified when either the ASPM L1.2 Enable and/or PCI-PM L1.2 Enable bit(s) are Set. Required for Downstream Ports for which either the PCI-PM L1.2 Supported bit is Set, ASPM L1.2 Supported bit is Set, or both are Set, otherwise this field is of type RsvdP. This field is of type RsvdP for Upstream Ports. Default value is implementation specific.	RW/RsvdP (See Description)
25:16	**LTR_L1.2_THRESHOLD_Value** – Along with the LTR_L1.2_THRESHOLD_Scale, this field indicates the LTR threshold used to determine if entry into L1 results in L1.1 (if enabled) or L1.2 (if enabled). The default value for this field is 00 0000 0000b. This field must only be modified when the ASPM L1.2 Enable bit is Clear. The Port behavior is undefined if this field is modified when the ASPM L1.2 Enable bit is Set. Required for all Ports for which the ASPM L1.2 Supported bit is Set, otherwise this field is of type RsvdP.	RW/RsvdP (See Description)
28:26	**Reserved**	RsvdP

Bit Location	Register Description	Attributes
31:29	**LTR_L1.2_THRESHOLD_Scale** – This field provides a scale for the value contained within the LTR_L1.2_THRESHOLD_Value. Encoding is the same as the LatencyScale fields in the LTR Message (see Section 6.18). The default value for this field is 000b. Hardware operation is undefined if software writes a Not-Permitted value to this field. This field must only be modified when the ASPM L1.2 Enable bit is Clear. The Port behavior is undefined if this field is modified when the ASPM L1.2 Enable bit is Set. Required for all Ports Ports for which the ASPM L1.2 Supported bit is Set, otherwise this field is of type RsvdP.	RW/RsvdP (See description)

7.8.3.4 *L1 PM Substates Control 2 Register (Offset 0Ch)*

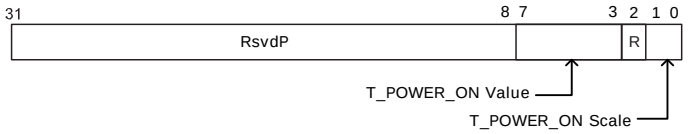


Figure 7-106: L1 PM Substates Control 2 Register

Table 7-85: L1 PM Substates Control 2 Register

Bit Location	Register Description	Attributes
1:0	**T_POWER_ON Scale –** Specifies the scale used for T_POWER_ON Value. Range of Values 00b = 2 µs 01b = 10 µs 10b = 100 µs 11b = Reserved Required for all Ports that support L1.2, otherwise this field is of type RsvdP. This field must only be modified when the ASPM L1.2 Enable and PCI-PM L1.2 Enable bits are both Clear. The Port behavior is undefined if this field is modified when either the ASPM L1.2 Enable and/or PCI-PM L1.2 Enable bit(s) are Set.	RW/ RsvdP

Bit Location	Register Description	Attributes
	Default value is 00b	
2	**Reserved**	RsvdP
7:3	**T_POWER_ON Value –** Along with the T_POWER_ON Scale sets the minimum amount of time (in µs) that the Port must wait in L1.2.Exit after sampling CLKREQ# asserted before actively driving the interface. T_POWER_ON is calculated by multiplying the value in this field by the value in the T_POWER_ON Scale field. This field must only be modified when the ASPM L1.2 Enable and PCI-PM L1.2 Enable bits are both Clear. The Port behavior is undefined if this field is modified when either the ASPM L1.2 Enable and/or PCI-PM L1.2 Enable bit(s) are Set. Default value is 00101b Required for all Ports that support L1.2, otherwise this field is of type RsvdP.	RW/ RsvdP
31:8	**Reserved**	RsvdP

7.8.4 Advanced Error Reporting Capability

The PCI Express Advanced Error Reporting Capability is an optional Extended Capability that may be implemented by PCI Express device Functions supporting advanced error control and reporting. The Advanced Error Reporting Capability structure definition has additional interpretation for Root Ports and Root Complex Event Collectors; software must interpret the Device/Port Type field in the PCI Express Capabilities register to determine the availability of additional registers for Root Ports and Root Complex Event Collectors.

Figure 7-107 shows the PCI Express Advanced Error Reporting Capability structure.

Note that if an error reporting bit field is marked as optional in the error registers, the bits must be implemented or not implemented as a group across the Status, Mask and Severity registers. In other words, a Function is required to implement the same error bit fields in corresponding Status, Mask and Severity registers. Bits corresponding to bit fields that are not implemented must be hardwired to 0, unless otherwise specified.

Except for Root Ports and Root Complex Event Collectors, if the End-End TLP Prefix Supported bit is Set, the Root Error Command and Error Source Identification registers must be RsvdP and the Root Error Status register must be RsvdZ.

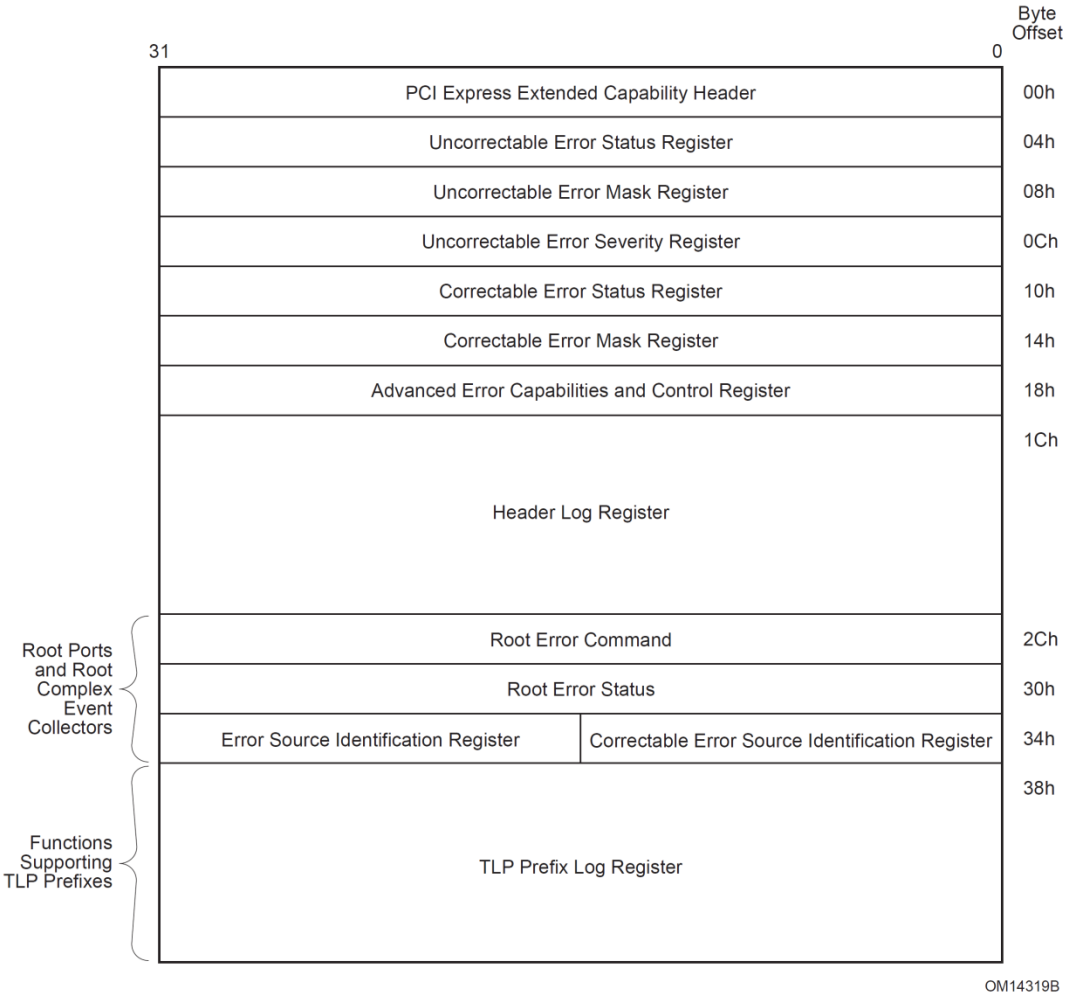


Figure 7-107: PCI Express Advanced Error Reporting Extended Capability Structure

7.8.4.1 *Advanced Error Reporting Extended Capability Header (Offset 00h)*

Figure 7-108 details the allocation of register fields of an Advanced Error Reporting Extended Capability header; Table 7-86 provides the respective bit definitions.

Refer to Section 7.6.3 for a description of the PCI Express Extended Capability header. The Extended Capability ID for the Advanced Error Reporting Capability is 0001h.

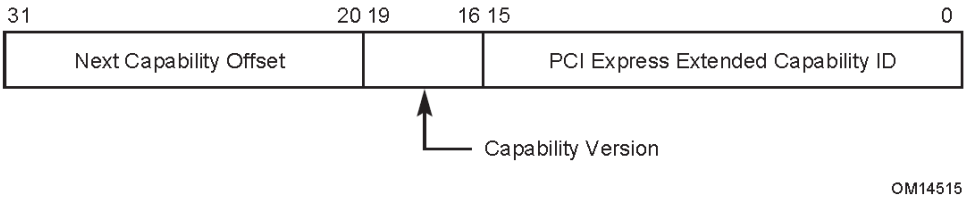


Figure 7-108: Advanced Error Reporting Extended Capability Header

Table 7-86: Advanced Error Reporting Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. The Extended Capability ID for the Advanced Error Reporting Capability is 0001h.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. This field must be 2h if the End-End TLP Prefix Supported bit (see Section 7.5.3.15) is Set and must be 1h or 2h otherwise.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh.	RO

7.8.4.2 *Uncorrectable Error Status Register (Offset 04h)*

The Uncorrectable Error Status register indicates error detection status of individual errors on a PCI Express device Function. An individual error status bit that is Set indicates that a particular error was detected; software may clear an error status by writing a 1b to the respective bit. Refer to Section 6.2 for further details. Register bits not implemented by the Function are hardwired to 0b. Figure 7-109 details the allocation of register fields of the Uncorrectable Error Status register; Table 7-87 provides the respective bit definitions.

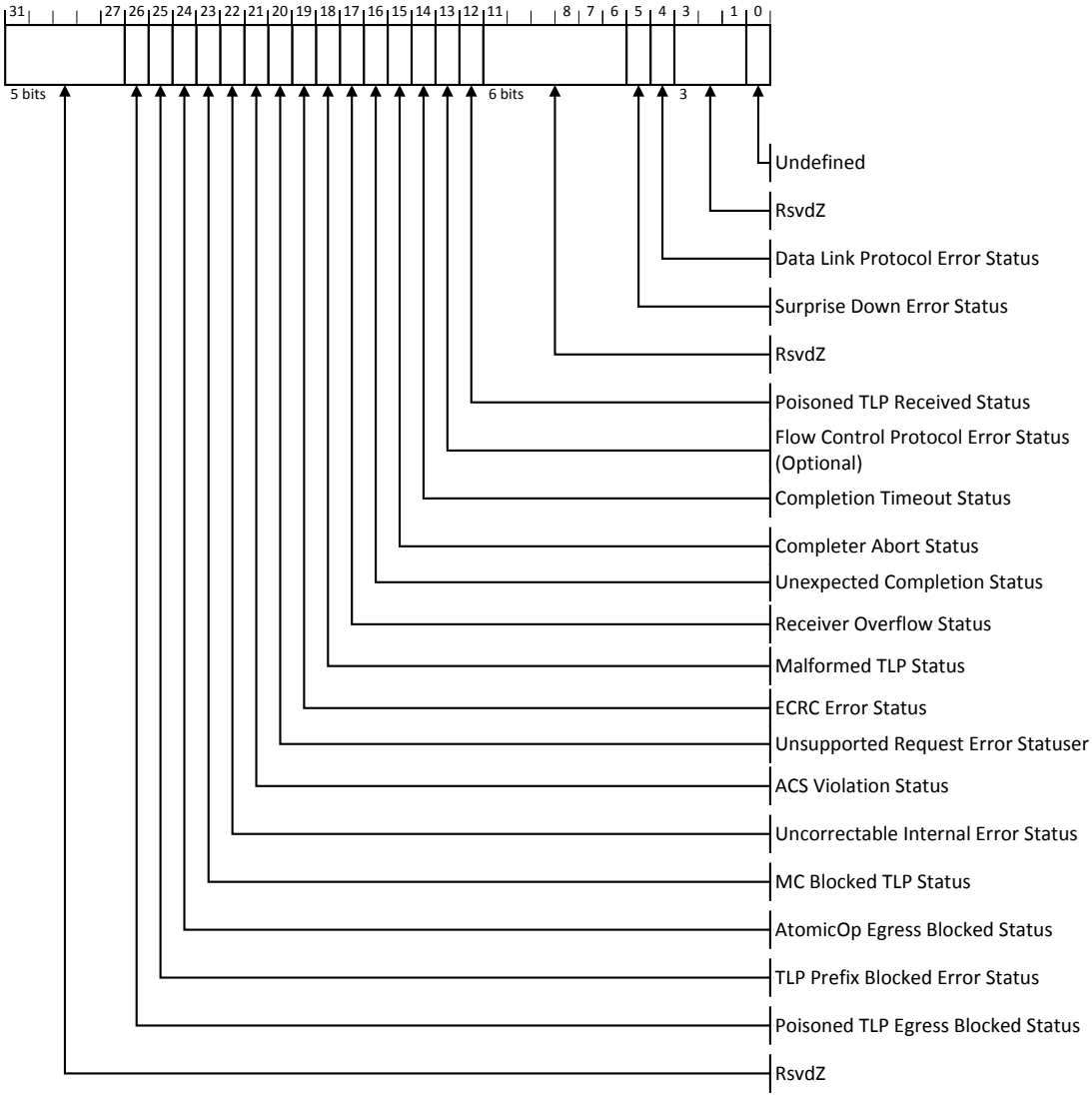


Figure 7-109: Uncorrectable Error Status Register

Table 7-87: Uncorrectable Error Status Register

Bit Location	Register Description	Attributes	Default
0	**Undefined** – The value read from this bit is undefined. In previous versions of this specification, this bit was used to indicate a Link Training Error. System software must ignore the value read from this bit. System software is permitted to write any value to this bit.	Undefined	Undefined
4	**Data Link Protocol Error Status**	RW1CS	0b
5	**Surprise Down Error Status (Optional)**	RW1CS	0b
12	**Poisoned TLP Received Status**	RW1CS	0b
13	**Flow Control Protocol Error Status** (Optional)	RW1CS	0b

Bit Location	Register Description	Attributes	Default
14	**Completion Timeout Status**[134]	RW1CS	0b
15	**Completer Abort Status** (Optional)	RW1CS	0b
16	**Unexpected Completion Status**	RW1CS	0b
17	**Receiver Overflow Status** (Optional)	RW1CS	0b
18	**Malformed TLP Status**	RW1CS	0b
19	**ECRC Error Status** (Optional)	RW1CS	0b
20	**Unsupported Request Error Status**	RW1CS	0b
21	**ACS Violation Status** (Optional)	RW1CS	0b
22	**Uncorrectable Internal Error Status** (Optional)	RW1CS	0b
23	**MC Blocked TLP Status** (Optional)	RW1CS	0b
24	**AtomicOp Egress Blocked Status** (Optional)	RW1CS	0b
25	**TLP Prefix Blocked Error Status** (Optional)	RW1CS	0b
26	**Poisoned TLP Egress Blocked Status** (Optional)	RW1CS	0b

7.8.4.3 *Uncorrectable Error Mask Register (Offset 08h)*

The Uncorrectable Error Mask register controls reporting of individual errors by the device Function to the PCI Express Root Complex via a PCI Express error Message. A masked error (respective bit Set in the mask register) is not recorded or reported in the Header Log, TLP Prefix Log, or First Error Pointer, and is not reported to the PCI Express Root Complex by this Function. Refer to Section 6.2 for further details. There is a mask bit per error bit of the Uncorrectable Error Status register. Register fields for bits not implemented by the Function are hardwired to 0b. Figure 7-110 details the allocation of register fields of the Uncorrectable Error Mask register; Table 7-88 provides the respective bit definitions.

[134] For Switch Ports, required if the Switch Port issues Non-Posted Requests on its own behalf (vs. only forwarding such Requests generated by other devices). If the Switch Port does not issue such Requests, then the Completion Timeout mechanism is not applicable and this bit must be hardwired to 0b.

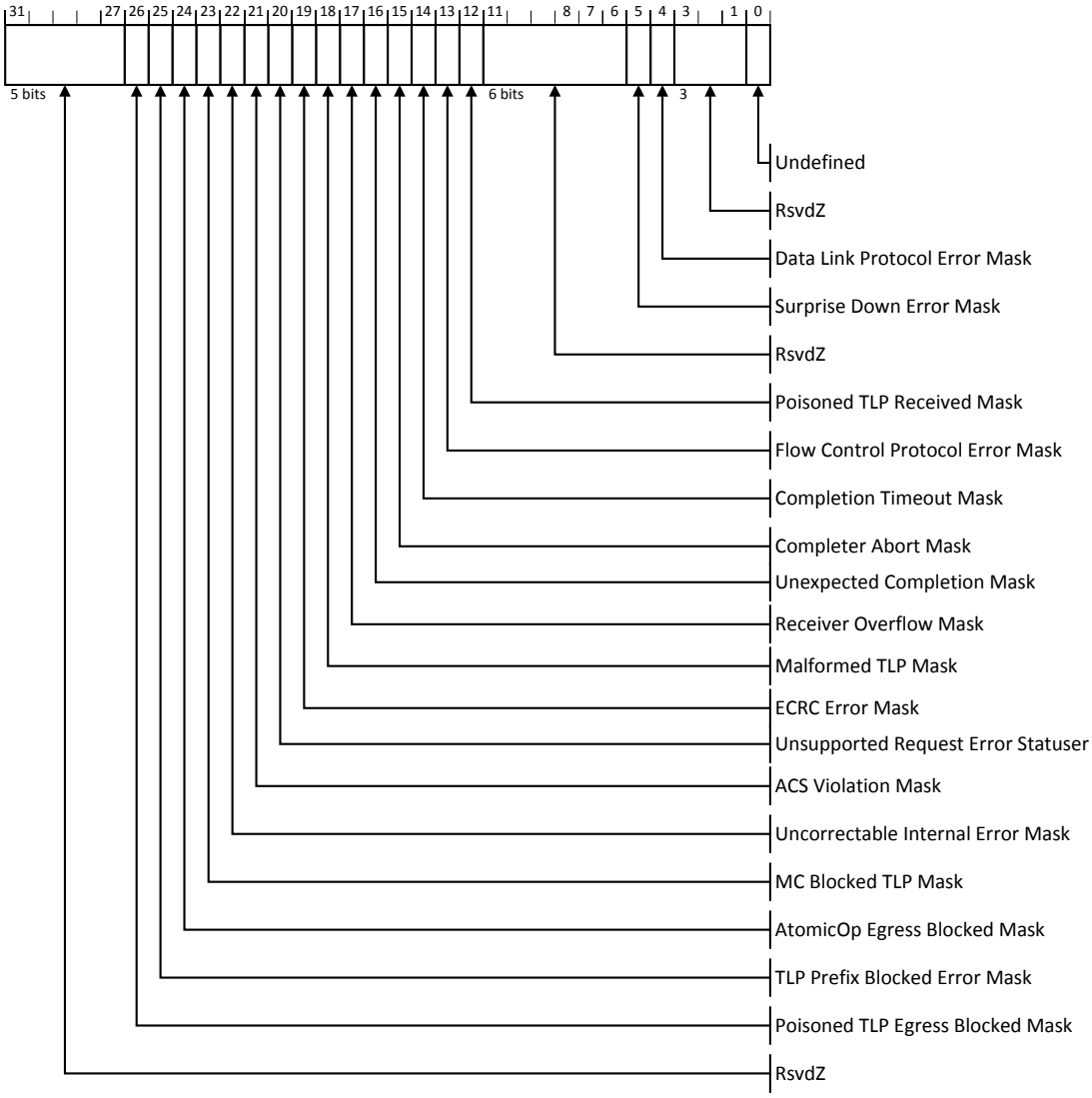


Figure 7-110: Uncorrectable Error Mask Register

Table 7-88: Uncorrectable Error Mask Register

Bit Location	Register Description	Attributes	Default
0	**Undefined** – The value read from this bit is undefined. In previous versions of this specification, this bit was used to mask a Link Training Error. System software must ignore the value read from this bit. System software must only write a value of 1b to this bit.	Undefined	Undefined
4	**Data Link Protocol Error Mask**	RWS	0b
5	**Surprise Down Error Mask (Optional)**	RWS	0b
12	**Poisoned TLP Received Mask**	RWS	0b
13	**Flow Control Protocol Error Mask** (Optional)	RWS	0b

Bit Location	Register Description	Attributes	Default
14	**Completion Timeout Mask**[135]	RWS	0b
15	**Completer Abort Mask** (Optional)	RWS	0b
16	**Unexpected Completion Mask**	RWS	0b
17	**Receiver Overflow Mask** (Optional)	RWS	0b
18	**Malformed TLP Mask**	RWS	0b
19	**ECRC Error Mask** (Optional)	RWS	0b
20	**Unsupported Request Error Mask**	RWS	0b
21	**ACS Violation Mask** (Optional)	RWS	0b
22	**Uncorrectable Internal Error Mask** (Optional)	RWS	1b
23	**MC Blocked TLP Mask** (Optional)	RWS	0b
24	**AtomicOp Egress Blocked Mask** (Optional)	RWS	0b
25	**TLP Prefix Blocked Error Mask** (Optional)	RWS	0b
26	**Poisoned TLP Egress Blocked Mask** (Optional)	RWS	1b

7.8.4.4 *Uncorrectable Error Severity Register (Offset 0Ch)*

The Uncorrectable Error Severity register controls whether an individual error is reported as a Non-fatal or Fatal error. An error is reported as fatal when the corresponding error bit in the severity register is Set. If the bit is Clear, the corresponding error is considered non-fatal. Refer to Section 6.2 for further details. Register fields for bits not implemented by the Function are hardwired to an implementation specific value. Figure 7-111 details the allocation of register fields of the Uncorrectable Error Severity register; Table 7-89 provides the respective bit definitions.

[135] For Switch Ports, required if the Switch Port issues Non-Posted Requests on its own behalf (vs. only forwarding such Requests generated by other devices). If the Switch Port does not issue such Requests, then the Completion Timeout mechanism is not applicable and this bit must be hardwired to 0b.

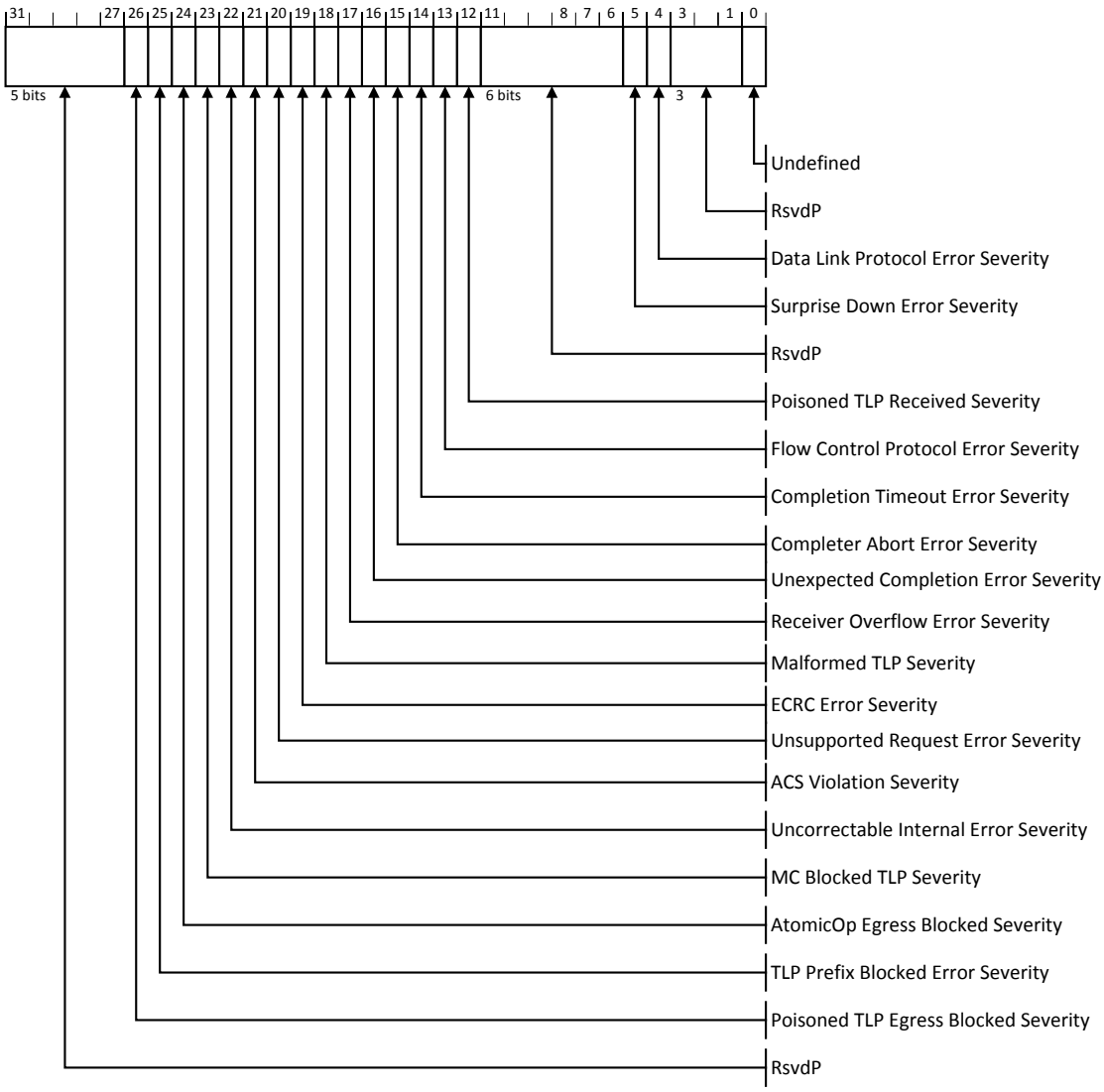


Figure 7-111: Uncorrectable Error Severity Register

Table 7-89: Uncorrectable Error Severity Register

Bit Location	Register Description	Attributes	Default
0	**Undefined** – The value read from this bit is undefined. In previous versions of this specification, this bit was used to Set the severity of a Link Training Error. System software must ignore the value read from this bit. System software is permitted to write any value to this bit.	Undefined	Undefined
4	**Data Link Protocol Error Severity**	RWS	1b
5	**Surprise Down Error Severity (Optional)**	RWS	1b
12	**Poisoned TLP Received Severity**	RWS	0b
13	**Flow Control Protocol Error Severity** (Optional)	RWS	1b

Bit Location	Register Description	Attributes	Default
14	**Completion Timeout Error Severity**[136]	RWS	0b
15	**Completer Abort Error Severity** (Optional)	RWS	0b
16	**Unexpected Completion Error Severity**	RWS	0b
17	**Receiver Overflow Error Severity** (Optional)	RWS	1b
18	**Malformed TLP Severity**	RWS	1b
19	**ECRC Error Severity** (Optional)	RWS	0b
20	**Unsupported Request Error Severity**	RWS	0b
21	**ACS Violation Severity** (Optional)	RWS	0b
22	**Uncorrectable Internal Error Severity** (Optional)	RWS	1b
23	**MC Blocked TLP Severity** (Optional)	RWS	0b
24	**AtomicOp Egress Blocked Severity** (Optional)	RWS	0b
25	**TLP Prefix Blocked Error Severity** (Optional)	RWS	0b
26	**Poisoned TLP Egress Blocked Severity** (Optional)	RWS	0b

7.8.4.5 *Correctable Error Status Register (Offset 10h)*

The Correctable Error Status register reports error status of individual correctable error sources on a PCI Express device Function. When an individual error status bit is Set, it indicates that a particular error occurred; software may clear an error status by writing a 1b to the respective bit. Refer to Section 6.2 for further details. Register bits not implemented by the Function are hardwired to 0b. Figure 7-112 details the allocation of register fields of the Correctable Error Status register; Table 7-90 provides the respective bit definitions.

[136] For Switch Ports, required if the Switch Port issues Non-Posted Requests on its own behalf (vs. only forwarding such Requests generated by other devices). If the Switch Port does not issue such Requests, then the Completion Timeout mechanism is not applicable and this bit must be hardwired to 0b.

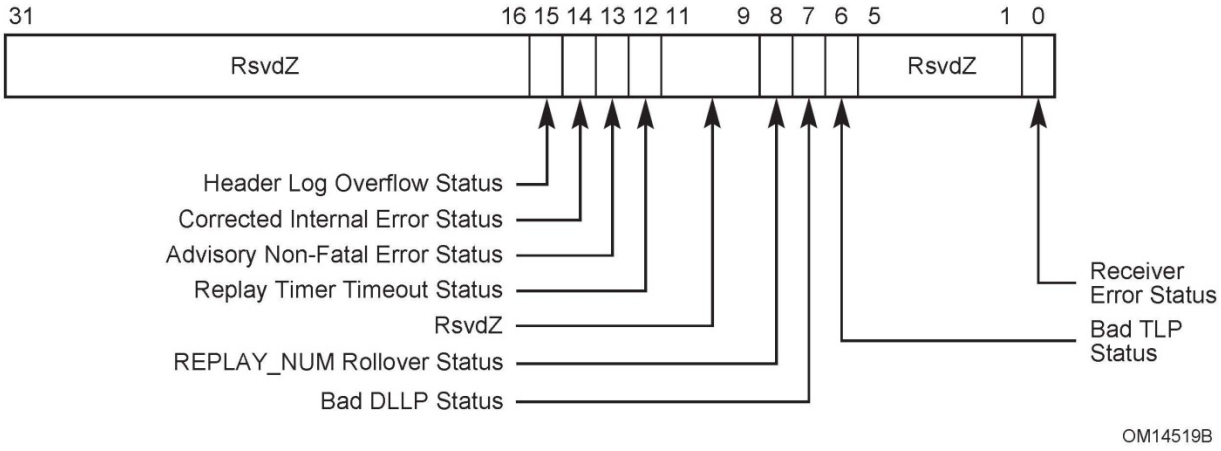


Figure 7-112: Correctable Error Status Register

Table 7-90: Correctable Error Status Register

Bit Location	Register Description	Attributes	Default
0	**Receiver Error Status**[137]	RW1CS	0b
6	**Bad TLP Status**	RW1CS	0b
7	**Bad DLLP Status**	RW1CS	0b
8	**REPLAY_NUM Rollover Status**	RW1CS	0b
12	**Replay Timer Timeout Status**	RW1CS	0b
13	**Advisory Non-Fatal Error Status**	RW1CS	0b
14	**Corrected Internal Error Status** (Optional)	RW1CS	0b
15	**Header Log Overflow Status** (Optional)	RW1CS	0b

7.8.4.6 *Correctable Error Mask Register (Offset 14h)*

The Correctable Error Mask register controls reporting of individual correctable errors by this Function to the PCI Express Root Complex via a PCI Express error Message. A masked error (respective bit Set in the mask register) is not reported to the PCI Express Root Complex by this Function. Refer to Section 6.2 for further details. There is a mask bit per error bit in the Correctable Error Status register. Register fields for bits not implemented by the Function are hardwired to 0b. Figure 7-113 details the allocation of register fields of the Correctable Error Mask register; Table 7-91 provides the respective bit definitions.

[137] For historical reasons, implementation of this bit is optional. If not implemented, this bit must be RsvdZ, and bit 0 of the Correctable Error Mask register must also not be implemented. Note that some checking for Receiver Errors is required in all cases (see Sections 4.2.1.1.3, 4.2.4.7, and 4.2.6).

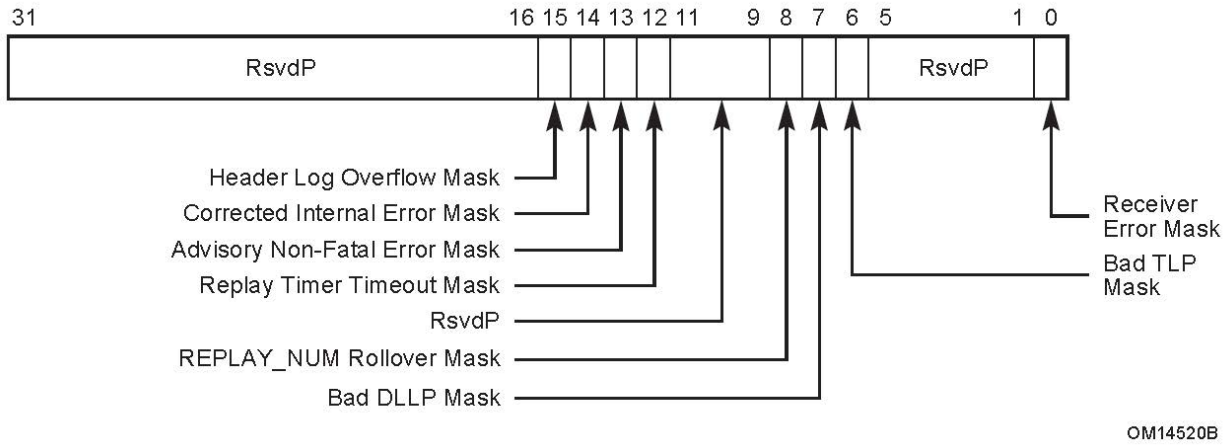


Figure 7-113: Correctable Error Mask Register

Table 7-91: Correctable Error Mask Register

Bit Location	Register Description	Attributes	Default
0	**Receiver Error Mask**[138]	RWS	0b
6	**Bad TLP Mask**	RWS	0b
7	**Bad DLLP Mask**	RWS	0b
8	**REPLAY_NUM Rollover Mask**	RWS	0b
12	**Replay Timer Timeout Mask**	RWS	0b
13	**Advisory Non-Fatal Error Mask – This bit is Set by default to enable compatibility with software that does not comprehend Role-Based Error Reporting.**	RWS	1b
14	**Corrected Internal Error Mask (Optional)**	RWS	1b
15	**Header Log Overflow Mask (Optional)**	RWS	1b

7.8.4.7 *Advanced Error Capabilities and Control Register (Offset 18h)*

Figure 7-114 details allocation of register fields in the Advanced Error Capabilities and Control register; Table 7-92 provides the respective bit definitions. Handling of multiple errors is discussed in Section 6.2.4.2.

[138] For historical reasons, implementation of this bit is optional. If not implemented, this bit must be RsvdP, and bit 0 of the Correctable Error Status register must also not be implemented. Note that some checking for Receiver Errors is required in all cases (see Sections 4.2.1.1.3, 4.2.4.7, and 4.2.6).

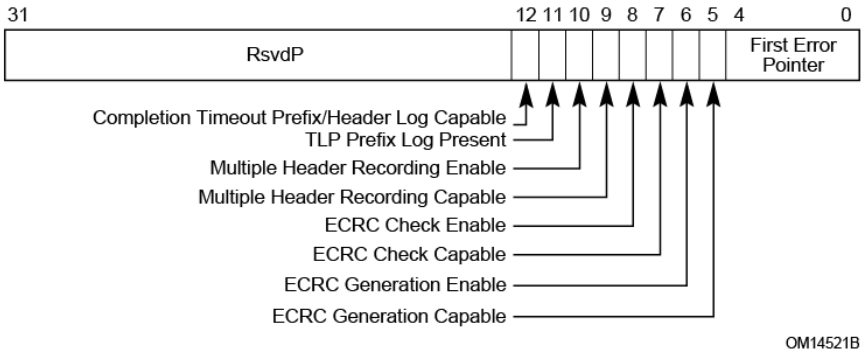


Figure 7-114: Advanced Error Capabilities and Control Register

Table 7-92: Advanced Error Capabilities and Control Register

Bit Location	Register Description	Attributes
4:0	**First Error Pointer** – The First Error Pointer is a field that identifies the bit position of the first error reported in the Uncorrectable Error Status register. Refer to Section 6.2 for further details.	ROS
5	**ECRC Generation Capable** – If Set, this bit indicates that the Function is capable of generating ECRC (see Section 2.7).	RO
6	**ECRC Generation Enable** – When Set, ECRC generation is enabled (see Section 2.7). Functions that do not implement the associated mechanism are permitted to hardwire this bit to 0b. Default value of this bit is 0b.	RWS
7	**ECRC Check Capable** – If Set, this bit indicates that the Function is capable of checking ECRC (see Section 2.7).	RO
8	**ECRC Check Enable** – When Set, ECRC checking is enabled (see Section 2.7). Functions that do not implement the associated mechanism are permitted to hardwire this bit to 0b. Default value of this bit is 0b.	RWS
9	**Multiple Header Recording Capable** – If Set, this bit indicates that the Function is capable of recording more than one error header. Refer to Section 6.2 for further details.	RO
10	**Multiple Header Recording Enable** – When Set, this bit enables the Function to record more than one error header. Functions that do not implement the associated mechanism are permitted to hardwire this bit to 0b. Default value of this bit is 0b.	RWS

Bit Location	Register Description	Attributes
11	**TLP Prefix Log Present** – If Set and the First Error Pointer is valid, indicates that the TLP Prefix Log register contains valid information. If Clear or if First Error Pointer is invalid, the TLP Prefix Log register is undefined. Default value of this bit is 0. This bit is RsvdP if the End-End TLP Prefix Supported bit is Clear.	ROS
12	**Completion Timeout Prefix/Header Log Capable** – If Set, this bit indicates that the Function records the prefix/header of Request TLPs that experience a Completion Timeout error.	RO

7.8.4.8 *Header Log Register (Offset 1Ch)*

The Header Log register contains the header for the TLP corresponding to a detected error; refer to Section 6.2 for further details. Section 6.2 also describes the conditions where the packet header is recorded. This register is 16 bytes and adheres to the format of the headers defined throughout this specification.

The header is captured such that, when read using DW accesses, the fields of the header are laid out in the same way the headers are presented in this document. Therefore, byte 0 of the header is located in byte 3 of the Header Log register, byte 1 of the header is in byte 2 of the Header Log register and so forth. For 12-byte headers, only bytes 0 through 11 of the Header Log register are used and values in bytes 12 through 15 are undefined.

In certain cases where a Malformed TLP is reported, the Header Log register may contain TLP Prefix information. See Section 6.2.4.4 for details.

Figure 7-115 details allocation of register fields in the Header Log register; Table 7-93 provides the respective bit definitions.

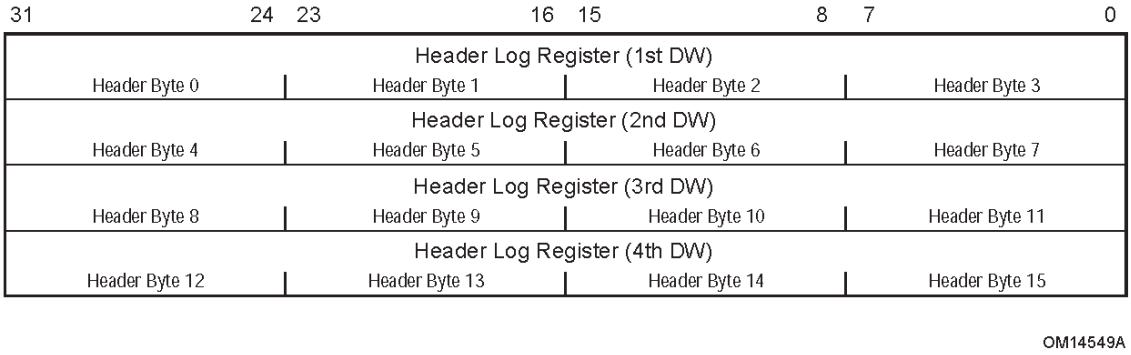


Figure 7-115: Header Log Register

Table 7-93: Header Log Register

Bit Location	Register Description	Attributes	Default
127:0	Header of TLP associated with error	ROS	0

7.8.4.9 *Root Error Command Register (Offset 2Ch)*

The Root Error Command register allows further control of Root Complex response to Correctable, Non-Fatal, and Fatal error Messages than the basic Root Complex capability to generate system errors in response to error Messages (either received or internally generated). Bit fields (see Figure 7-116) enable or disable generation of interrupts (claimed by the Root Port or Root Complex Event Collector) in addition to system error Messages according to the definitions in Table 7-94.

For both Root Ports and Root Complex Event Collectors, in order for a received error Message or an internally generated error Message to generate an interrupt enabled by this register, the error Message must be enabled for "transmission" by the Root Port or Root Complex Event Collector (see Section 6.2.4.1 and Section 6.2.8.1).

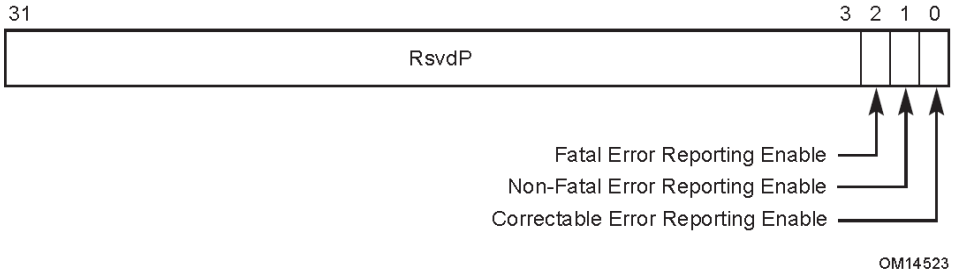


Figure 7-116: Root Error Command Register

Table 7-94: Root Error Command Register

Bit Location	Register Description	Attributes	Default
0	**Correctable Error Reporting Enable** – When Set, this bit enables the generation of an interrupt when a correctable error is reported by any of the Functions in the Hierarchy Domain associated with this Root Port. Root Complex Event Collectors provide support for the above described functionality for RCiEPs. Refer to Section 6.2 for further details.	RW	0b

Bit Location	Register Description	Attributes	Default
1	**Non-Fatal Error Reporting Enable** – When Set, this bit enables the generation of an interrupt when a Non-fatal error is reported by any of the Functions in the Hierarchy Domain associated with this Root Port. Root Complex Event Collectors provide support for the above described functionality for RCiEPs. Refer to Section 6.2 for further details.	RW	0b
2	**Fatal Error Reporting Enable** – When Set, this bit enables the generation of an interrupt when a Fatal error is reported by any of the Functions in the Hierarchy Domain associated with this Root Port. Root Complex Event Collectors provide support for the above described functionality for RCiEPs. Refer to Section 6.2 for further details.	RW	0b

System error generation in response to PCI Express error Messages may be turned off by system software using the PCI Express Capability structure described in Section 7.5.2 when advanced error reporting via interrupts is enabled. Refer to Section 6.2 for further details.

7.8.4.10 *Root Error Status Register (Offset 30h)*

The Root Error Status register reports status of error Messages (ERR_COR, ERR_NONFATAL, and ERR_FATAL) received by the Root Port, and of errors detected by the Root Port itself (which are treated conceptually as if the Root Port had sent an error Message to itself). In order to update this register, error Messages received by the Root Port and/or internally generated error Messages must be enabled for "transmission" by the primary interface of the Root Port. ERR_NONFATAL and ERR_FATAL Messages are grouped together as uncorrectable. Each correctable and uncorrectable (Non-fatal and Fatal) error source has a first error bit and a next error bit associated with it respectively. When an error is received by a Root Complex, the respective first error bit is Set and the Requester ID is logged in the Error Source Identification register. A Set individual error status bit indicates that a particular error category occurred; software may clear an error status by writing a 1b to the respective bit. If software does not clear the first reported error before another error Message is received of the same category (correctable or uncorrectable), the corresponding next error status bit will be set but the Requester ID of the subsequent error Message is discarded. The next error status bits may be cleared by software by writing a 1b to the respective bit as well. Refer to Section 6.2 for further details. This register is updated regardless of the settings of the Root Control register and the Root Error Command register. Figure 7-117 details allocation of register fields in the Root Error Status register; Table 7-95 provides the respective bit definitions. Root Complex Event Collectors provide support for the above-described functionality for RCiEPs (and for the Root Complex Event Collector itself). In order to update this register, error Messages received by the Root Complex Event Collector from its associated RCiEPs and/or internally generated error Messages must be enabled for "transmission" by the Root Complex Event Collector.

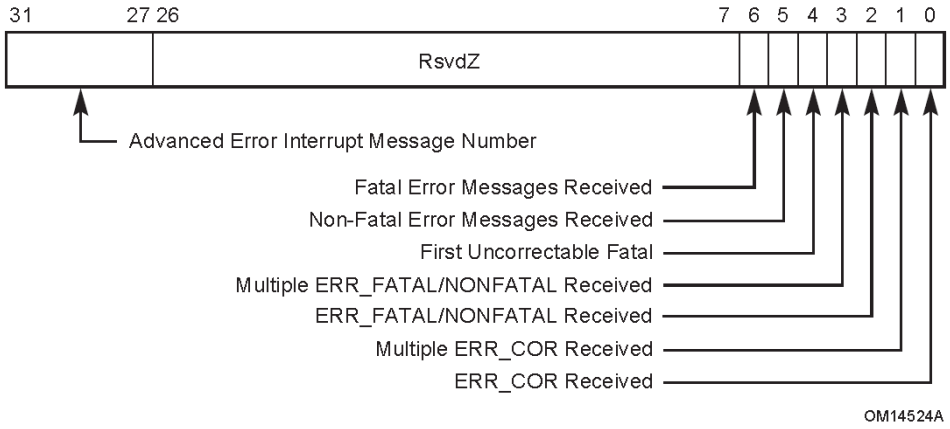


Figure 7-117: Root Error Status Register

Table 7-95: Root Error Status Register

Bit Location	Register Description	Attributes
0	**ERR_COR Received** – Set when a Correctable error Message is received and this bit is not already Set. Default value of this bit is 0b.	RW1CS
1	**Multiple ERR_COR Received** – Set when a Correctable error Message is received and ERR_COR Received is already Set. Default value of this bit is 0b.	RW1CS
2	**ERR_FATAL/NONFATAL Received** – Set when either a Fatal or a Non-fatal error Message is received and this bit is not already Set. Default value of this bit is 0b.	RW1CS
3	**Multiple ERR_FATAL/NONFATAL Received** – Set when either a Fatal or a Non-fatal error is received and **ERR_FATAL/NONFATAL Received** is already Set. Default value of this bit is 0b.	RW1CS
4	**First Uncorrectable Fatal** – Set when the first Uncorrectable error Message received is for a Fatal error. Default value of this field is 0b.	RW1CS
5	**Non-Fatal Error Messages Received** – Set when one or more Non-Fatal Uncorrectable error Messages have been received. Default value of this bit is 0b.	RW1CS

Bit Location	Register Description	Attributes
6	**Fatal Error Messages Received** – Set when one or more Fatal Uncorrectable error Messages have been received. Default value of this bit is 0b.	RW1CS
31:27	**Advanced Error Interrupt Message Number** – This register indicates which MSI/MSI-X vector is used for the interrupt message generated in association with any of the status bits of this Capability. For MSI, the value in this register indicates the offset between the base Message Data and the interrupt message that is generated. Hardware is required to update this field so that it is correct if the number of MSI Messages assigned to the Function changes when software writes to the Multiple Message Enable field in the MSI Message Control register. For MSI-X, the value in this register indicates which MSI-X Table entry is used to generate the interrupt message. The entry must be one of the first 32 entries even if the Function implements more than 32 entries. For a given MSI-X implementation, the entry must remain constant. If both MSI and MSI-X are implemented, they are permitted to use different vectors, though software is permitted to enable only one mechanism at a time. If MSI-X is enabled, the value in this register must indicate the vector for MSI-X. If MSI is enabled or neither is enabled, the value in this register must indicate the vector for MSI. If software enables both MSI and MSI-X at the same time, the value in this register is undefined.	RO

7.8.4.11 *Error Source Identification Register (Offset 34h)*

The Error Source Identification register identifies the source (Requester ID) of first correctable and uncorrectable (Non-fatal/Fatal) errors reported in the Root Error Status register. Refer to Section 6.2 for further details. This register is updated regardless of the settings of the Root Control register and the Root Error Command register. Figure 7-118 details allocation of register fields in the Error Source Identification register; Table 7-96 provides the respective bit definitions.

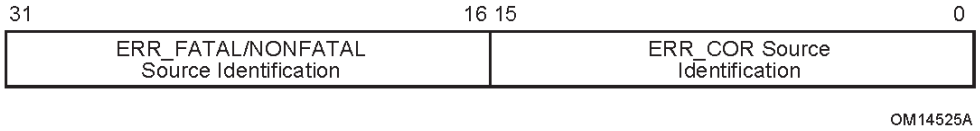


Figure 7-118: Error Source Identification Register

Table 7-96: Error Source Identification Register

Bit Location	Register Description	Attributes
15:0	**ERR_COR Source Identification** – Loaded with the Requester ID indicated in the received ERR_COR Message when the ERR_COR Received bit is not already set. Default value of this field is 0000h.	ROS
31:16	**ERR_FATAL/NONFATAL Source Identification** – Loaded with the Requester ID indicated in the received ERR_FATAL or ERR_NONFATAL Message when the ERR_FATAL/NONFATAL Received bit is not already set. Default value of this field is 0000h.	ROS

7.8.4.12 *TLP Prefix Log Register (Offset 38h)*

The TLP Prefix Log register captures the End-End TLP Prefix(s) for the TLP corresponding to the detected error; refer to Section 6.2 for further details. The TLP Prefix Log register is only meaningful when the TLP Prefix Log Present bit is Set (see Section 7.8.4.7).

The TLP Prefixes are captured such that, when read using DW accesses, the fields of the TLP Prefix are laid out in the same way the fields of the TLP Prefix are described. Therefore, byte 0 of a TLP Prefix is located in byte 3 of the associated TLP Prefix Log register; byte 1 of a TLP Prefix is located in byte 2; and so forth.

The First TLP Prefix Log register contains the first End-End TLP Prefix from the TLP (see Section 6.2.4.4). The Second TLP Prefix Log register contains the second End-End TLP Prefix and so forth. If the TLP contains fewer than four End-End TLP Prefixes, the remaining TLP Prefix Log registers contain zero. A TLP that contains more End-End TLP Prefixes than are indicated by the Function's Max End-End TLP Prefixes field must be handled as an error (see Section 2.2.10.2 for specifics). To allow software to detect this condition, the supported number of End-End TLP Prefixes are logged in this register, the first overflow End-End TLP Prefix is logged in the first DW of the Header Log register and the remaining DWs of the Header Log register are undefined (see Section 6.2.4.4).

The TLP Prefix Log registers beyond the number supported by the Function are hardwired to zero. For example, if a Functions, Max End-End TLP Prefixes field contains 10b (indicating 2 DW of buffering) then the third and fourth TLP Prefix Log registers are hardwired to zero. If the End-End TLP Prefix Supported bit (Section 7.5.3.15) is Clear, the TLP Prefix Log register is not required to be implemented.

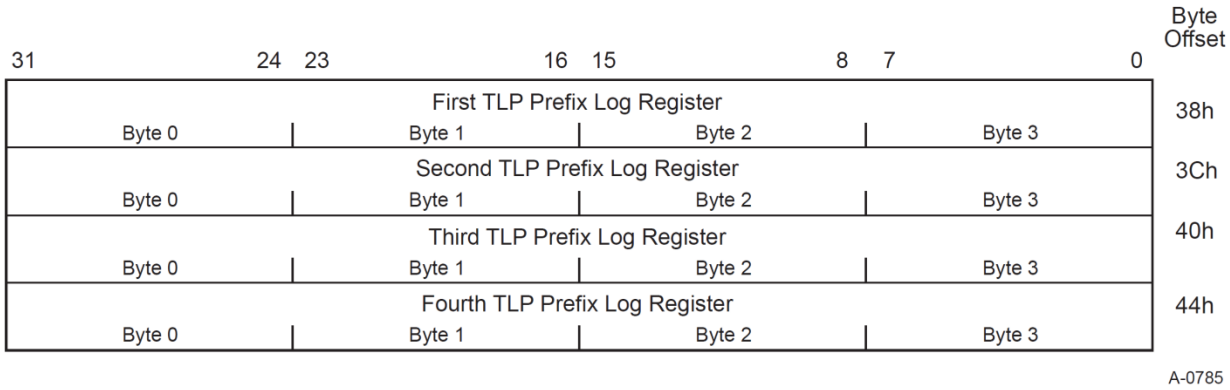


Figure 7-119: TLP Prefix Log Register

Table 7-97: TLP Prefix Log Register

Bit Location	Register Description	Attributes	Default
127:0	TLP Prefix Log	ROS	0

7.8.5 Enhanced Allocation (EA) Capability Structure

Each function that supports the Enhanced Allocation mechanism must implement the Enhanced Allocation capability structure.

Each field is defined in the following sections. Reserved registers must return 0 when read and write operations must have no effect. Read-only registers return valid data when read, and write operations must have no effect.

7.8.5.1 *Enhanced Allocation Capability First DW (Offset 00h)*

The first DW of the Enhanced Allocation capability is illustrated in Figure 7-120, and is documented in Table 7-98.

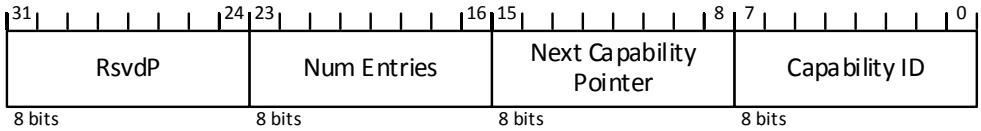


Figure 7-120: First DW of Enhanced Allocation Capability

Table 7-98: First DW of Enhanced Allocation Capability

Bit Location	Register Description	Attributes
7:0	**Capability ID** – Must be set to 14h to indicate Enhanced Allocation capability. This field is read only.	HwInit
15:8	**Next Capability Pointer –** Pointer to the next item in the capabilities list. Must be NULL for the final item in the list. This field is read only.	HwInit

Bit Location	Register Description	Attributes
21:16	**Num Entries –** Number of entries following the first DW of the capability. Value of 00 0000b is permitted and means there are no entries. This field is read only.	HwInit

7.8.5.2 *Enhanced Allocation Per-Entry Format*

The first DW of each entry in the Enhanced Allocation capability is illustrated in Figure 7-121, and is defined in Table 7-99.

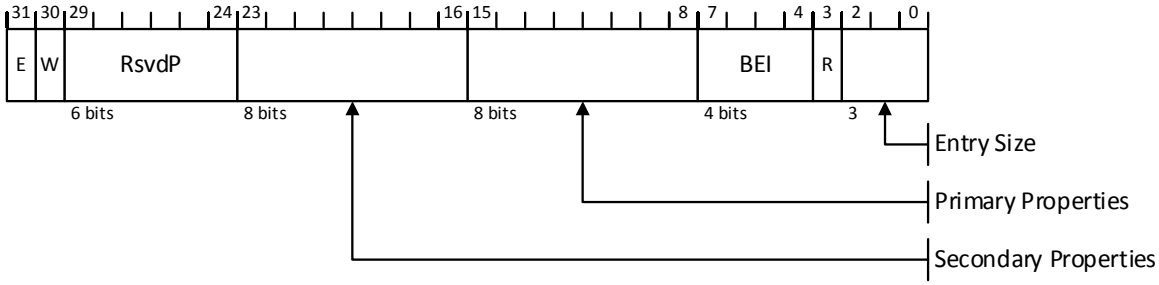


Figure 7-121: First DW of Each Entry for Enhanced Allocation Capability

Table 7-99: First DW of Each Entry for Enhanced Allocation Capability

Bit Location	Register Description	Attributes
2:0	**Entry Size (ES) –** Number of DW following the initial DW in this entry. When processing this capability, software is required to use the value in this field to determine the size of this entry, and if this entry is not the final entry, the start of the following entry in the capability. This requirement must be strictly followed by software, even if the indicated entry size does not correspond to any entry defined in this specification. Value of 000b indicates only the first DW (containing the Entry Size field) is included in the entry.	HwInit

<table>
<tr><th>Bit Location</th><th>Register Description</th><th>Attributes</th></tr>
<tr><td>7:4</td><td>BAR Equivalent Indicator (BEI) – This field indicates the equivalent BAR for this entry.

Specific rules for use of this field are given in the text following this table.
<table>
<tr><th>BEI Value</th><th>Description</th></tr>
<tr><td>0</td><td>Entry is equivalent to BAR at location 10h</td></tr>
<tr><td>1</td><td>Entry is equivalent to BAR at location 14h</td></tr>
<tr><td>2</td><td>Entry is equivalent to BAR at location 18h</td></tr>
<tr><td>3</td><td>Entry is equivalent to BAR at location 1Ch</td></tr>
<tr><td>4</td><td>Entry is equivalent to BAR at location 20h</td></tr>
<tr><td>5</td><td>Entry is equivalent to BAR at location 24h</td></tr>
<tr><td>6</td><td>Permitted to be used by a Function with a Type 1 Configuration Space header only, optionally used to indicate a resource that is located behind the Function</td></tr>
<tr><td>7</td><td>Equivalent Not Indicated</td></tr>
<tr><td>8</td><td>Expansion ROM Base Address</td></tr>
<tr><td>9-14</td><td>Entry relates to VF BARs 0-5 respectively</td></tr>
<tr><td>15</td><td>Reserved – Software must treat values in this range as “Equivalent Not Indicated”</td></tr>
</table></td><td>HwInit</td></tr>
<tr><td>15:8</td><td>Primary Properties – Indicates the entry properties as defined in Table 7-100.</td><td>HwInit</td></tr>
<tr><td>23:16</td><td>Secondary Properties – Optionally used to indicate a different but compatible entry property, using properties as defined in Table 7-100.</td><td>HwInit</td></tr>
<tr><td>30</td><td>Writable (W) – The value 1b indicates that the Base and MaxOffset fields for this entry are RW and that the Field Size bits for this entry are either RW or HwInit. The value 0b indicates those fields are HwInit.
See Table 7-100 for additional requirements on the value of this field.</td><td>HwInit</td></tr>
</table>

Bit Location	Register Description	Attributes
31	**Enable (E)** – 1b indicates this entry is enabled, 0b indicates this entry is disabled. If system software disables this entry, the resource indicated must still be associated with this function, and it is not permitted to reallocate this resource to any other entity. This field is permitted to be implemented as HwInit for functions that require the allocation of the associated resource, or as RW for functions that can allow system software to disable this resource, for example if BAR mechanisms are to be used instead of this resource.	RW/HwInit

Rules for use of BEI field:

- A Type 0 Function is permitted to use EA to allocate resources for itself, and such resources must indicate a BEI value of 0-5, 7 or 8.
- A Physical Function (Type 0 Function that supports SR-IOV) is permitted to use EA to allocate resources for its associated Virtual Functions, and such resources must indicate a BEI value of 9-14.
- A Type 1 (bridge) function is permitted to use EA to allocate resources for itself, and such resources must indicate a BEI value of 0, 1 or 7.
- A Type 1 function is permitted but not required to indicate resources mapped behind that Function, but if such resources are indicated by the Type 1 function, the entry must indicate a BEI value of 6.
- For a 64-bit Base Address register, the BEI indicates the equivalent BAR location for lower DWORD.
- For Memory BARs where the Primary or Secondary Properties is 00h or 01h, it is permitted to assign the same BEI in the range of 0 to 5 once for a range where Base + MaxOffset is below 4GB, and again for a range where Base + MaxOffset is greater than 4GB; It is not otherwise permitted to assign the same BEI in the range 0 to 5 for more than one entry.
- For Virtual Function BARs where the Primary or Secondary Properties is 03h or 04h it is permitted to assign the same BEI in the range of 9 to 14 once for a range where Base + MaxOffset is below 4GB, and again for a range where Base + MaxOffset is greater than 4GB; It is not otherwise permitted to assign the same BEI in the range 9 to 14 for more than one VF entry.
- For all cases where two entries with the same BEI are permitted, Software must enable use of only one of the two ranges at a time for a given Function.
- It is permitted for an arbitrary number of entries to assign a BEI of 6 or 7.

- At most one entry is permitted with a BEI of 8. If such an entry is present, the Expansion ROM Base Address Register must be hardwired to 0.
- For Type 1 functions, BEI values 2 through 5 are reserved.

Figure 7-122 illustrates the format of a complete Enhanced Allocation entry for a Type 0 function. For the Base and MaxOffset fields, bit 1 indicates if the field is a 32b (0) or 64b (1) field.

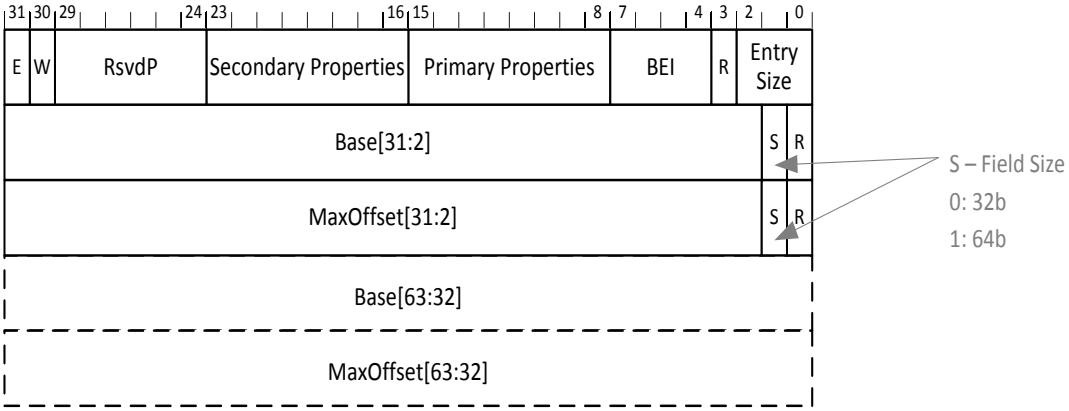


Figure 7-122: Format of Entry for Enhanced Allocation Capability

The value in the Base field ([63:2] or [31:2]) indicates the DW address of the start of the resource range. Bits [1:0] of the address are not included in the Base field, and must always be interpreted as 00b.

The value in the Base field plus the value in the MaxOffset field ([63:2] or [31:2]) indicates the address of the last included DW of the resource range. Bits [1:0] of the MaxOffset are not included in the MaxOffset field, and must always be interpreted as 11b.

For the Base and MaxOffset fields, when bits [63:32] are not provided then those bits must be interpreted as all 0's.

Although it is permitted for a Type 0 Function to indicate the use of a range that is not naturally aligned and/or not a power of two in size, some system software may fail if this is done. Particularly for ranges that are mapped to legacy BARs by indicating a BEI in the range of 0 to 5, it is strongly recommended that the Base and MaxOffset fields for a Type 0 Function indicate a naturally aligned region.

The Primary Properties[7:0] field must be set by hardware to identify the type of resource indicated by the entry. It is strongly recommended that hardware set the Secondary Properties[7:0] to indicate an alternate resource type which can be used by software when the Primary Properties[7:0] field value is not comprehended by that software, for example when older system software is used with new hardware that implements resources using a value for Primary Properties that was reserved at the time the older system software was implemented. When this is done, hardware must ensure that software operating using the resource according to the value indicated in the Secondary Properties field will operate in a functionally correct way, although it is not required that this operation will result in optimal system performance or behavior.

The Primary Properties[7:0] and Secondary Properties[7:0] fields are defined in Table 7-100. This table also defines whether or not the entry is permitted to be writeable. The Writeable bit in any entry must be 0b unless both the Primary and Secondary properties of that entry allow otherwise.

Table 7-100: Enhanced Allocation Entry Field Value Definitions for both the Primary Properties and Secondary Properties Fields

Value (h)	Resource Definition	Writeable permitted
00	Memory Space, Non-Prefetchable.	No
01	Memory Space, Prefetchable.	No
02	I/O Space.	No
03	For use only by Physical Functions to indicate resources for Virtual Function use, Memory Space, Prefetchable.	No
04	For use only by Physical Functions to indicate resources for Virtual Function use, Memory Space, Non-Prefetchable.	No
05	For use only by Type 1 Functions to indicate Memory, Non-Prefetchable, for Allocation Behind that Bridge.	No
06	For use only by Type 1 Functions to indicate Memory, Prefetchable, for Allocation Behind that Bridge.	No
07	For use only by Type 1 Functions to indicate I/O Space for Allocation Behind that Bridge.	No
08-FC	Reserved for future use; System firmware/software must not write to this entry, and must not attempt to interpret this entry or to use this resource. When software reads a Primary Properties value that is within this range, is it strongly recommended that software treat this resource according to the value in the Secondary Properties field, if that field contains a non-reserved value.	Yes
FD	Memory Space Resource Unavailable For Use - – System firmware/software must not write to this entry, and must not attempt to use the resource described by this entry for any purpose.	No
FE	I/O Space Resource Unavailable For Use - – System firmware/software must not write to this entry, and must not attempt to use the resource described by this entry for any purpose.	No

Value (h)	Resource Definition	Writeable permitted
FF	Entry Unavailable For Use – System firmware/software must not write to this entry, and must not attempt to interpret this entry as indicating any resource. It is strongly recommended that hardware use this value in the Secondary Properties field to indicate that for proper operation, the hardware requires the use of the resource definition indicated in the Primary Properties field .	No

The following figures illustrate the layout of Enhanced Allocation entries for various cases.

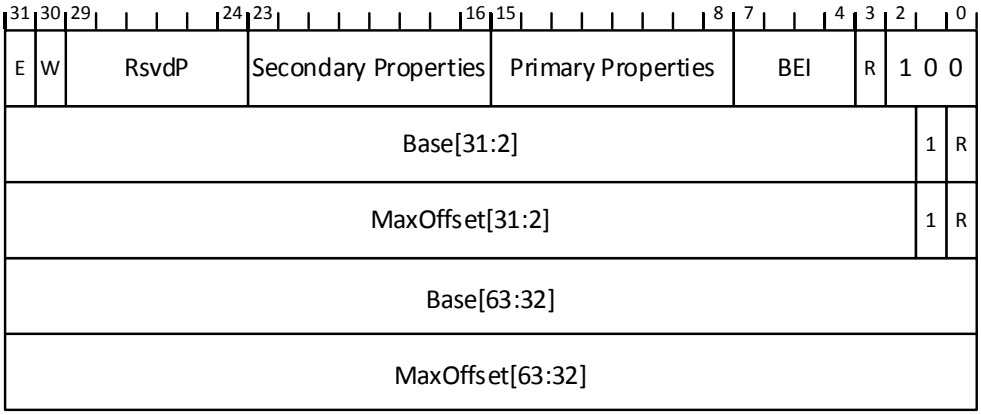


Figure 7-123: Example Entry with 64b Base and 64b MaxOffset

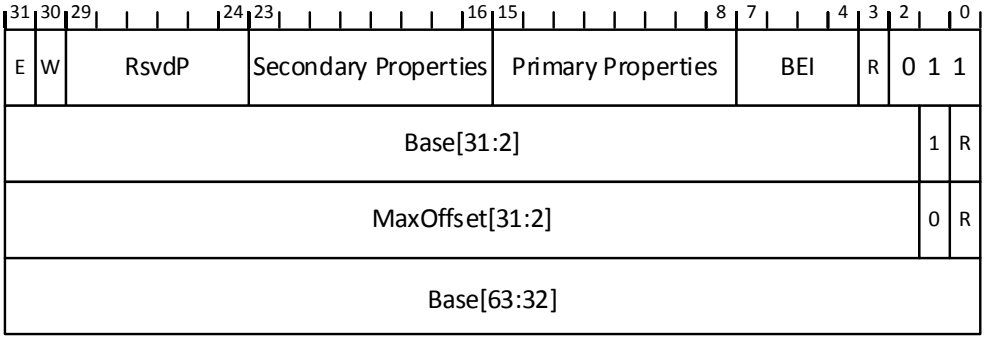


Figure 7-124: Example Entry with 64b Base and 32b MaxOffset

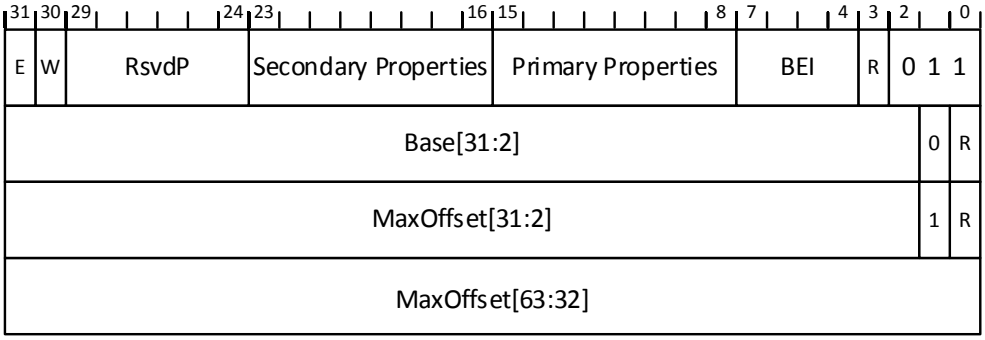


Figure 7-125: Example Entry with 32b Base and 64b MaxOffset

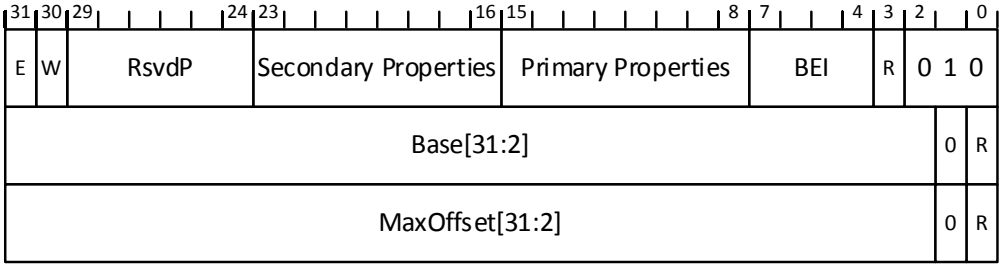


Figure 7-126: Example Entry with 32b Base and 32b MaxOffset

7.8.6 Resizable BAR Capability

The Resizable BAR Capability is an optional capability that allows hardware to communicate resource sizes, and system software, after determining the optimal size, to communicate this optimal size back to the hardware. Hardware communicates the resource sizes that are acceptable for operation via the Resizable BAR Capability and Control registers. Hardware must support at least one size in the range from 1 MB to 512 GB.

Implementation NOTE

Resizable BAR Backward Compatibility With Software

The Resizable BAR capability initially supported 20 sizes, ranging from 1 MB to 512 GB, and was later expanded with 16 larger sizes. The hardware requirement to support at least one of the initial sizes ensures backward compatibility with software that comprehends only the initial sizes.

Software determines, through a proprietary mechanism, what the optimal size is for the resource, and programs that size via the BAR Size field of the Resizable BAR Control register. Hardware immediately reflects the size inference in the read-only bits of the appropriate Base Address register. Hardware must Clear any bits that change from RW to read-only, so that subsequent reads return zero. Software must clear the Memory Space Enable bit in the Command register before writing the BAR Size field. After writing the BAR Size field, the contents of the corresponding BAR are undefined. To ensure that it contains a valid address after resizing the BAR, system software must reprogram the BAR, and Set the Memory Space Enable bit (unless the resource is not allocated).

The Resizable BAR Capability and Control registers are permitted to indicate the ability to operate at 4 GB or greater only if the associated BAR is a 64-bit BAR.

This capability is applicable to Functions that have Base Address registers only. It is strongly recommended that a Function not advertise any supported BAR sizes that are larger than the space it would effectively utilize if allocated.

IMPLEMENTATION NOTE

Using the Capability During Resource Allocation

System software that allocates resources can use this capability to resize the resources inferred by the Function's BAR's read-only bits. Previous versions of this software determined the resource size by writing FFFFh to the BAR, reading back the value, and determining the size by the number of bits that are Set. Following this, the base address is written to the BAR.

System software uses this capability in place of the above mentioned method of determining the resource size, and prior to assigning the base address to the BAR. Potential usable resource sizes are reported by the Function via its Resizable BAR Capability and Control registers. It is intended that the software allocate the largest of the reported sizes that it can, since allocating less address space than the largest reported size can result in lower performance. Software then writes the size to the Resizable BAR Control register for the appropriate BAR for the Function. Following this, the base address is written to the BAR.

For interoperability reasons, it is possible that hardware will set the default size of the BAR to a low size; that is, a size lower than the largest reported in the Resizable BAR Capability and Control registers. Software that does not use this capability to size resources will likely result in sub-optimal resource allocation, where the resources are smaller than desirable, or not allocatable because there is no room for them.

With the Resizable BAR capability, the amount of address space consumed by a device can change. In a resource constrained environment, the allocation of more address space to a device may result in allocation of less of the address space to other memory-mapped hardware, like system RAM. System software responsible for allocating resources in this kind of environment is recommended to distribute the limited address space appropriately.

The Resizable BAR Capability structure defines a PCI Express Extended Capability, which is located in PCI Express Extended Configuration Space, that is, above the first 256 bytes, and is shown below in Figure 7-127. This structure allows devices with this capability to be identified and controlled. A Capability and a Control register is implemented for each BAR that is resizable. Since a maximum of six BARs may be implemented by any Function, the Resizable BAR Capability structure can range from 12 bytes long (for a single BAR) to 52 bytes long (for all six BARs).

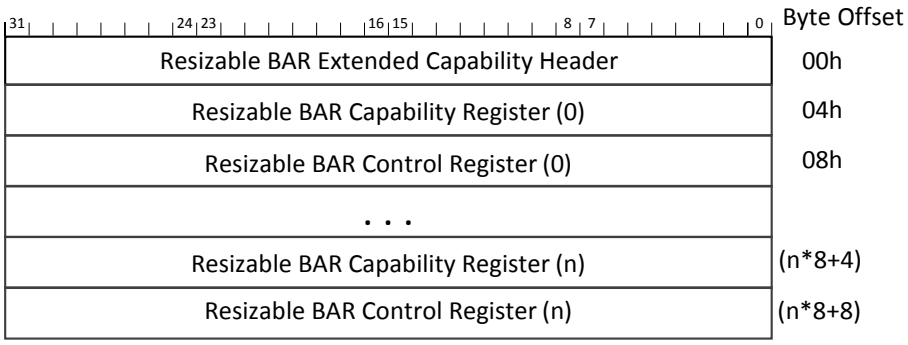


Figure 7-127: Resizable BAR Capability

7.8.6.1 *Resizable BAR Extended Capability Header (Offset 00h)*

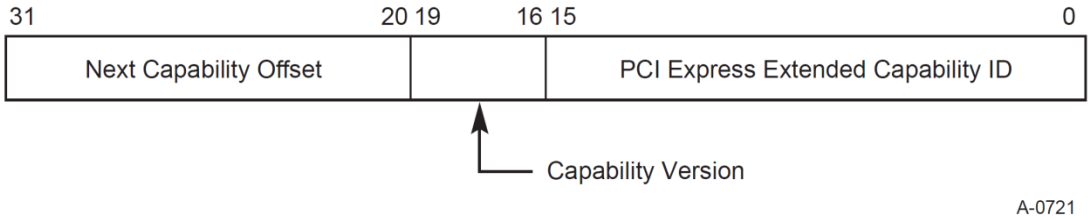


Figure 7-128: Resizable BAR Extended Capability Header

Table 7-101: Resizable BAR Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the extended capability. The PCI Express Extended Capability ID for the Resizable BAR Capability is 0015h.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the capability structure present. Must be 1h for this version of the specification.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express Extended Capability structure or 000h if no other items exist in the linked list of Capabilities.	RO

7.8.6.2 *Resizable BAR Capability Register*

For backward compatibility with software, hardware must Set at least one bit in the range from 4 to 23. See the associated Implementation Note in Section 7.8.6.

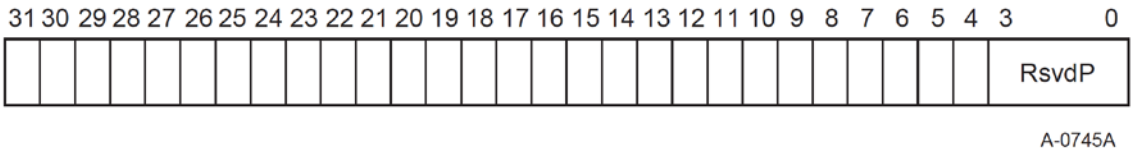


Figure 7-129: Resizable BAR Capability Register

Table 7-102: Resizable BAR Capability Register

Bit Location	Register Description	Attributes
4	When Set, indicates that the Function supports operating with the BAR sized to 1 MB (2^{20} bytes)	RO
5	When Set, indicates that the Function supports operating with the BAR sized to 2 MB (2^{21} bytes)	RO
6	When Set, indicates that the Function supports operating with the BAR sized to 4 MB (2^{22} bytes)	RO
7	When Set, indicates that the Function supports operating with the BAR sized to 8 MB (2^{23} bytes)	RO
8	When Set, indicates that the Function supports operating with the BAR sized to 16 MB (2^{24} bytes)	RO
9	When Set, indicates that the Function supports operating with the BAR sized to 32 MB (2^{25} bytes)	RO
10	When Set, indicates that the Function supports operating with the BAR sized to 64 MB (2^{26} bytes)	RO
11	When Set, indicates that the Function supports operating with the BAR sized to 128 MB (2^{27} bytes)	RO
12	When Set, indicates that the Function supports operating with the BAR sized to 256 MB (2^{28} bytes)	RO
13	When Set, indicates that the Function supports operating with the BAR sized to 512 MB (2^{29} bytes)	RO
14	When Set, indicates that the Function supports operating with the BAR sized to 1 GB (2^{30} bytes)	RO
15	When Set, indicates that the Function supports operating with the BAR sized to 2 GB (2^{31} bytes)	RO
16	When Set, indicates that the Function supports operating with the BAR sized to 4 GB (2^{32} bytes)	RO
17	When Set, indicates that the Function supports operating with the BAR sized to 8 GB (2^{33} bytes)	RO
18	When Set, indicates that the Function supports operating with the BAR sized to 16 GB (2^{34} bytes)	RO
19	When Set, indicates that the Function supports operating with the BAR sized to 32 GB (2^{35} bytes)	RO
20	When Set, indicates that the Function supports operating with the BAR sized to 64 GB (2^{36} bytes)	RO

Bit Location	Register Description	Attributes
21	When Set, indicates that the Function supports operating with the BAR sized to 128 GB (2^{37} bytes)	RO
22	When Set, indicates that the Function supports operating with the BAR sized to 256 GB (2^{38} bytes)	RO
23	When Set, indicates that the Function supports operating with the BAR sized to 512 GB (2^{39} bytes)	RO
24	When Set, indicates that the Function supports operating with the BAR sized to 1 TB (2^{40} bytes)	RO
25	When Set, indicates that the Function supports operating with the BAR sized to 2 TB (2^{41} bytes)	RO
26	When Set, indicates that the Function supports operating with the BAR sized to 4 TB (2^{42} bytes)	RO
27	When Set, indicates that the Function supports operating with the BAR sized to 8 TB (2^{43} bytes)	RO
28	When Set, indicates that the Function supports operating with the BAR sized to 16 TB (2^{44} bytes)	RO
29	When Set, indicates that the Function supports operating with the BAR sized to 32 TB (2^{45} bytes)	RO
30	When Set, indicates that the Function supports operating with the BAR sized to 64 TB (2^{46} bytes)	RO
31	When Set, indicates that the Function supports operating with the BAR sized to 128 TB (2^{47} bytes)	RO

7.8.6.3 *Resizable BAR Control Register*

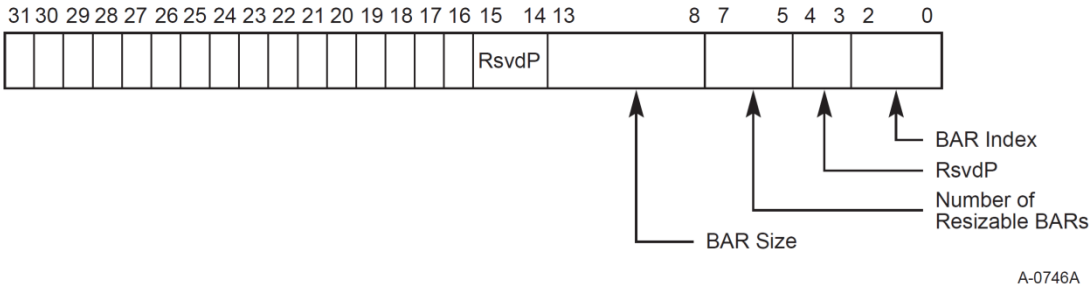


Figure 7-130: Resizable BAR Control Register

Table 7-103: Resizable BAR Control Register

Bit Location	Register Description	Attributes

Bit Location	Register Description	Attributes
2:0	**BAR Index** – This encoded value points to the beginning of the BAR. 0 = BAR located at offset 10h 1 = BAR located at offset 14h 2 = BAR located at offset 18h 3 = BAR located at offset 1Ch 4 = BAR located at offset 20h 5 = BAR located at offset 24h All other encodings are Reserved. For a 64-bit Base Address register, the BAR Index indicates the lower DWORD. This value indicates which BAR supports a negotiable size.	RO
7:5	**Number of Resizable BARs** – Indicates the total number of resizable BARs in the capability structure for the Function. See Figure 7-127. The value of this field must be in the range of 01h to 06h. The field is valid in Resizable BAR Control register (0) (at offset 008h), and is RsvdP for all others.	RO/RsvdP
13:8	**BAR Size** – This is an encoded value. 0 = 1 MB (2^{20} bytes) 1 = 2 MB (2^{21} bytes) 2 = 4 MB (2^{22} bytes) 3 = 8 MB (2^{23} bytes) … 43 = 8 EB (2^{63} bytes) The default value of this field is equal to the default size of the address space that the BAR resource is requesting via the BAR's read-only bits. For backward compatibility with software, the default value must be in the range from 0 to 19. When this register field is programmed, the value is immediately reflected in the size of the resource, as encoded in the number of read-only bits in the BAR. Software must only write values that correspond to those indicated as supported in the Resizable BAR Capability and Control registers. Writing an unsupported value will produce undefined results. BAR Size bits that never need to be Set in order to indicate every supported size are permitted to be hardwired to 0.	RW
16	When Set, indicates that the Function supports operating with the BAR sized to 256 TB (2^{48} bytes)	RO
17	When Set, indicates that the Function supports operating with the BAR sized to 512 TB (2^{49} bytes)	RO
18	When Set, indicates that the Function supports operating with the BAR sized to 1 PB (2^{50} bytes)	RO

Bit Location	Register Description	Attributes
19	When Set, indicates that the Function supports operating with the BAR sized to 2 PB (2^{51} bytes)	RO
20	When Set, indicates that the Function supports operating with the BAR sized to 4 PB (2^{52} bytes)	RO
21	When Set, indicates that the Function supports operating with the BAR sized to 8 PB (2^{53} bytes)	RO
22	When Set, indicates that the Function supports operating with the BAR sized to 16 PB (2^{54} bytes)	RO
23	When Set, indicates that the Function supports operating with the BAR sized to 32 PB (2^{55} bytes)	RO
24	When Set, indicates that the Function supports operating with the BAR sized to 64 PB (2^{56} bytes)	RO
25	When Set, indicates that the Function supports operating with the BAR sized to 128 PB (2^{57} bytes)	RO
26	When Set, indicates that the Function supports operating with the BAR sized to 256 PB (2^{58} bytes)	RO
27	When Set, indicates that the Function supports operating with the BAR sized to 512 PB (2^{59} bytes)	RO
28	When Set, indicates that the Function supports operating with the BAR sized to 1 EB (2^{60} bytes)	RO
29	When Set, indicates that the Function supports operating with the BAR sized to 2 EB (2^{61} bytes)	RO
30	When Set, indicates that the Function supports operating with the BAR sized to 4 EB (2^{62} bytes)	RO
31	When Set, indicates that the Function supports operating with the BAR sized to 8 EB (2^{63} bytes)	RO

7.8.7 ARI Capability

ARI is an optional capability. This capability must be implemented by each Function in an ARI Device. It is not applicable to a Root Port, a Switch Downstream Port, an RCiEP, or a Root Complex Event Collector.

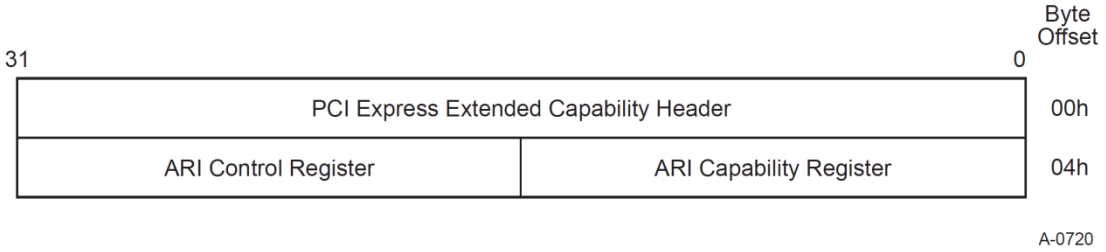


Figure 7-131: ARI Capability

7.8.7.1 ARI Capability Header (Offset 00h)

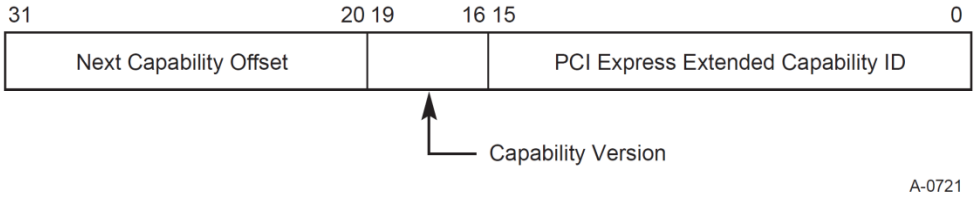


Figure 7-132: ARI Capability Header

Table 7-104: ARI Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the extended capability. PCI Express Extended Capability ID for the ARI Capability is 000Eh.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the capability structure present. Must be 1h for this version of the specification.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express Extended Capability structure or 000h if no other items exist in the linked list of Capabilities.	RO

7.8.7.2 ARI Capability Register (Offset 04h)

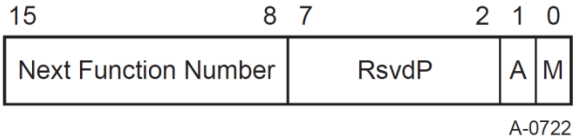


Figure 7-133: ARI Capability Register

Table 7-105: ARI Capability Register

Bit Location	Register Description	Attributes
0	**MFVC Function Groups Capability (M)** – Applicable only for Function 0; must be 0b for all other Functions. If 1b, indicates that the ARI Device supports Function Group level arbitration via its Multi-Function Virtual Channel (MFVC) Capability structure.	RO
1	**ACS Function Groups Capability (A)** – Applicable only for Function 0; must be 0b for all other Functions. If 1b, indicates that the ARI Device supports Function Group level granularity for ACS P2P Egress Control via its ACS Capability structures.	RO

Bit Location	Register Description	Attributes
15:8	**Next Function Number** – This field indicates the Function Number of the next higher numbered Function in the Device, or 00h if there are no higher numbered Functions. Function 0 starts this linked list of Functions.	RO

7.8.7.3 *ARI Control Register (Offset 06h)*

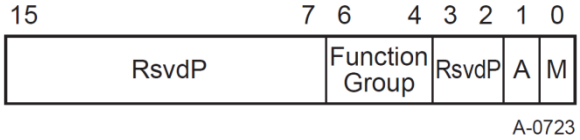


Figure 7-134: ARI Control Register

Table 7-106: ARI Control Register

Bit Location	Register Description	Attributes
0	**MFVC Function Groups Enable (M)** – Applicable only for Function 0; must be hardwired to 0b for all other Functions. When set, the ARI Device must interpret entries in its Function Arbitration Table as Function Group Numbers rather than Function Numbers. Default value of this bit is 0b. Must be hardwired to 0b if the MFVC Function Groups Capability bit is 0b.	RW
1	**ACS Function Groups Enable (A)** – Applicable only for Function 0; must be hardwired to 0b for all other Functions. When set, each Function in the ARI Device must associate bits within its Egress Control Vector with Function Group Numbers rather than Function Numbers. Default value of this bit is 0b. Must be hardwired to 0b if the ACS Function Groups Capability bit is 0b.	RW
6:4	**Function Group** – Assigns a Function Group Number to this Function. Default value of this field is 000b. Must be hardwired to 000b if in Function 0, the MFVC Function Groups Capability bit and ACS Function Groups Capability bit are both 0b.	RW

7.8.8 PASID Extended Capability Structure

The presence of a PASID Extended Capability indicates that the Endpoint supports sending and receiving TLPs containing a PASID TLP Prefix. Separate support and enables are provided for the various optional features.

This capability is applicable to Endpoints and RCiEPs. For Root Ports, support and control is outside the scope of this specification.

This capability is independent of both the ATS and PRI features defined in Chapter 10. Endpoints that contain a PASID Extended Capability need not support ATS or PRI. Endpoints that support ATS or PRI need not support PASID.

Figure 7-135 details allocation of the register bits in the PASID Extended Capability structure.

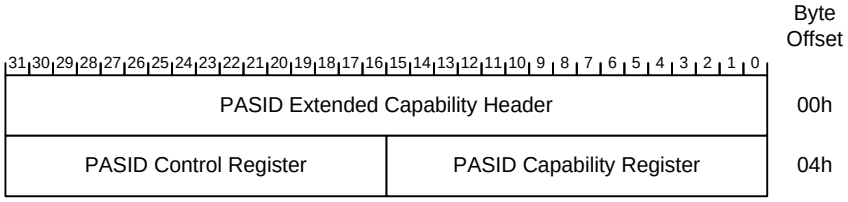


Figure 7-135: PASID Extended Capability Structure

7.8.8.1 *PASID Extended Capability Header (Offset 00h)*

Figure 7-136 details allocation of the register fields in the PASID Extended Capability header; Table 7-107 provides the respective bit definitions.

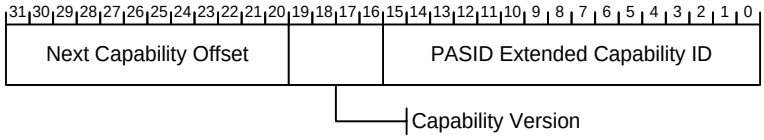


Figure 7-136: PASID Extended Capability Header

Table 7-107: PASID Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PASID Extended Capability ID** – Indicates the PASID Extended Capability structure. This field must return a Capability ID of 001Bh indicating that this is a PASID Extended Capability structure.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.	RO
31:20	**Next Capability Offset** – The offset to the next PCI Extended Capability structure or 000h if no other items exist in the linked list of capabilities.	RO

7.8.8.2 PASID Capability Register (Offset 04h)

Figure 7-137 details the allocation of register bits of the PASID Capability register; Table 7-108 provides the respective bit definitions.

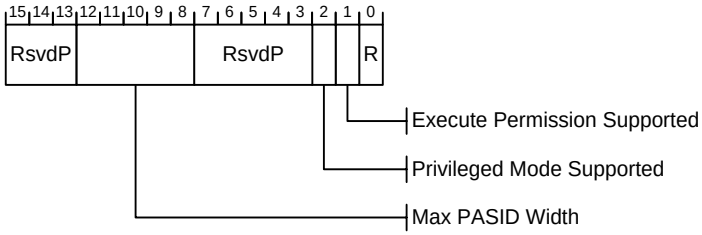


Figure 7-137: PASID Capability Register

Table 7-108: PASID Capability Register

Bit Location	Register Description	Attributes
0	Reserved	RsvdP
1	**Execute Permission Supported** – If Set, the Endpoint supports sending TLPs that have the Execute Requested bit Set. If Clear, the Endpoint will never Set the Execute Requested bit.	RO
2	**Privileged Mode Supported** – If Set, the Endpoint supports operating in Privileged and Non-Privileged modes, and supports sending requests that have the Privileged Mode Requested bit Set. If Clear, the Endpoint will never Set the Privileged Mode Requested bit.	RO
7:3	Reserved	RsvdP
12:8	**Max PASID Width –** Indicates the width of the PASID field supported by the Endpoint. The value n indicates support for PASID values 0 through 2^n-1 (inclusive). The value 0 indicates support for a single PASID (0). The value 20 indicates support for all PASID values (20 bits). This field must be between 0 and 20 (inclusive).	RO

7.8.8.3 PASID Control Register (Offset 06h)

Figure 7-138 details the allocation of register bits of the PASID Control register; Table 7-109 provides the respective bit definitions.

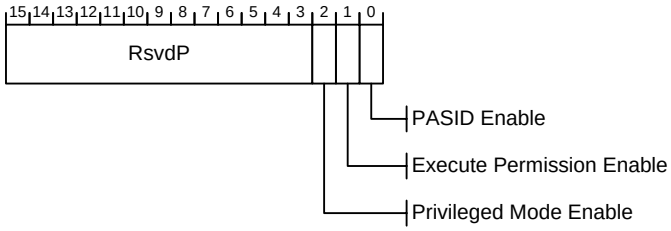


Figure 7-138: PASID Control Register

Table 7-109: PASID Control Register

Bit Location	Register Description	Attributes
0	**PASID Enable –** If Set, the Endpoint is permitted to send and receive TLPs that contain a PASID TLP Prefix. If Clear, the Endpoint is not permitted to do so. Behavior is undefined if the Endpoint supports ATS and this bit changes value when the Enable (E) bit in the ATS Control register is Set (see Table 10-10). Default is 0b.	RW
1	**Execute Permission Enable –** If Set, the Endpoint is permitted to send Requests that have the Execute Requested bit Set. If Clear, the Endpoint is not permitted to do so. Behavior is undefined if the Endpoint supports ATS and this bit changes value when the Enable bit in the ATS Control register is Set (see Table 10-10). If Execute Permission Supported is Clear, this bit is RsvdP. Default is 0b.	RW / RsvdP (see description)
2	**Privileged Mode Enable –** If Set, the Endpoint is permitted to send Requests that have the Privileged Mode Requested bit Set. If Clear, the Endpoint is not permitted to do so. Behavior is undefined if the Endpoint supports ATS and this bit changes value when the Enable bit in the ATS Control register is Set (see Table 10-10). If Privileged Mode Supported is Clear, this bit is RsvdP. Default is 0b.	RW / RsvdP (see description)

7.8.9 Function Readiness Status (FRS) Queuing Extended Capability

The Function Readiness Status (FRS) Queuing Extended Capability is required for Root Ports and Root Complex Event Collectors that support the optional normative FRS Queuing capability. See Section 6.23. This extended capability is only permitted in Root Ports and Root Complex Event Collectors.

If this capability is present in a Function, that Function must also implement either MSI, MSI-X, or both.

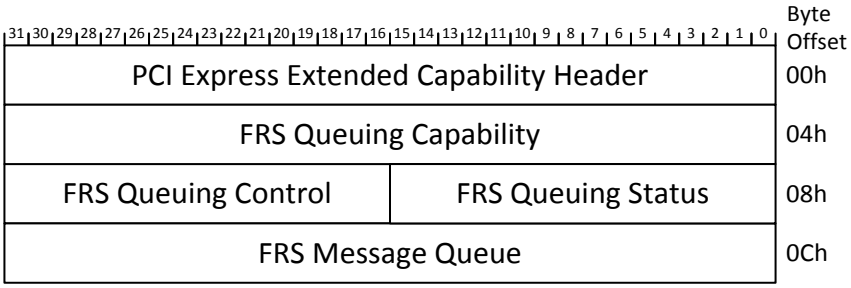


Figure 7-139: FRS Extended Capability

7.8.9.1 *Function Readiness Status (FRS) Queuing Extended Capability Header (Offset 00h)*

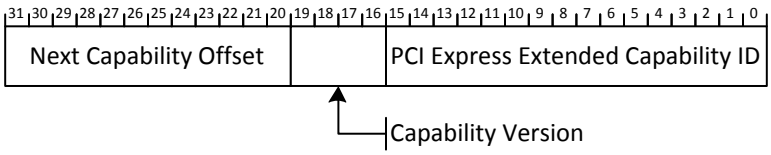


Figure 7-140: FRS Queuing Extended Capability Header

Table 7-110: FRS Queuing Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the extended capability. PCI Express Extended Capability ID for the FRS Queuing Extended Capability is 0021h.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the capability structure present. Must be 1h for this version of the specification.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express Extended Capability structure or 000h if no other items exist in the linked list of capabilities.	RO

7.8.9.2 *FRS Queuing Capability Register (Offset 04h)*

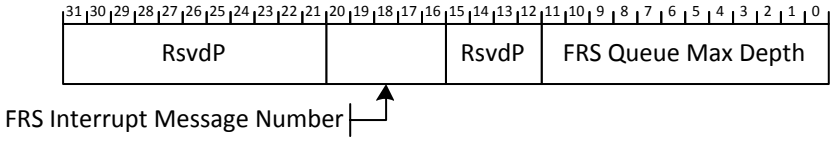


Figure 7-141: FRS Queuing Capability Register

Table 7-111: FRS Queuing Capability Register

Bit Location	Register Description	Attributes
11:0	**FRS Queue Max Depth** – Indicates the implemented queue depth, with valid values ranging from 001h (queue depth of 1) to FFFh (queue depth of 4095) The value of FRS Message Queue Depth must not exceed this value. The value 000h is Reserved.	HwInit
20:16	**FRS Interrupt Message Number** – This register indicates which MSI/MSI-X vector is used for the interrupt message generated in association with FRS Message Received or FRS Message Overflow. For MSI, the value in this register indicates the offset between the base Message Data and the interrupt message that is generated. Hardware is required to update this field so that it is correct if the number of MSI Messages assigned to the Function changes when software writes to the Multiple Message Enable field in the MSI Message Control register. For MSI-X, the value in this register indicates which MSI-X Table entry is used to generate the interrupt message. The entry must be one of the first 32 entries even if the Function implements more than 32 entries. For a given MSI-X implementation, the entry must remain constant. If both MSI and MSI-X are implemented, they are permitted to use different vectors, though software is permitted to enable only one mechanism at a time. If MSI-X is enabled, the value in this register must indicate the vector for MSI-X. If MSI is enabled or neither is enabled, the value in this register must indicate the vector for MSI. If software enables both MSI and MSI-X at the same time, the value in this register is undefined.	RO

7.8.9.3 *FRS Queuing Status Register (Offset 08h)*

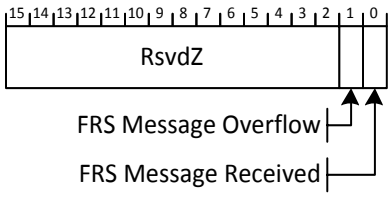


Figure 7-142: FRS Queueing Status Register

Table 7-112: FRS Queuing Status Register

Bit Location	Register Description	Attributes
0	**FRS Message Received** – This bit is Set when a new FRS Message is Received or generated by this Root Port or Root Complex Event Collector. Root Ports must Clear this bit when the Link is DL_Down. Default value of this bit is 0b.	RW1C
1	FRS Message Overflow – This bit is set if the FRS Message queue is full and a new FRS Message is received or generated by this Root Port or Root Complex Event Collector. Root Ports must Clear this bit when the Link is DL_Down. Default value of this bit is 0b.	RW1C
15:2	Reserved	RsvdZ

7.8.9.4 *FRS Queuing Control Register (Offset 0Ah)*

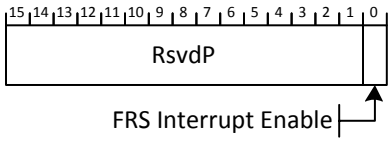


Figure 7-143: FRS Queueing Control Register

Table 7-113: FRS Queuing Control Register

Bit Location	Register Description	Attributes
0	**FRS Interrupt Enable** – When Set and MSI or MSI-X is enabled, the Port must issue an MSI/MSI-X interrupt to indicate the 0b to 1b transition of either the FRS Message Received or the FRS Message Overflow bits. Default value of this bit is 0b.	RW
15:1	Reserved	RsvdP

7.8.9.5 *FRS Message Queue Register (Offset 0Ch)*

The FRS Message Queue Register contains fields from the oldest FRS message in the queue. It also indicates the number of FRS messages in the queue.

A write of any value that includes byte 0 to this register removes the oldest FRS Message from the queue and updates these fields. A write to this register when the queue is empty has no effect.

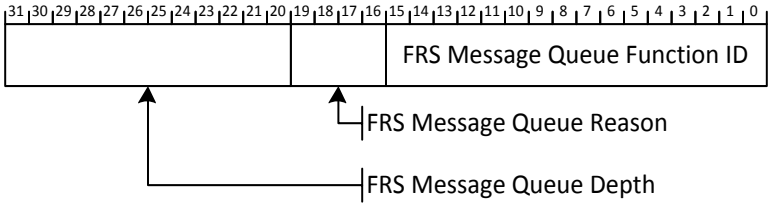


Figure 7-144: FRS Message Queue Register

Table 7-114: FRS Message Queue Register

Bit Location	Register Description	Attributes
15:0	**FRS Message Queue Function ID** – Recorded from the Requester ID of the oldest FRS Message received or generated by this Root Port or Root Complex Event Collector and still in the queue. Undefined if FRS Message Queue Depth is 000h.	RO
19:16	**FRS Message Queue Reason** – Recorded from the FRS Reason of the oldest FRS Message received or generated by this Root Port or Root Complex Event Collector and still in the queue. Undefined if FRS Message Queue Depth is 000h.	RO
31:20	**FRS Message Queue Depth** – indicates the current number of FRS Messages in the queue. The value of 000h indicates an empty queue. Default value of this field is 000h.	RO

7.8.10 Flattening Portal Bridge (FPB) Capability

The Flattening Portal Bridge (FPB) Capability is an optional Capability that is required for any bridge Function that implements FPB. The FPB Capability structure is shown in Figure 7-145.

31 – 16	15 – 8	7 – 0	Byte Offset
RsvdP	Next Pointer	Capability ID	00h
FPB Capabilities Register			04h
FPB RID Vector Control 1 Register			08h
FPB RID Vector Control 2 Register			0Ch
FPB MEM Low Vector Control Register			10h
FPB MEM High Vector Control 1 Register			14h
FPB MEM High Vector Control 2 Register			18h
FPB Vector Access Control Register			1Ch
FPB Vector Access Data Register			20h

Figure 7-145: FPB Capability Structure

If a Switch implements FPB then each of its Ports of the Switch must implement an FPB Capability Structure. A Root Complex is permitted to implement the FPB Capability Structure on some or on all of its Root Ports. A Root Complex is permitted to implement the FPB Capability for internal logical busses.

7.8.10.1 *FPB Capability Header (Offset 00h)*

31 – 16	15 – 8	7 – 0
RsvdP	Next Pointer	Capability ID

Figure 7-146: FPB Capability Header

Table 7-115: FPB Capability Header

Bit Location	Register Description	Attributes
7:0	**Capability ID** – Must be set to 15h	RO
15:8	**Next Pointer** - Pointer to the next item in the capabilities list. Must be 00h for the final item in the list.	RO
31:16	Reserved	RsvdP

7.8.10.2 *FPB Capabilities Register (Offset 04h)*

Figure 7-147 details allocation of register fields for FPB Capability register and Table 7-116 describes the requirements for this register.

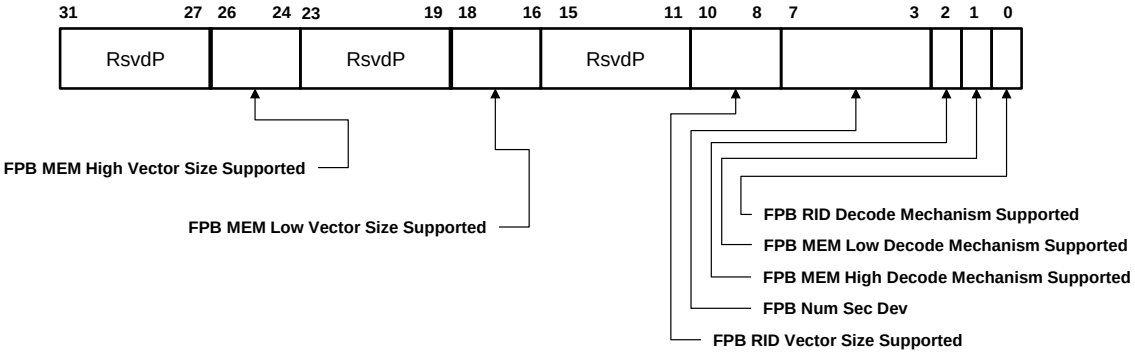


Figure 7-147: FPB Capabilities Register

Table 7-116: FPB Capabilities Register

Bit Location	Register Description	Attributes
0	**FPB RID Decode Mechanism Supported** – If Set, indicates that the FPB RID Vector mechanism is supported.	HwInit
1	**FPB MEM Low Decode Mechanism Supported** - If Set, indicates that the FPB MEM Low Vector mechanism is supported.	HwInit
2	**FPB MEM High Decode Mechanism Supported** - If Set, indicates that the FPB Mem High mechanism is supported.	HwInit
7:3	**FPB Num Sec Dev** - For Upstream Ports of Switches only, this field indicates the quantity of Device Numbers associated with the Secondary Side of the Upstream Port bridge. The quantity is determined by adding one to the numerical value of this field. Although it is recommended that Switch implementations assign Downstream Ports using all 8 allowed Functions per allocated Device Number, such that all Downstream Ports are assigned within a contiguous range of Device and Function Numbers, it is, however, explicitly permitted to assign Downstream Ports to Function Numbers that are not contiguous within the indicated range of Device Numbers, and system software is required to scan for Switch Downstream Ports at every Function Number within the indicated quantity of Device Numbers associated with the Secondary Side of the Upstream Port. This field is Reserved for Downstream Ports.	HwInit/RsvdP

Bit Location	Register Description	Attributes
10:8	**FPB RID Vector Size Supported** – Indicates the size of the FPB RID Vector implemented in hardware, and constrains the allowed values software is permitted to write to the FPB RID Vector Granularity field. Defined encodings are: Value Size Allowed Granularities in RID units 000b 256 bits 8, 64, 256 010b 1 K bits 8, 64 101b 8 K bits 8 All other encodings are Reserved. If the FPB RID Decode Mechanism Supported bit is Clear, then the value in this field is undefined and must be ignored by software.	HwInit
15:11	Reserved	RsvdP
18:16	**FPB MEM Low Vector Size Supported** – Indicates the size of the FPB MEM Low Vector implemented in hardware, and constrains the allowed values software is permitted to write to the FPB MEM Low Vector Start field. Defined encodings are: Value Size Allowed Granularities in MB units 000b 256 bits 1, 2, 4, 8, 16 001b 512 bits 1, 2, 4, 8 010b 1 K bits 1, 2, 4 011b 2 K bits 1, 2 100b 4 K bits 1 All other encodings are Reserved. If the FPB MEM Low Decode Mechanism Supported bit is Clear, then the value in this field is undefined and must be ignored by software.	HwInit
23:19	Reserved	RsvdP
26:24	**FPB MEM High Vector Size Supported** – Indicates the size of the FPB MEM High Vector implemented in hardware. Defined encodings are: 000b 256 bits 001b 512 bits 010b 1 K bits 011b 2 K bits 100b 4 K bits 101b 8 K bits All other encodings are Reserved. All defined Granularities are allowed for all defined vector sizes. If the FPB MEM High Decode Mechanism Supported bit is Clear, then the value in this field is undefined and must be ignored by software.	HwInit
31:27	Reserved	RsvdP

7.8.10.3 *FPB RID Vector Control 1 Register (Offset 08h)*

Figure 7-148 details allocation of register fields for FPB RID Control 1 register and Table 7-117 describes the requirements for this register.

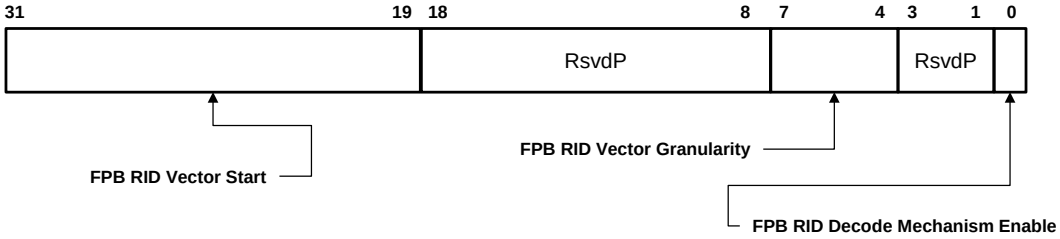


Figure 7-148: FPB RID Vector Control 1 Register

Table 7-117: FPB RID Vector Control 1 Register

Bit Location	Register Description	Attributes
0	**FPB RID Decode Mechanism Enable** – When Set, enables the FPB RID Decode mechanism If the FPB RID Decode Mechanism Supported bit is Clear, then it is permitted for hardware to implement this bit as RO, and in this case the value in this field is undefined. Default value of this bit is 0b.	RW/RO
3:1	Reserved	RsvdP
7:4	**FPB RID Vector Granularity** – The value written by software to this field controls the granularity of the FPB RID Vector and the required alignment of the FPB RID Vector Start field (below). Defined encodings are: ValueGranularity 0000b 8 RIDs 0011b64 RIDs 0101b256 RIDs All other encodings are Reserved. Based on the implemented FPB RID Vector size, hardware is permitted to implement as RW only those bits of this field that can be programmed to non-zero values, in which case the upper order bits are permitted but not required to be hardwired to 0. If the FPB RID Decode Mechanism Supported bit is Clear, then it is permitted for hardware to implement this field as RO, and the value in this field is undefined. For Downstream Ports, if the ARI Forwarding Enable bit in the Device Control 2 Register and the FPB RID Decode Mechanism Enable bit are Set, then software must program 0101b into this field, if this field is programmable. Default value for this field is 0000b.	RW/RO

Bit Location	Register Description	Attributes
18:8	Reserved	RsvdP
31:19	**FPB RID Vector Start** – The value written by software to this field controls the offset at which the FPB RID Vector is applied. The value represents a RID offset in units of 8 RIDs, such that bit 0 of the FPB RID Vector represents the range of RIDs starting from the value represented in this register up to that value plus the FPB RID Vector Granularity minus 1, and bit 1 represents range from this register value plus granularity up to that value plus FPB RID Vector Granularity minus 1, etc. Software must program this field to a value that is naturally aligned (meaning the lower order bits must be 0's) according to the value in the FPB RID Vector Granularity Field as indicated here: FPB RID Vector Granularity Start Alignment Constraint 0000b <no constraint> 0011b …00 0b 0101b …0000 0b All other encodings are Reserved. If this requirement is violated, the hardware behavior is undefined. For Downstream Ports, if the ARI Forwarding Enable bit in the Device Control 2 Register and the FPB RID Decode Mechanism Enable bit are Set, then software must program bits 23:19 of this field to a value of 0000 0b, and the hardware behavior is undefined if any other value is programmed. If the FPB RID Decode Mechanism Supported bit is Clear, then it is permitted for hardware to implement this field as RO, and the value in this field is undefined. Default value for this field is 0000 0000 0000 0b.	RW/RO

7.8.10.4 *FPB RID Vector Control 2 Register (Offset 0Ch)*

Figure 7-149 details allocation of register fields for FPB RID Vector Control 2 register and Table 7-118 describes the requirements for this register

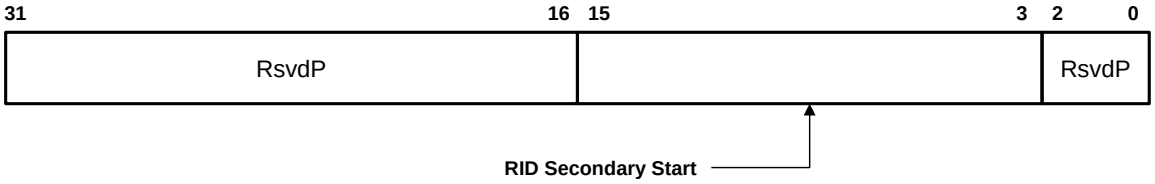


Figure 7-149: FPB RID Vector Control 2 Register

Table 7-118: FPB RID Vector Control 2 Register

Bit Location	Register Description	Attributes
2:0	Reserved	RsvdP
15:3	**RID Secondary Start** – The value written by software to this field controls the RID offset at which Type 1 Configuration Requests passing downstream through the bridge must be converted to Type 0. Bits[2:0] of the RID offset are fixed by hardware as 000b and cannot be modified. For Downstream Ports, if the ARI Forwarding Enable bit in the Device Control 2 register is Set, then software must write bits 7:3 of this field to 0 0000b. If the FPB RID Decode Mechanism Supported bit is Clear, then it is permitted for hardware to implement this field as RO, and the value in this field is undefined. Default value for this field is 0000 0000 0000 0b.	RW/RO
31:16	Reserved	RsvdP

7.8.10.5 *FPB MEM Low Vector Control Register (Offset 10h)*

Figure 7-150 details allocation of register fields for FPB MEM Low Vector Control register and Table 7-119 describes the requirements for this register.

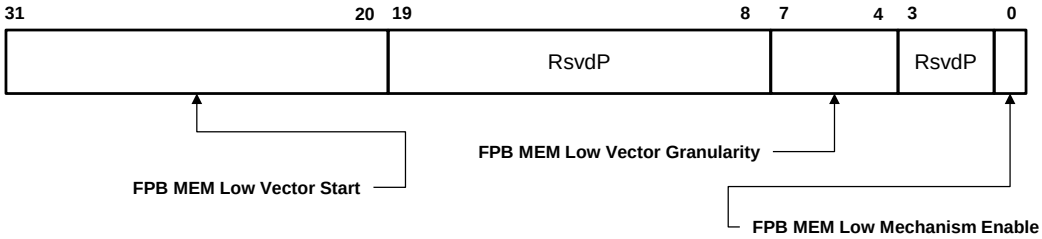


Figure 7-150: FPB MEM Low Vector Control Register

Table 7-119: FPB MEM Low Vector Control Register

Bit Location	Register Description	Attributes
0	**FPB MEM Low Decode Mechanism Enable** - When Set, enables the FPB MEM Low Decode mechanism. If the FPB MEM Low Decode Mechanism Supported bit is Clear, then it is permitted for hardware to implement this bit as RO, and in this case the value in this field is undefined. Default value of this bit is 0b.	RW/RO
3:1	Reserved	RsvdP

Bit Location	Register Description	Attributes
7:4	**FPB MEM Low Vector Granularity** – The value written by software to this field controls the granularity of the FPB MEM Low Vector, and the required alignment of the FPB MEM Low Vector Start field (below). Defined encodings are: ValueGranularity 0000b 1 MB 0001b 2 MB 0010b 4 MB 0011b 8 MB 0100b 16 MB All other encodings are Reserved. Based on the implemented FPB MEM Low Vector size, hardware is permitted to implement as RW only those bits of this field that can be programmed to non-zero values, in which case the upper order bits are permitted but not required to be hardwired to 0. If the FPB MEM Low Decode Mechanism Supported bit is Clear, then it is permitted for hardware to implement this field as RO, and the value in this field is undefined. Default value for this field is 0000b.	RW/RO
19:8	Reserved	RsvdP
31:20	**FPB MEM Low Vector Start** – The value written by software to this field sets bits 31:20 of the base address at which the FPB MEM Low Vector is applied. Software must program this field to a value that is naturally aligned (meaning the lower order bits must be 0's) according to the value in the FPB MEM Low Vector Granularity field as indicated here: FPB MEM Low Vector Granularity Constraint 0000b <no constraint> 0001b …0b 0010b …00b 0011b …000b 0100b …0000b If this requirement is violated, the hardware behavior is undefined. If the FPB MEM Low Decode Mechanism Supported bit is Clear, then it is permitted for hardware to implement this field as RO, and the value in this field is undefined. Default value for this field is 000h.	RW/RO

7.8.10.6 *FPB MEM High Vector Control 1 Register (Offset 14h)*

Figure 7-151 details allocation of register fields for FPB MEM High Vector Control 1 register

and Table 7-120 describes the requirements for this register.

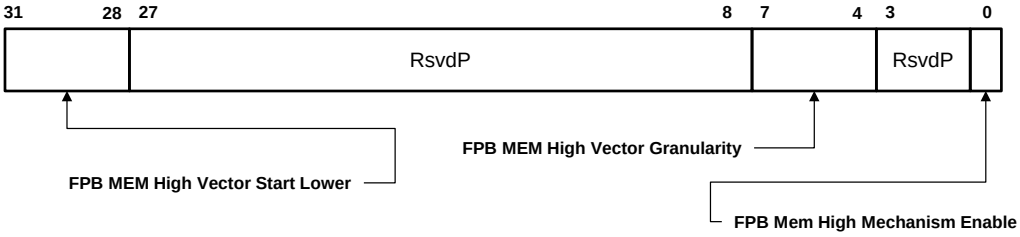


Figure 7-151: FPB MEM High Vector Control 1 Register

Table 7-120: FPB MEM High Vector Control 1 Register

Bit Location	Register Description	Attributes
0	**FPB MEM High Decode Mechanism Enable** - When Set, enables the FPB MEM High Decode mechanism. If the FPB MEM High Decode Mechanism Supported bit is Clear, then it is permitted for hardware to implement this bit as RO, and in this case the value in this field is undefined. Default value of this bit is 0b.	RW/RO
3:1	Reserved	RsvdP
7:4	**FPB MEM High Vector Granularity** – The value written by software to this field controls the granularity of the FPB MEM High Vector, and the required alignment of the FPB MEM High Vector Start Lower field (below). Software is permitted to select any allowed Granularity from the table below regardless of the value in the FPB MEM High Vector Size Supported field. Defined encodings are: ValueGranularity 0000b256 MB 0001b512 MB 0010b 1 GB 0011b 2 GB 0100b 4 GB 0101b 8 GB 0110b 16 GB 0111b 32 GB All other encodings are Reserved. Based on the implemented FPB MEM High Vector size, hardware is permitted to implement as RW only those bits of this field that can be programmed to non-zero values, in which case the upper order bits are permitted but not required to be hardwired to 0. If the FPB MEM High Decode Mechanism Supported bit is Clear, then it is permitted for hardware to implement this field as RO, and the value in this field is undefined. Default value for this field is 0000b.	RW/RO

Bit Location	Register Description	Attributes
27:8	Reserved	RsvdP
31:28	**FPB MEM High Vector Start Lower** – The value written by software to this field sets the lower bits of the base address at which the FPB MEM High Vector is applied. Software must program this field to a value that is naturally aligned (meaning the lower order bits must be 0's) according to the value in the FPB MEM High Vector Granularity Field as indicated here: FPB MEM High Vector Granularity Constraint 0000b <no constraint> 0001b …0b 0010b …00b 0011b …000b 0100b …0000b 0101b …0 0000b 0110b …00 0000b 0111b …000 0000b If this requirement is violated, the hardware behavior is undefined. If the FPB MEM High Decode Mechanism Supported bit is Clear, then it is permitted for hardware to implement this field as RO, and the value in this field is undefined. Default value for this field is 0h.	RW/RO

7.8.10.7 *FPB MEM High Vector Control 2 Register (Offset 18h)*

Figure 7-152 details allocation of register fields for FPB MEM High Vector Control 2 register and Table 7-121 describes the requirements for this register.

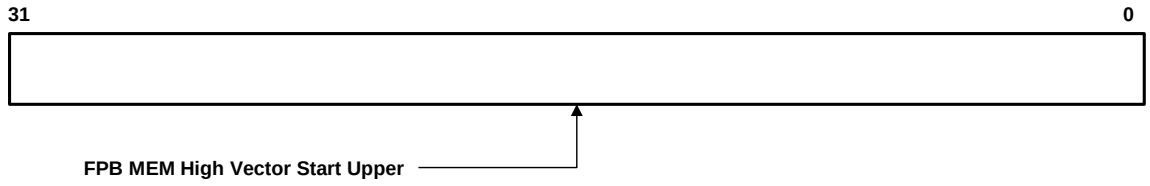


Figure 7-152: FPB MEM High Vector Control 2 Register

Table 7-121: FPB MEM High Vector Control 2 Register

Bit Location	Register Description	Attributes
31:0	**FPB MEM High Vector Start Upper** – The value written by software to this field sets bits 63:32 of the base address at which the FPB MEM High Vector is applied. Software must program this field to a value that is naturally aligned (meaning the lower order bits must be 0's) according to the value in the FPB MEM High Vector Granularity Field as indicated here: FPB MEM High Vector Granularity — Constraint 0000b — <no constraint> 0001b — <no constraint> 0010b — <no constraint> 0011b — <no constraint> 0100b — <no constraint> 0101b — …0b 0110b — …00b 0111b — …000b If this requirement is violated, the hardware behavior is undefined If the FPB MEM High Decode Mechanism Supported bit is Clear, then it is permitted for hardware to implement this field as RO, and the value in this field is undefined. Default value for this field is 0000 0000h.	RW/RO

7.8.10.8 *FPB Vector Access Control Register (Offset 1Ch)*

Figure 7-153 details allocation of register fields for FPB Vector Access Control register and Table 7-122 describes the requirements for this register.

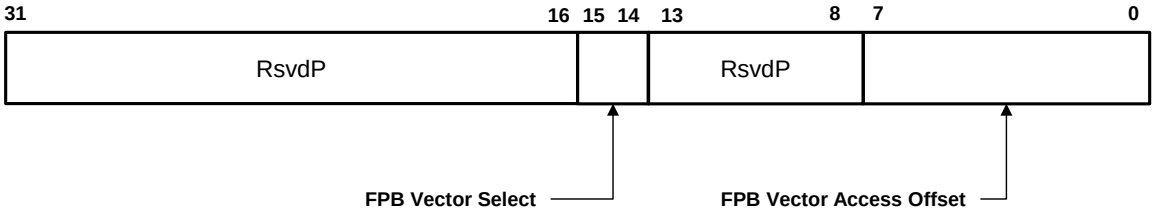


Figure 7-153: FPB Vector Access Control Register

Table 7-122: FPB Vector Access Control Register

Bit Location	Register Description	Attributes
7:0	**FPB Vector Access Offset** – The value in this field indicates the offset of the DWORD portion of the FPB RID, MEM Low or MEM High, Vector that can be read or written by means of the FPB Vector Access Data register. The selection of RID, MEM Low or MEM High is made by the value written to the FPB Vector Select field. The bits of this field map to the offset according to the value in the corresponding FPB RID, MEM Low, or MEM High Vector Size Supported field as shown here: Vector Size Supported / Offset Bits / Vector Access Offset 000b / 2:0 / 2:0 (7:3 unused) 001b / 3:0 / 3:0 (7:4 unused) 010b / 4:0 / 4:0 (7:5 unused) 011b / 5:0 / 5:0 (7:6 unused) 100b / 6:0 / 6:0 (7 unused) 101b / 7:0 / 7:0 All other encodings are Reserved. Bits in this field that are unused per the table above must be written by software as 0b, and are permitted but not required to be implemented as RO. Default value for this field is 00h	RW/RO
13:8	Reserved	RsvdP
15:14	**FPB Vector Select** – The value written to this field selects the Vector to be accessed at the indicated FPB Vector Access Offset. Software must only write this field with values that correspond to supported FPB mechanisms, otherwise the results are undefined. Defined encodings are: 00b: RID 01b: MEM Low 10b: MEM High 11b: Reserved Default value for this field is 00b	RW
31:16	Reserved	RsvdP

7.8.10.9 *FPB Vector Access Data Register (Offset 20h)*

Figure 7-154 details allocation of register fields for FPB Vector Access Data register and Table 7-123 describes the requirements for this register.

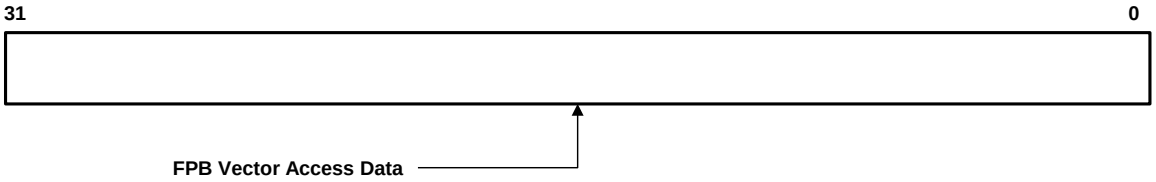


Figure 7-154: FPB Vector Access Data Register

Table 7-123: FPB Vector Access Data Register

Bit Location	Register Description	Attributes
31:0	**FPB Vector Access Data** – Reads from this register return the DW of data from the FPB Vector at the location determined by the value in the FPB Vector Access Offset Register. Writes to this register replace the DW of data from the FPB Vector at the location determined by the value in the FPB Vector Access Offset Register. Behavior of this field is undefined if software programs unsupported values for FPB Vector Select or FPB Vector Access Offset fields, however hardware is required to complete the access to this register normally. Default value for this field is 0000 0000h	RW

7.9 Additional PCI and PCIe Capabilities

This section, contains a description of additional PCI and PCIe capabilities that are individually optional in the *PCI Express Base Specification* but may be required by other PCISIG specifications.

7.9.1 Virtual Channel Capability

The Virtual Channel (VC) Capability is an optional Extended Capability required for devices that have Ports (or for individual Functions) that support functionality beyond the default Traffic Class (TC0) over the default Virtual Channel (VC0). This may apply to devices with only one VC that support TC filtering or to devices that support multiple VCs. Note that a PCI Express device that supports only TC0 over VC0 does not require VC Extended Capability and associated registers. Figure 7-155 provides a high level view of the PCI Express Virtual Channel Capability structure. This structure controls Virtual Channel assignment for PCI Express Links and may be present in any device (or RCRB) that contains (controls) a Port, or any device that has a Multi-Function Virtual Channel (MFVC) Capability structure. Some registers/fields in the PCI Express Virtual Channel Capability structure may have different interpretation for Endpoints, Switch Ports, Root Ports and RCRB. Software must interpret the Device/Port Type field in the PCI Express Capabilities register to determine the availability and meaning of these registers/fields.

The PCI Express VC Capability structure is permitted in the Extended Configuration Space of all single-Function devices or in RCRBs.

A Multi-Function Device at an Upstream Port is permitted to optionally contain a Multi-Function Virtual Channel (MFVC) Capability structure (see Section 7.9.2). If a Multi-Function Device contains an MFVC Capability structure, any or all of its Functions are permitted to contain a VC Capability structure. Per-Function VC Capability structures are also permitted for devices inside a Switch that contain only Switch Downstream Port Functions, or for RCiEPs. Otherwise, only Function 0 is permitted to contain a VC Capability structure.

To preserve software backward compatibility, two Extended Capability IDs are permitted for VC Capability structures: 0002h and 0009h. Any VC Capability structure in a device that also contains an MFVC Capability structure must use the Extended Capability ID 0009h. A VC Capability structure in a device that does not contain an MFVC Capability structure must use the Extended Capability ID 0002h.

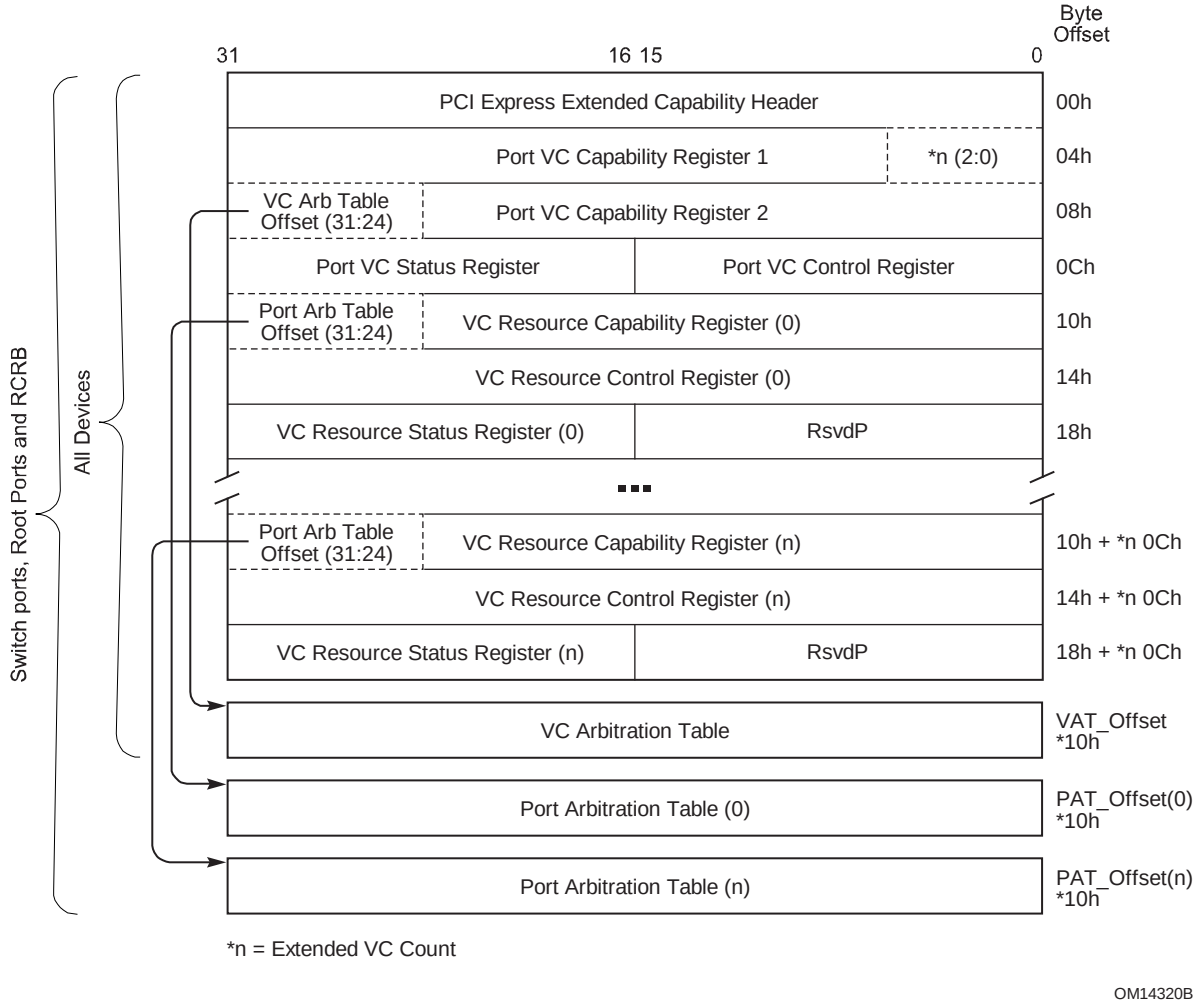


Figure 7-155: PCI Express Virtual Channel Capability Structure

The following sections describe the registers/fields of the PCI Express Virtual Channel Capability structure.

7.9.1.1 *Virtual Channel Extended Capability Header (Offset 00h)*

Refer to Section 7.6.3 for a description of the PCI Express Extended Capability header. A Virtual Channel Capability must use one of two Extended Capability IDs: 0002h or 0009h. Refer to Section 7.9.1 for rules governing when each should be used. Figure 7-156 details allocation of register fields in the Virtual Channel Extended Capability header; Table 7-124 provides the respective bit definitions.

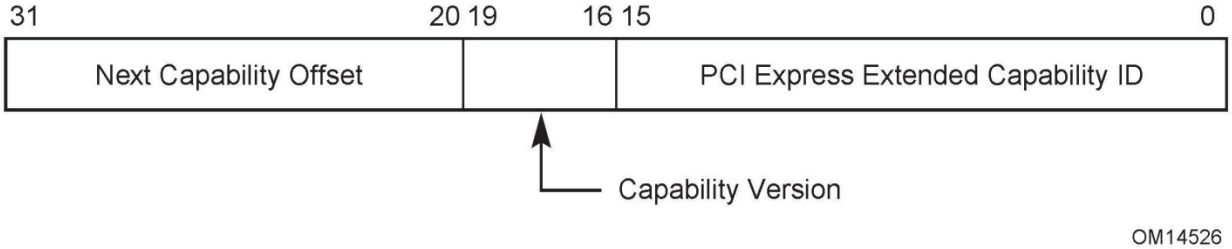


Figure 7-156: Virtual Channel Extended Capability Header

Table 7-124: Virtual Channel Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. Extended Capability ID for the Virtual Channel Capability is either 0002h or 0009h.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh.	RO

7.9.1.2 *Port VC Capability Register 1 (Offset 04h)*

The Port VC Capability register 1 describes the configuration of the Virtual Channels associated with a PCI Express Port. Figure 7-157 details allocation of register fields in the Port VC Capability register 1; Table 7-125 provides the respective bit definitions.

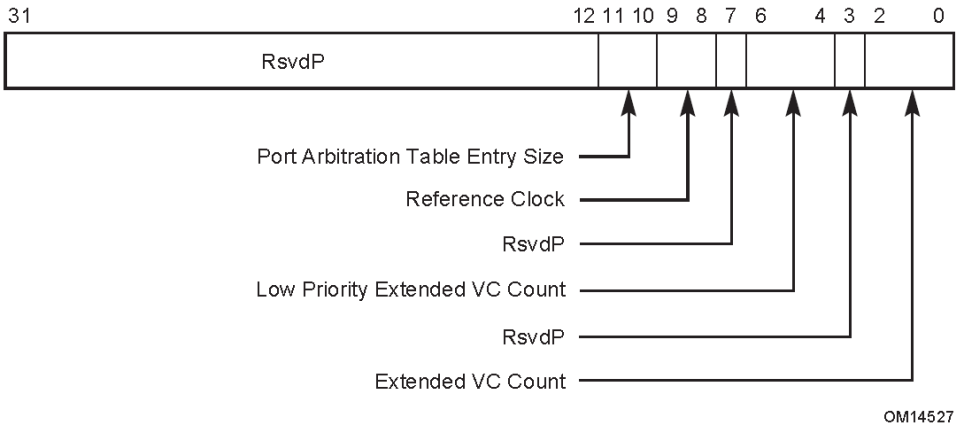


Figure 7-157: Port VC Capability Register 1

Table 7-125: Port VC Capability Register 1

Bit Location	Register Description	Attributes
2:0	**Extended VC Count** – Indicates the number of (extended) Virtual Channels in addition to the default VC supported by the device. This field is valid for all Functions. The minimum value of this field is 000b (for devices that only support the default VC). The maximum value is 7.	RO
6:4	**Low Priority Extended VC Count** – Indicates the number of (extended) Virtual Channels in addition to the default VC belonging to the low-priority VC (LPVC) group that has the lowest priority with respect to other VC resources in a strict-priority VC Arbitration. This field is valid for all Functions. The minimum value of this field is 000b and the maximum value is Extended VC Count.	RO
9:8	**Reference Clock** – Indicates the reference clock for Virtual Channels that support time-based WRR Port Arbitration. This field is valid for RCRBs, Switch Ports, and Root Ports that support peer-to-peer traffic. It is not valid for Root Ports that do not support peer-to-peer traffic, Endpoints, and Switches or Root Complexes not implementing WRR, and must be hardwired to 00b. Defined encodings are: 00b 100 ns reference clock 01b – 11b Reserved	RO

Bit Location	Register Description	Attributes
11:10	**Port Arbitration Table Entry Size** – Indicates the size (in bits) of Port Arbitration table entry in the Function. This field is valid only for RCRBs, Switch Ports, and Root Ports that support peer-to-peer traffic. It is not valid and must be hardwired to 00b for Root Ports that do not support peer-to-peer traffic and Endpoints. Defined encodings are: 00b The size of Port Arbitration table entry is 1 bit. 01b The size of Port Arbitration table entry is 2 bits. 10b The size of Port Arbitration table entry is 4 bits. 11b The size of Port Arbitration table entry is 8 bits.	RO

7.9.1.3 *Port VC Capability Register 2 (Offset 08h)*

The Port VC Capability register 2 provides further information about the configuration of the Virtual Channels associated with a PCI Express Port. Figure 7-158 details allocation of register fields in the Port VC Capability register 2; Table 7-126 provides the respective bit definitions.

Figure 7-158: Port VC Capability Register 2

Table 7-126: Port VC Capability Register 2

Bit Location	Register Description	Attributes
7:0	**VC Arbitration Capability** – Indicates the types of VC Arbitration supported by the Function for the LPVC group. This field is valid for all Functions that report a Low Priority Extended VC Count field greater than 0. For all other Functions, this field must be hardwired to 00h. Each bit location within this field corresponds to a VC Arbitration Capability defined below. When more than 1 bit in this field is Set, it indicates that the Port can be configured to provide different VC arbitration services. Defined bit positions are: Bit 0 — Hardware fixed arbitration scheme, e.g., Round Robin Bit 1 — Weighted Round Robin (WRR) arbitration with 32 phases Bit 2 — WRR arbitration with 64 phases Bit 3 — WRR arbitration with 128 phases Bits 4-7 — Reserved	RO
31:24	**VC Arbitration Table Offset** – Indicates the location of the VC Arbitration Table. This field is valid for all Functions. This field contains the zero-based offset of the table in DQWORDS (16 bytes) from the base address of the Virtual Channel Capability structure. A value of 0 indicates that the table is not present.	RO

7.9.1.4 *Port VC Control Register (Offset 0Ch)*

Figure 7-159 details allocation of register fields in the Port VC Control register; Table 7-127 provides the respective bit definitions.

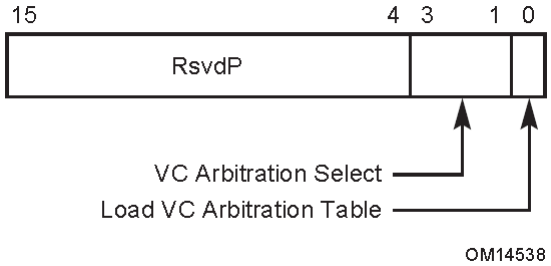


Figure 7-159: Port VC Control Register

Table 7-127: Port VC Control Register

Bit Location	Register Description	Attributes
0	**Load VC Arbitration Table** – Used by software to update the VC Arbitration Table. This bit is valid for all Functions when the selected VC Arbitration uses the VC Arbitration Table. Software sets this bit to request hardware to apply new values programmed into VC Arbitration Table; clearing this bit has no effect. Software checks the VC Arbitration Table Status bit to confirm that new values stored in the VC Arbitration Table are latched by the VC arbitration logic. This bit always returns 0b when read.	RW
3:1	**VC Arbitration Select** – Used by software to configure the VC arbitration by selecting one of the supported VC Arbitration schemes indicated by the VC Arbitration Capability field in the Port VC Capability register 2. This field is valid for all Functions. The permissible values of this field are numbers corresponding to one of the asserted bits in the VC Arbitration Capability field. This field cannot be modified when more than one VC in the LPVC group is enabled.	RW

7.9.1.5 *Port VC Status Register (Offset 0Eh)*

The Port VC Status register provides status of the configuration of Virtual Channels associated with a Port. Figure 7-160 details allocation of register fields in the Port VC Status register; Table 7-128 provides the respective bit definitions.

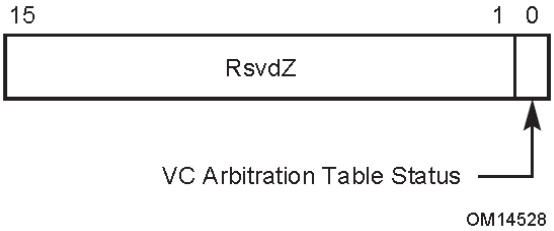


Figure 7-160: Port VC Status Register

Table 7-128: Port VC Status Register

Bit Location	Register Description	Attributes
0	**VC Arbitration Table Status** – Indicates the coherency status of the VC Arbitration Table. This bit is valid for all Functions when the selected VC uses the VC Arbitration Table. This bit is Set by hardware when any entry of the VC Arbitration Table is written by software. This bit is Cleared by hardware when hardware finishes loading values stored in the VC Arbitration Table after software sets the Load VC Arbitration Table bit in the Port VC Control register. Default value of this bit is 0b.	RO

7.9.1.6 *VC Resource Capability Register*

The VC Resource Capability register describes the capabilities and configuration of a particular Virtual Channel resource. Figure 7-161 details allocation of register fields in the VC Resource Capability register; Table 7-129 provides the respective bit definitions.

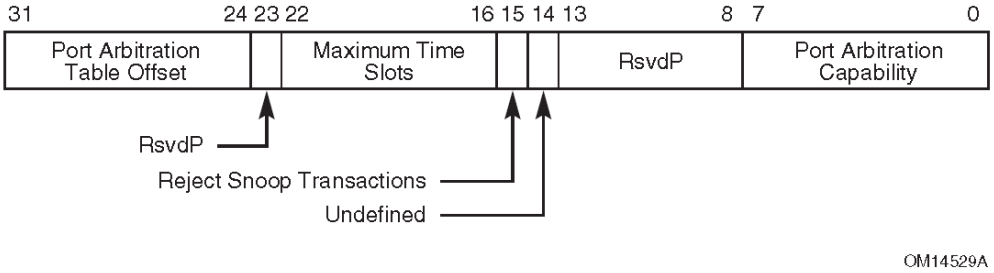


Figure 7-161: VC Resource Capability Register

Table 7-129: VC Resource Capability Register

Bit Location	Register Description	Attributes
7:0	**Port Arbitration Capability** – Indicates types of Port Arbitration supported by the VC resource. This field is valid for all Switch Ports, Root Ports that support peer-to-peer traffic, and RCRBs, but not for Endpoints or Root Ports that do not support peer-to-peer traffic. Each bit location within this field corresponds to a Port Arbitration Capability defined below. When more than 1 bit in this field is Set, it indicates that the VC resource can be configured to provide different arbitration services. Software selects among these capabilities by writing to the Port Arbitration Select field (see Section 7.9.1.7). Defined bit positions are: Bit 0 — Non-configurable hardware-fixed arbitration scheme, e.g., Round Robin (RR) Bit 1 — Weighted Round Robin (WRR) arbitration with 32 phases Bit 2 — WRR arbitration with 64 phases Bit 3 — WRR arbitration with 128 phases Bit 4 — Time-based WRR with 128 phases Bit 5 — WRR arbitration with 256 phases Bits 6-7 — Reserved	RO
14	**Undefined** – The value read from this bit is undefined. In previous versions of this specification, this bit was used to indicate Advanced Packet Switching. System software must ignore the value read from this bit.	RO
15	**Reject Snoop Transactions** – When Clear, transactions with or without the No Snoop bit Set within the TLP header are allowed on this VC. When Set, any transaction for which the No Snoop attribute is applicable but is not Set within the TLP header is permitted to be rejected as an Unsupported Request. Refer to Section 2.2.6.5 for information on where the No Snoop attribute is applicable. This bit is valid for Root Ports and RCRB; it is not valid for Endpoints or Switch Ports.	HwInit
22:16	**Maximum Time Slots** – Indicates the maximum number of time slots (minus one) that the VC resource is capable of supporting when it is configured for time-based WRR Port Arbitration. For example, a value 000 0000b in this field indicates the supported maximum number of time slots is 1 and a value of 111 1111b indicates the supported maximum number of time slots is 128. This field is valid for all Switch Ports, Root Ports that support peer-to-peer traffic, and RCRBs, but is not valid for Endpoints or Root Ports that do not support peer-to-peer traffic. In addition, this field is valid only when the Port Arbitration Capability field indicates that the VC resource supports time-based WRR Port Arbitration.	HwInit

Bit Location	Register Description	Attributes
31:24	**Port Arbitration Table Offset** – Indicates the location of the Port Arbitration Table associated with the VC resource. This field is valid for all Switch Ports, Root Ports that support peer-to-peer traffic, and RCRBs, but is not valid for Endpoints or Root Ports that do not support peer-to-peer traffic. This field contains the zero-based offset of the table in DQWORDS (16 bytes) from the base address of the Virtual Channel Capability structure. A value of 00h indicates that the table is not present.	RO

7.9.1.7 *VC Resource Control Register*

Figure 7-162 details allocation of register fields in the VC Resource Control register; Table 7-130 provides the respective bit definitions.

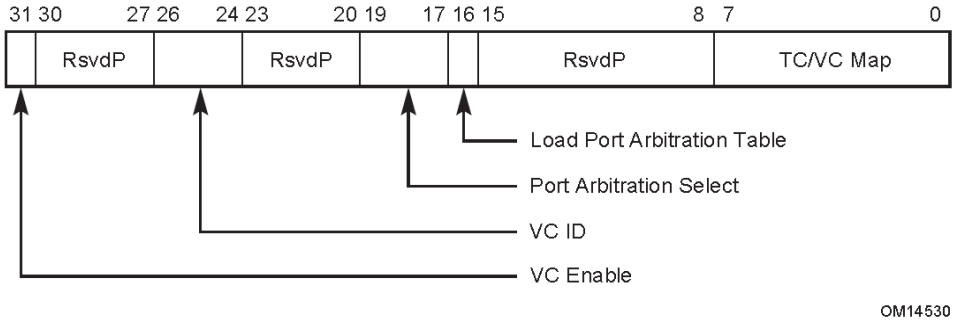


Figure 7-162: VC Resource Control Register

Table 7-130: VC Resource Control Register

Bit Location	Register Description	Attributes
7:0	**TC/VC Map** – This field indicates the TCs that are mapped to the VC resource. This field is valid for all Functions. Bit locations within this field correspond to TC values. For example, when bit 7 is Set in this field, TC7 is mapped to this VC resource. When more than 1 bit in this field is Set, it indicates that multiple TCs are mapped to the VC resource. In order to remove one or more TCs from the TC/VC Map of an enabled VC, software must ensure that no new or outstanding transactions with the TC labels are targeted at the given Link. Default value of this field is FFh for the first VC resource and is 00h for other VC resources. Note: Bit 0 of this field is read-only. It must be Set for the default VC0 and Clear for all other enabled VCs.	RW (see the note for exceptions)

Bit Location	Register Description	Attributes
16	**Load Port Arbitration Table** – When Set, this bit updates the Port Arbitration logic from the Port Arbitration Table for the VC resource. This bit is valid for all Switch Ports, Root Ports that support peer-to-peer traffic, and RCRBs, but is not valid for Endpoints or Root Ports that do not support peer-to-peer traffic. In addition, this bit is only valid when the Port Arbitration Table is used by the selected Port Arbitration scheme (that is indicated by a Set bit in the Port Arbitration Capability field selected by Port Arbitration Select). Software sets this bit to signal hardware to update Port Arbitration logic with new values stored in Port Arbitration Table; clearing this bit has no effect. Software uses the Port Arbitration Table Status bit to confirm whether the new values of Port Arbitration Table are completely latched by the arbitration logic. This bit always returns 0b when read. Default value of this bit is 0b.	RW
19:17	**Port Arbitration Select** – This field configures the VC resource to provide a particular Port Arbitration service. This field is valid for RCRBs, Root Ports that support peer-to-peer traffic, and Switch Ports, but is not valid for Endpoints or Root Ports that do not support peer-to-peer traffic. The permissible value of this field is a number corresponding to one of the asserted bits in the Port Arbitration Capability field of the VC resource.	RW
26:24	**VC ID** – This field assigns a VC ID to the VC resource (see note for exceptions). This field is valid for all Functions. This field cannot be modified when the VC is already enabled. Note: For the first VC resource (default VC), this field is read-only and must be hardwired to 000b.	RW

Bit Location	Register Description	Attributes
31	**VC Enable** – This bit, when Set, enables a Virtual Channel (see note 1 for exceptions). The Virtual Channel is disabled when this bit is cleared. This bit is valid for all Functions. Software must use the VC Negotiation Pending bit to check whether the VC negotiation is complete. Default value of this bit is 1b for the first VC resource and is 0b for other VC resource(s). Notes: 1. This bit is hardwired to 1b for the default VC (VC0), i.e., writing to this bit has no effect for VC0. 2. To enable a Virtual Channel, the VC Enable bits for that Virtual Channel must be Set in both components on a Link. 3. To disable a Virtual Channel, the VC Enable bits for that Virtual Channel must be cleared in both components on a Link. 4. Software must ensure that no traffic is using a Virtual Channel at the time it is disabled. 5. Software must fully disable a Virtual Channel in both components on a Link before re-enabling the Virtual Channel.	RW

7.9.1.8 *VC Resource Status Register*

Figure 7-163 details allocation of register fields in the VC Resource Status register; Table 7-131 provides the respective bit definitions.

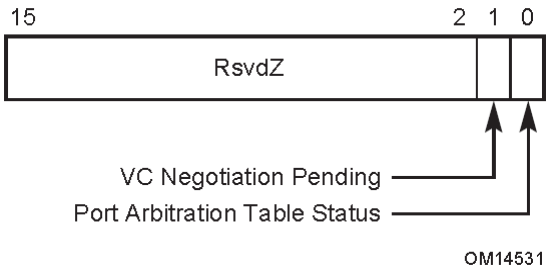


Figure 7-163: VC Resource Status Register

Table 7-131: VC Resource Status Register

Bit Location	Register Description	Attributes
0	**Port Arbitration Table Status** – This bit indicates the coherency status of the Port Arbitration Table associated with the VC resource. This bit is valid for RCRBs, Root Ports that support peer-to-peer traffic, and Switch Ports, but is not valid for Endpoints or Root Ports that do not support peer-to-peer traffic. In addition, this bit is valid only when the Port Arbitration Table is used by the selected Port Arbitration for the VC resource. This bit is Set by hardware when any entry of the Port Arbitration Table is written to by software. This bit is Cleared by hardware when hardware finishes loading values stored in the Port Arbitration Table after software sets the Load Port Arbitration Table bit. Default value of this bit is 0b.	RO
1	**VC Negotiation Pending** –This bit indicates whether the Virtual Channel negotiation (initialization or disabling) is in pending state. This bit is valid for all Functions. The value of this bit is defined only when the Link is in the DL_Active state and the Virtual Channel is enabled (its VC Enable bit is Set). When this bit is Set by hardware, it indicates that the VC resource has not completed the process of negotiation. This bit is Cleared by hardware after the VC negotiation is complete (on exit from the FC_INIT2 state). For VC0, this bit is permitted to be hardwired to 0b. Before using a Virtual Channel, software must check whether the VC Negotiation Pending bits for that Virtual Channel are Clear in both components on the Link.	RO

7.9.1.9 *VC Arbitration Table*

The VC Arbitration Table is a read-write register array that is used to store the arbitration table for VC Arbitration. This register array is valid for all Functions when the selected VC Arbitration uses a WRR table. If it exists, the VC Arbitration Table is located by the VC Arbitration Table Offset field.

The VC Arbitration Table is a register array with fixed-size entries of 4 bits. Figure 7-164 depicts the table structure of an example VC Arbitration Table with 32 phases. Each 4-bit table entry corresponds to a phase within a WRR arbitration period. The definition of table entry is depicted in Table 7-132. The lower 3 bits (bits 0-2) contain the VC ID value, indicating that the corresponding phase within the WRR arbitration period is assigned to the Virtual Channel indicated by the VC ID (must be a valid VC ID that corresponds to an enabled VC).

The highest bit (bit 3) of the table entry is Reserved. The length of the table depends on the selected VC Arbitration as shown in Table 7-133.

When the VC Arbitration Table is used by the default VC Arbitration method, the default values of the table entries must be all zero to ensure forward progress for the default VC (with VC ID of 0).

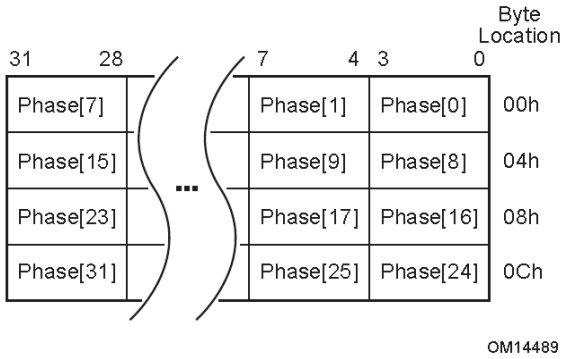


Figure 7-164: Example VC Arbitration Table with 32 Phases

Table 7-132: Definition of the 4-bit Entries in the VC Arbitration Table

Bit Location	Description	Attributes
2:0	VC ID	RW
3	RsvdP	RW

Table 7-133: Length of the VC Arbitration Table

VC Arbitration Select	VC Arbitration Table Length (in # of Entries)
001b	32
010b	64
011b	128

7.9.1.10 *Port Arbitration Table*

The Port Arbitration Table register is a read-write register array that is used to store the WRR or time-based WRR arbitration table for Port Arbitration for the VC resource. This register array is valid for all Switch Ports, Root Ports that support peer-to-peer traffic, and RCRBs, but is not valid for Endpoints or Root Ports that do not support peer-to-peer traffic. It is only present when one or more asserted bits in the Port Arbitration Capability field indicate that the component supports a Port Arbitration scheme that uses a programmable arbitration table. Furthermore, it is only valid when one of the above-mentioned bits in the Port Arbitration Capability field is selected by the Port Arbitration Select field.

The Port Arbitration Table represents one Port arbitration period. Figure 7-165 shows the structure of an example Port Arbitration Table with 128 phases and 2-bit table entries. Each table entry containing a Port Number corresponds to a phase within a Port arbitration period. For example, a table with 2-bit entries can be used by a Switch component with up to four Ports. A Port Number written to a table entry indicates that the phase within the Port Arbitration period is assigned to the selected PCI Express Port (the Port Number must be a valid one).

- When the WRR Port Arbitration is used for a VC of any Egress Port, at each arbitration phase, the Port Arbiter serves one transaction from the Ingress Port indicated by the Port Number of the current phase. When finished, it immediately advances to the next phase. A phase is skipped, i.e., the Port Arbiter simply moves to the next phase immediately if the Ingress Port indicated by the phase does not contain any transaction for the VC (note that a phase cannot contain the Egress Port's Port Number).
- When the Time-based WRR Port Arbitration is used for a VC of any given Port, at each arbitration phase aligning to a virtual timeslot, the Port Arbiter serves one transaction from the Ingress Port indicated by the Port Number of the current phase. It advances to the next phase at the next virtual timeslot. A phase indicates an "idle" timeslot, i.e., the Port Arbiter does not serve any transaction during the phase, if:
  - the phase contains the Egress Port's Port Number, or
  - the Ingress Port indicated by the phase does not contain any transaction for the VC.
  - The Port Arbitration Table Entry Size field in the Port VC Capability register 1 determines the table entry size. The length of the table is determined by the Port Arbitration Select field as shown in Table 7-134.
  - When the Port Arbitration Table is used by the default Port Arbitration for the default VC, the default values for the table entries must contain at least one entry for each of the other PCI Express Ports of the component to ensure forward progress for the default VC for each Port. The table may contain RR or RR-like fair Port Arbitration for the default VC.

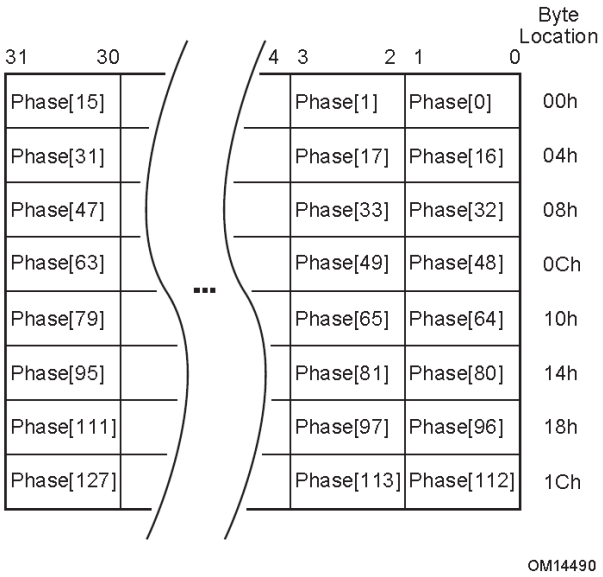


Figure 7-165: Example Port Arbitration Table with 128 Phases and 2-bit Table Entries

Table 7-134: Length of Port Arbitration Table

Port Arbitration Select	Port Arbitration Table Length (in Number of Entries)
001b	32
010b	64
011b	128
100b	128
101b	256

7.9.2 Multi-Function Virtual Channel Capability

The Multi-Function Virtual Channel (MFVC) Capability is an optional Extended Capability that permits enhanced QoS management in a Multi-Function Device, including TC/VC mapping, optional VC arbitration, and optional Function arbitration for Upstream Requests. When implemented, the MFVC Capability structure must be present in the Extended Configuration Space of Function 0 of the Multi-Function Device's Upstream Port. Figure 7-166 provides a high level view of the MFVC Capability structure. This MFVC Capability structure controls Virtual Channel assignment at the PCI Express Upstream Port of the Multi-Function Device, while a VC Capability structure, if present in a Function, controls the Virtual Channel assignment for that individual Function.

A Multi-Function Device is permitted to have an MFVC Capability structure even if none of its Functions have a VC Capability structure. However, an MFVC Capability structure is permitted only in Function 0 in the Upstream Port of a Multi-Function Device.

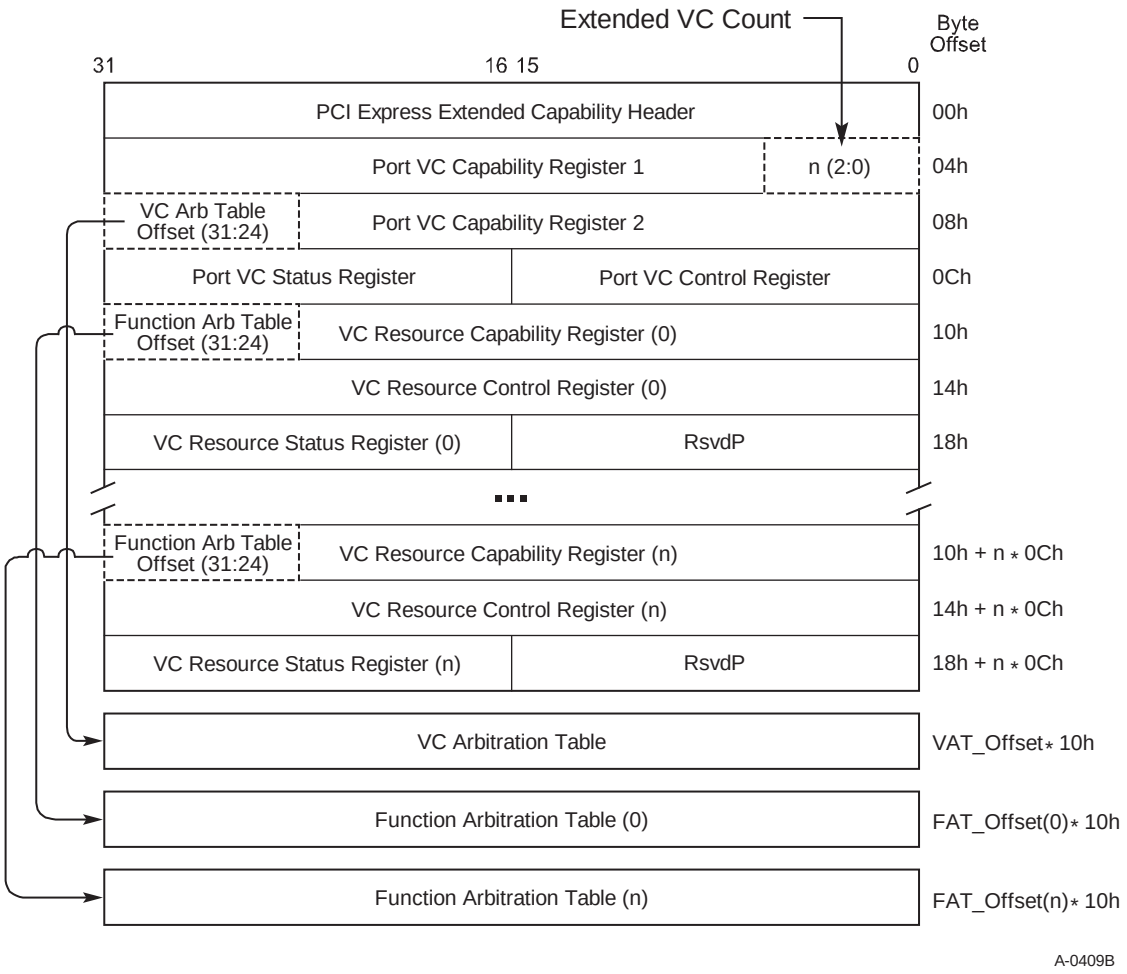


Figure 7-166: PCI Express MFVC Capability Structure

The following sections describe the registers/fields of the PCI Express MFVC Capability structure.

7.9.2.1 *MFVC Extended Capability Header (Offset 00h)*

Refer to Section 7.6.3 for a description of the PCI Express Extended Capability header. The Extended Capability ID for the MFVC Capability is 0008h. Figure 7-167 details allocation of register fields in the MFVC Extended Capability header; Table 7-135 provides the respective bit definitions.

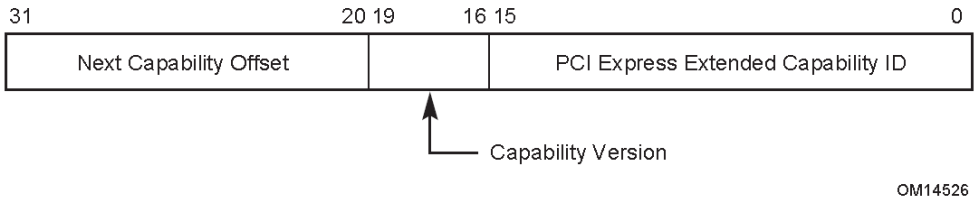


Figure 7-167: MFVC Extended Capability Header

Table 7-135: MFVC Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. The Extended Capability ID for the MFVC Capability is 0008h.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh.	RO

7.9.2.2 *Port VC Capability Register 1 (Offset 04h)*

The Port VC Capability register 1 describes the configuration of the Virtual Channels associated with a PCI Express Port of the Multi-Function Device. Figure 7-168 details allocation of register fields in the Port VC Capability register 1; Table 7-136 provides the respective bit definitions.

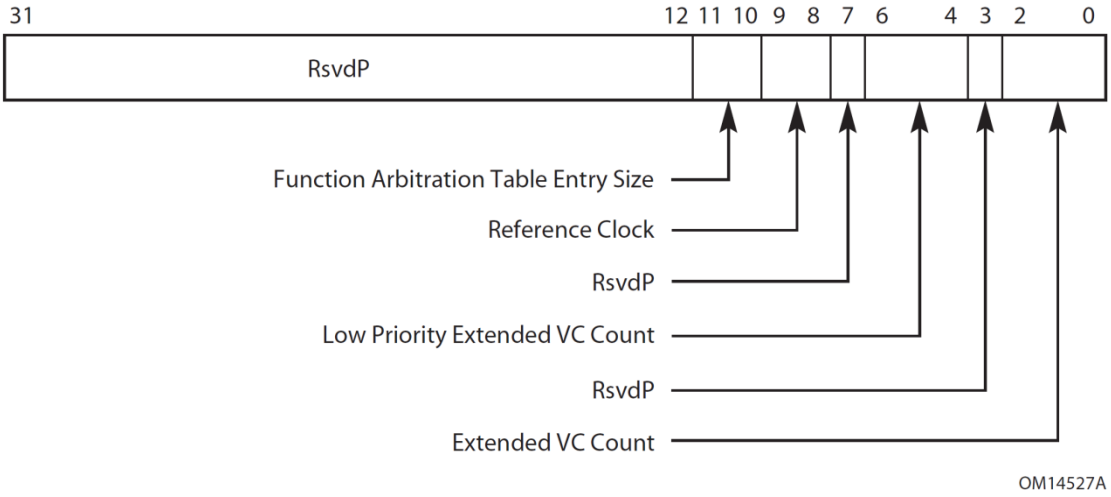


Figure 7-168: Port VC Capability Register 1

Table 7-136: Port VC Capability Register 1

Bit Location	Register Description	Attributes
2:0	**Extended VC Count** – Indicates the number of (extended) Virtual Channels in addition to the default VC supported by the device. The minimum value of this field is zero (for devices that only support the default VC). The maximum value is seven.	RO
6:4	**Low Priority Extended VC Count** – Indicates the number of (extended) Virtual Channels in addition to the default VC belonging to the low-priority VC (LPVC) group that has the lowest priority with respect to other VC resources in a strict-priority VC Arbitration. The minimum value of this field is 000b and the maximum value is Extended VC Count.	RO
9:8	**Reference Clock** – Indicates the reference clock for Virtual Channels that support time-based WRR Function Arbitration. Defined encodings are: 00b 100 ns reference clock 01b – 11b Reserved	RO

Bit Location	Register Description	Attributes
11:10	**Function Arbitration Table Entry Size** – Indicates the size (in bits) of Function Arbitration table entry in the device. Defined encodings are: 00b Size of Function Arbitration table entry is 1 bit 01b Size of Function Arbitration table entry is 2 bits 10b Size of Function Arbitration table entry is 4 bits 11b Size of Function Arbitration table entry is 8 bits	RO

7.9.2.3 *Port VC Capability Register 2 (Offset 08h)*

The Port VC Capability register 2 provides further information about the configuration of the Virtual Channels associated with a PCI Express Port of the Multi-Function Device. Figure 7-169 details allocation of register fields in the Port VC Capability register 2; Table 7-137 provides the respective bit definitions.

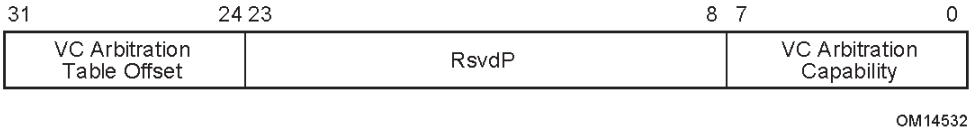


Figure 7-169: Port VC Capability Register 2

Table 7-137: Port VC Capability Register 2

Bit Location	Register Description	Attributes
7:0	**VC Arbitration Capability** – Indicates the types of VC Arbitration supported by the device for the LPVC group. This field is valid for all devices that report a Low Priority Extended VC Count greater than 0. Each bit location within this field corresponds to a VC Arbitration Capability defined below. When more than 1 bit in this field is Set, it indicates that the device can be configured to provide different VC arbitration services. Defined bit positions are: Bit 0 Hardware fixed arbitration scheme, e.g., Round Robin Bit 1 Weighted Round Robin (WRR) arbitration with 32 phases Bit 2 WRR arbitration with 64 phases Bit 3 WRR arbitration with 128 phases Bits 4-7 Reserved	RO

Bit Location	Register Description	Attributes
31:24	**VC Arbitration Table Offset** – Indicates the location of the VC Arbitration Table. This field contains the zero-based offset of the table in DQWORDS (16 bytes) from the base address of the MFVC Capability structure. A value of 00h indicates that the table is not present.	RO

7.9.2.4 *Port VC Control Register (Offset 0Ch)*

Figure 7-170 details allocation of register fields in the Port VC Control register; Table 7-138 provides the respective bit definitions.

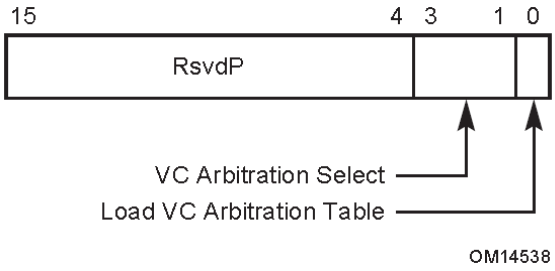


Figure 7-170: Port VC Control Register

Table 7-138: Port VC Control Register

Bit Location	Register Description	Attributes
0	**Load VC Arbitration Table** – Used by software to update the VC Arbitration Table. This bit is valid when the selected VC Arbitration uses the VC Arbitration Table. Software Sets this bit to request hardware to apply new values programmed into VC Arbitration Table; Clearing this bit has no effect. Software checks the VC Arbitration Table Status bit to confirm that new values stored in the VC Arbitration Table are latched by the VC arbitration logic. This bit always returns 0b when read.	RW
3:1	**VC Arbitration Select** – Used by software to configure the VC arbitration by selecting one of the supported VC Arbitration schemes indicated by the VC Arbitration Capability field in the Port VC Capability register 2. The permissible values of this field are numbers corresponding to one of the asserted bits in the VC Arbitration Capability field. This field cannot be modified when more than one VC in the LPVC group is enabled.	RW

7.9.2.5 *Port VC Status Register (Offset 0Eh)*

The Port VC Status register provides status of the configuration of Virtual Channels associated with a Port of the Multi-Function Device. Figure 7-171 details allocation of register fields in the Port VC Status register; Table 7-139 provides the respective bit definitions.

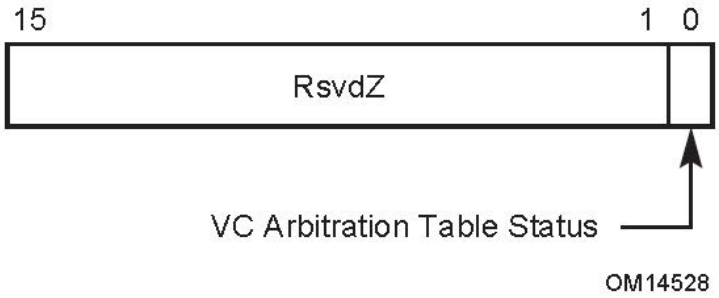


Figure 7-171: Port VC Status Register

Table 7-139: Port VC Status Register

Bit Location	Register Description	Attributes
0	**VC Arbitration Table Status** – Indicates the coherency status of the VC Arbitration Table. This bit is valid when the selected VC uses the VC Arbitration Table. This bit is Set by hardware when any entry of the VC Arbitration Table is written by software. This bit is Cleared by hardware when hardware finishes loading values stored in the VC Arbitration Table after software sets the Load VC Arbitration Table bit in the Port VC Control register. Default value of this bit is 0b.	RO

7.9.2.6 *VC Resource Capability Register*

The VC Resource Capability register describes the capabilities and configuration of a particular Virtual Channel resource. Figure 7-172 details allocation of register fields in the VC Resource Capability register; Table 7-140 provides the respective bit definitions.

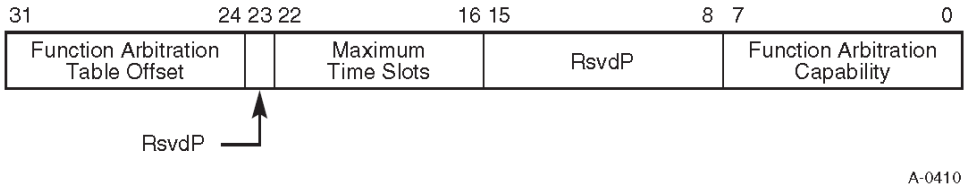


Figure 7-172: VC Resource Capability Register

Table 7-140: VC Resource Capability Register

Bit Location	Register Description	Attributes
7:0	**Function Arbitration Capability** – Indicates types of Function Arbitration supported by the VC resource. Each bit location within this field corresponds to a Function Arbitration Capability defined below. When more than 1 bit in this field is Set, it indicates that the VC resource can be configured to provide different arbitration services. Software selects among these capabilities by writing to the Function Arbitration Select field (see Section 7.9.2.7). Defined bit positions are: Bit 0 — Non-configurable hardware-fixed arbitration scheme, e.g., Round Robin (RR) Bit 1 — Weighted Round Robin (WRR) arbitration with 32 phases Bit 2 — WRR arbitration with 64 phases Bit 3 — WRR arbitration with 128 phases Bit 4 — Time-based WRR with 128 phases Bit 5 — WRR arbitration with 256 phases Bits 6-7 — Reserved	RO
22:16	**Maximum Time Slots** – Indicates the maximum number of time slots (minus 1) that the VC resource is capable of supporting when it is configured for time-based WRR Function Arbitration. For example, a value of 000 0000b in this field indicates the supported maximum number of time slots is 1 and a value of 111 1111b indicates the supported maximum number of time slots is 128. This field is valid only when the Function Arbitration Capability indicates that the VC resource supports time-based WRR Function Arbitration.	HwInit
31:24	**Function Arbitration Table Offset** – Indicates the location of the Function Arbitration Table associated with the VC resource. This field contains the zero-based offset of the table in DQWORDS (16 bytes) from the base address of the MFVC Capability structure. A value of 00h indicates that the table is not present.	RO

7.9.2.7 *VC Resource Control Register*

Figure 7-173 details allocation of register fields in the VC Resource Control register; Table 7-141 provides the respective bit definitions.

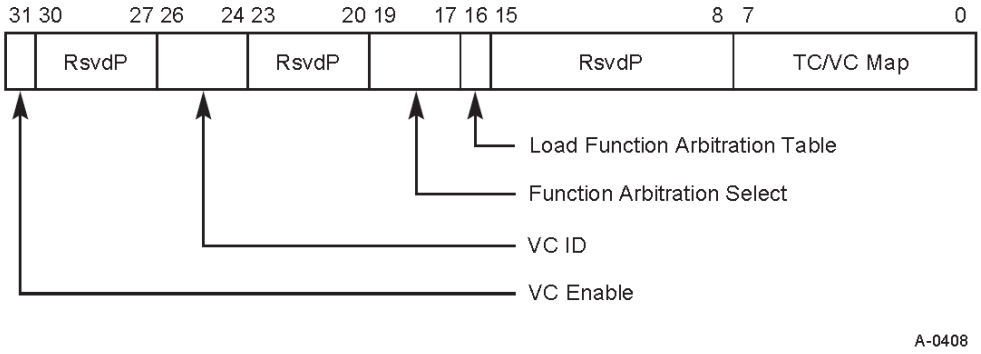


Figure 7-173: VC Resource Control Register

Table 7-141: VC Resource Control Register

Bit Location	Register Description	Attributes
7:0	**TC/VC Map** – This field indicates the TCs that are mapped to the VC resource. Bit locations within this field correspond to TC values. For example, when bit 7 is Set in this field, TC7 is mapped to this VC resource. When more than 1 bit in this field is Set, it indicates that multiple TCs are mapped to the VC resource. In order to remove one or more TCs from the TC/VC Map of an enabled VC, software must ensure that no new or outstanding transactions with the TC labels are targeted at the given Link. Default value of this field is FFh for the first VC resource and is 00h for other VC resources. Note: Bit 0 of this field is read-only. It must be hardwired to 1b for the default VC0 and hardwired to 0b for all other enabled VCs.	RW (see the note for exceptions)

Bit Location	Register Description	Attributes
16	**Load Function Arbitration Table** – When Set, this bit updates the Function Arbitration logic from the Function Arbitration Table for the VC resource. This bit is only valid when the Function Arbitration Table is used by the selected Function Arbitration scheme (that is indicated by a Set bit in the Function Arbitration Capability field selected by Function Arbitration Select). Software sets this bit to signal hardware to update Function Arbitration logic with new values stored in the Function Arbitration Table; clearing this bit has no effect. Software uses the Function Arbitration Table Status bit to confirm whether the new values of Function Arbitration Table are completely latched by the arbitration logic. This bit always returns 0b when read. Default value of this bit is 0b.	RW
19:17	**Function Arbitration Select** – This field configures the VC resource to provide a particular Function Arbitration service. The permissible value of this field is a number corresponding to one of the asserted bits in the Function Arbitration Capability field of the VC resource.	RW
26:24	**VC ID** – This field assigns a VC ID to the VC resource (see note for exceptions). This field cannot be modified when the VC is already enabled. Note: For the first VC resource (default VC), this field is a read-only field that must be hardwired to 000b.	RW
31	**VC Enable** – When Set, this bit enables a Virtual Channel (see note 1 for exceptions). The Virtual Channel is disabled when this bit is cleared. Software must use the VC Negotiation Pending bit to check whether the VC negotiation is complete. Default value of this bit is 1b for the first VC resource and 0b for other VC resource(s). Notes: 1. This bit is hardwired to 1b for the default VC (VC0), i.e., writing to this field has no effect for VC0. 2. To enable a Virtual Channel, the VC Enable bits for that Virtual Channel must be Set in both components on a Link. 3. To disable a Virtual Channel, the VC Enable bits for that Virtual Channel must be Cleared in both components on a Link. 4. Software must ensure that no traffic is using a Virtual Channel at the time it is disabled. 5. Software must fully disable a Virtual Channel in both components on a Link before re-enabling the Virtual Channel.	RW

7.9.2.8 *VC Resource Status Register*

Figure 7-174 details allocation of register fields in the VC Resource Status register; Table 7-142 provides the respective bit definitions.

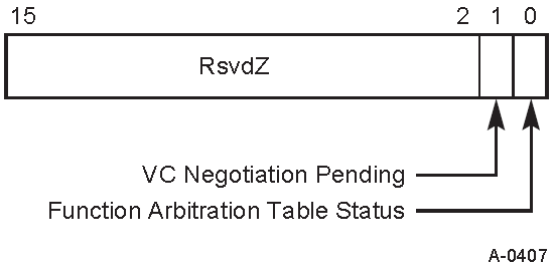


Figure 7-174: VC Resource Status Register

Table 7-142: VC Resource Status Register

Bit Location	Register Description	Attributes
0	**Function Arbitration Table Status** – This bit indicates the coherency status of the Function Arbitration Table associated with the VC resource. This bit is valid only when the Function Arbitration Table is used by the selected Function Arbitration for the VC resource. This bit is Set by hardware when any entry of the Function Arbitration Table is written to by software. This bit is Cleared by hardware when hardware finishes loading values stored in the Function Arbitration Table after software sets the Load Function Arbitration Table bit. Default value of this bit is 0b.	RO
1	**VC Negotiation Pending** – This bit indicates whether the Virtual Channel negotiation (initialization or disabling) is in pending state. When this bit is Set by hardware, it indicates that the VC resource is still in the process of negotiation. This bit is Cleared by hardware after the VC negotiation is complete. For a non-default Virtual Channel, software may use this bit when enabling or disabling the VC. For the default VC, this bit indicates the status of the process of Flow Control initialization. Before using a Virtual Channel, software must check whether the VC Negotiation Pending bits for that Virtual Channel are Clear in both components on a Link.	RO

7.9.2.9 *VC Arbitration Table*

The definition of the VC Arbitration Table in the MFVC Capability structure is identical to that in the VC Capability structure (see Section 7.9.1.9).

7.9.2.10 *Function Arbitration Table*

The Function Arbitration Table register in the MFVC Capability structure takes the same form as the Port Arbitration Table register in the VC Capability structure (see Section 7.9.1.10).

The Function Arbitration Table register is a read-write register array that is used to store the WRR or time-based WRR arbitration table for Function Arbitration for the VC resource. It is only present when one or more asserted bits in the Function Arbitration Capability field indicate that the Multi-Function Device supports a Function Arbitration scheme that uses a programmable arbitration table. Furthermore, it is only valid when one of the above-mentioned bits in the Function Arbitration Capability field is selected by the Function Arbitration Select field.

The Function Arbitration Table represents one Function arbitration period. Each table entry containing a Function Number or Function Group[139] Number corresponds to a phase within a Function Arbitration period. The table entry size requirements are as follows:

- The table entry size for non-ARI devices must support enough values to specify all implemented Functions plus at least one value that does not correspond to an implemented Function. For example, a table with 2-bit entries can be used by a Multi-Function Device with up to three Functions.
- The table entry size for ARI Devices must be either 4 bits or 8 bits.
    - If MFVC Function Groups are enabled, each entry maps to a single Function Group. Arbitration between multiple Functions within a Function Group is implementation specific, but must guarantee forward progress.
    - If MFVC Function Groups are not enabled and 4-bit entries are implemented, a given entry maps to all Functions whose Function Number modulo 8 matches its value. Similarly, if 8-bit entries are implemented, a given entry maps to all Functions whose Function Number modulo 128 matches its value. If a given entry maps to multiple Functions, arbitration between those Functions is implementation specific, but must guarantee forward progress.

A Function Number or Function Group Number written to a table entry indicates that the phase within the Function Arbitration period is assigned to the selected Function or Function Group (the Function Number or Function Group Number must be a valid one).

- When the WRR Function Arbitration is used for a VC of the Egress Port of the Multi-Function Device, at each arbitration phase the Function Arbiter serves one transaction from the Function or Function Group indicated by the Function Number or Function Group Number of the current phase. When finished, it immediately advances to the next phase. A phase is skipped, i.e., the Function Arbiter simply moves to the next phase immediately if the Function or Function Group indicated by the phase does not contain any transaction for the VC.
- When the Time-based WRR Function Arbitration is used for a VC of the Egress Port of the Multi-Function Device, at each arbitration phase aligning to a virtual timeslot, the Function Arbiter serves one transaction from the Function or Function Group indicated by the Function Number or Function Group Number of the current phase. It advances to the next phase at the

[139] If an ARI Device supports MFVC Function Groups capability and ARI-aware software enables it, arbitration is based on Function Groups instead of Functions. See Section 7.23.

next virtual timeslot. A phase indicates an "idle" timeslot, i.e., the Function Arbiter does not serve any transaction during the phase, if:

- the phase contains the Number of a Function or a Function Group that does not exist, or
- the Function or Function Group indicated by the phase does not contain any transaction for the VC.

The Function Arbitration Table Entry Size field in the Port VC Capability register 1 determines the table entry size. The length of the table is determined by the Function Arbitration Select field as shown in Table 7-143.

When the Function Arbitration Table is used by the default Function Arbitration for the default VC, the default values for the table entries must contain at least one entry for each of the active Functions or Function Groups in the Multi-Function Device to ensure forward progress for the default VC for the Multi-Function Device's Upstream Port. The table may contain RR or RR-like fair Function Arbitration for the default VC.

Table 7-143: Length of Function Arbitration Table

Function Arbitration Select	Function Arbitration Table Length (in Number of Entries)
001b	32
010b	64
011b	128
100b	128
101b	256

7.9.3 Device Serial Number Capability

The PCI Express Device Serial Number Capability is an optional Extended Capability that may be implemented by any PCI Express device Function. The Device Serial Number is a read-only 64-bit value that is unique for a given PCI Express device. Figure 7-175 details allocation of register fields in the PCI Express Capability structure.

It is permitted but not recommended for RCiEPs to implement this Capability.

RCiEPs that implement this Capability are permitted but not required to return the same Device Serial Number value as that reported by other RCiEPs of the same Root Complex.

All Multi-Function DeviceDevices other than RCiEPs that implement this Capability must implement it for Function 0; other Functions that implement this Capability must return the same Device Serial Number value as that reported by Function 0.

RCiEPs are permitted to implement or not implement this Capability on an individual basis, independent of whether they are part of a Multi-Function Device.

A PCI Express component other than a Root Complex containing multiple Devices such as a PCI Express Switch that implements this Capability must return the same Device Serial Number for each device.

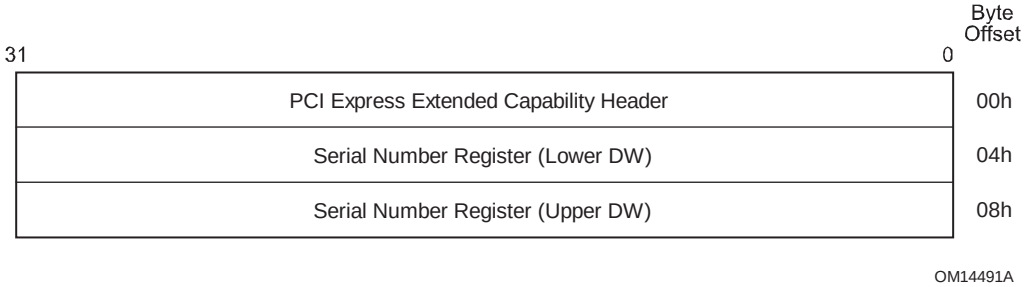


Figure 7-175: PCI Express Device Serial Number Capability Structure

7.9.3.1 *Device Serial Number Extended Capability Header (Offset 00h)*

Figure 7-176 details allocation of register fields in the Device Serial Number Extended Capability header; Table 7-144 provides the respective bit definitions. Refer to Section 7.6.3 for a description of the PCI Express Extended Capability header. The Extended Capability ID for the Device Serial Number Capability is 0003h.

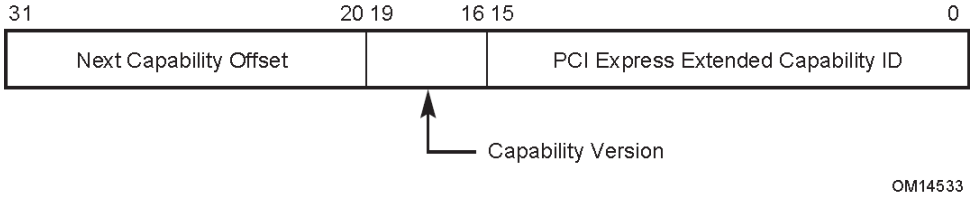


Figure 7-176: Device Serial Number Extended Capability Header

Table 7-144: Device Serial Number Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. Extended Capability ID for the Device Serial Number Capability is 0003h.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.	RO

Bit Location	Register Description	Attributes
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh.	RO

7.9.3.2 *Serial Number Register (Offset 04h)*

The Serial Number register is a 64-bit field that contains the IEEE defined 64-bit extended unique identifier (EUI-64™). Figure 7-177 details allocation of register fields in the Serial Number register; Table 7-145 provides the respective bit definitions.

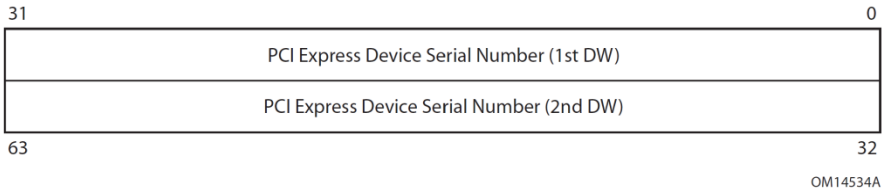


Figure 7-177: Serial Number Register

Table 7-145: Serial Number Register

Bit Location	Register Description	Attributes
63:0	**PCI Express Device Serial Number** – This field contains the IEEE defined 64-bit extended unique identifier (EUI-64™). This identifier includes a 24-bit company id value assigned by IEEE registration authority and a 40-bit extension identifier assigned by the manufacturer.	RO

7.9.4 Vendor-Specific Capability

The Vendor-Specific Capability is a capability structure in PCI-compatible Configuration Space (first 256 bytes) as shown in Figure 7-178.

The Vendor-Specific Capability allows device vendors to use the Capability mechanism for vendor-specific information. The layout of the information is vendor-specific, except for the first three bytes, as explained below.

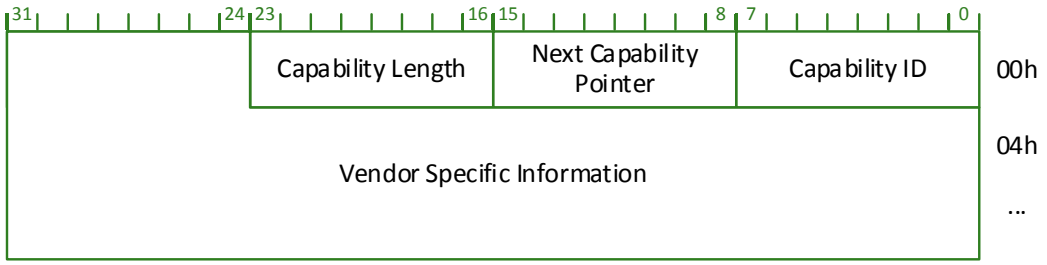


Figure 7-178: Vendor-Specific Capability

Table 7-146: Vendor-Specific Capability

Bit Location	Register Description	Attributes
7:0	**Capability ID** – Indicates the PCI Express Capability structure. This field must return a Capability ID of 09h indicating that this is a Vendor-Specific Capability structure. Each Function may have only one item in its capability list with Capability ID set to 09h.	RO
15:8	**Next Capability Pointer** – This field contains the offset to the next PCI Capability structure or 00h if no other items exist in the linked list of Capabilities.	RO
23:16	**Capability Length** – This field provides the number of bytes in the Capability structure (including the three bytes consumed by the Capability ID, Next Capability Pointer, and Capability Length field).	RO
31:24	**Vendor Specific Information**	Vendor Specific

7.9.5 Vendor-Specific Extended Capability

The PCI Express Vendor-Specific Extended Capability (VSEC) is an optional Extended Capability that is permitted to be implemented by any PCI Express Function or RCRB. This allows PCI Express component vendors to use the Extended Capability mechanism to expose vendor-specific registers.

A single PCI Express Function or RCRB is permitted to contain multiple VSEC structures.

An example usage is a set of vendor-specific features that are intended to go into an on-going series of components from that vendor. A VSEC structure can tell vendor-specific software which features a particular component supports, including components developed after the software was released.

Figure 7-179 details allocation of register fields in the VSEC structure. The structure of the PCI Express Extended Capability header and the vendor-specific header is architected by this specification.

With a PCI Express Function, the structure and definition of the vendor-specific Registers area is determined by the vendor indicated by the Vendor ID field located at byte offset 00h in PCI-

compatible Configuration Space. With an RCRB, a VSEC is permitted only if the RCRB also contains an RCRB Header Capability structure, which contains a Vendor ID field indicating the vendor.

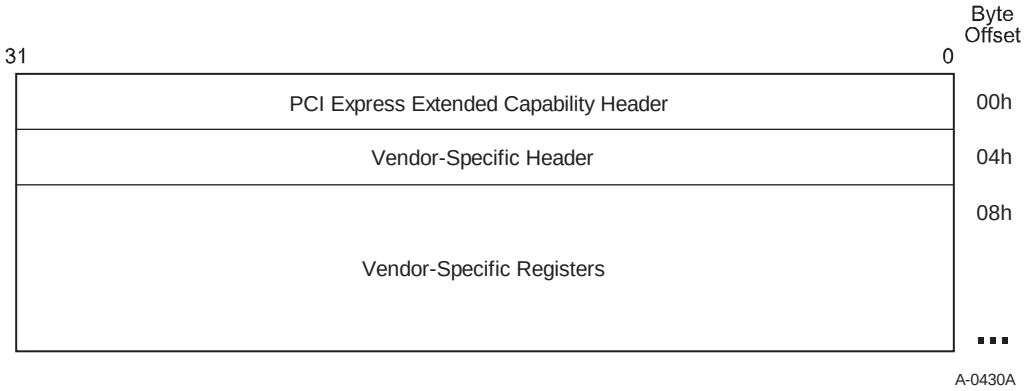


Figure 7-179: PCI Express VSEC Structure

7.9.5.1 *Vendor-Specific Extended Capability Header (Offset 00h)*

Figure 7-180 details allocation of register fields in the Vendor-Specific Extended Capability header; Table 7-147 provides the respective bit definitions. Refer to Section 7.6.3 for a description of the PCI Express Extended Capability header. The Extended Capability ID for the Vendor-Specific Extended Capability is 000Bh.

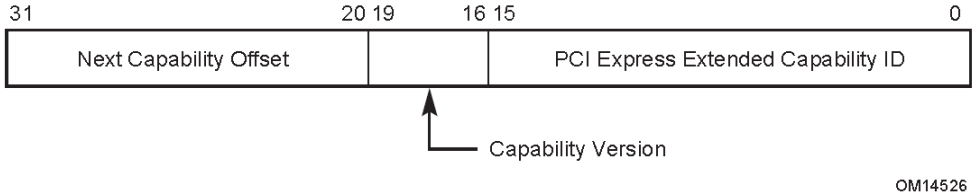


Figure 7-180: Vendor-Specific Extended Capability Header

Table 7-147: Vendor-Specific Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. Extended Capability ID for the Vendor-Specific Extended Capability is 000Bh.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.	RO

Bit Location	Register Description	Attributes
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh.	RO

7.9.5.2 *Vendor-Specific Header (Offset 04h)*

Figure 7-181 details allocation of register fields in the vendor-specific header; Table 7-148 provides the respective bit definitions.

Vendor-specific software must qualify the associated Vendor ID of the PCI Express Function or RCRB before attempting to interpret the values in the VSEC ID or VSEC Rev fields.

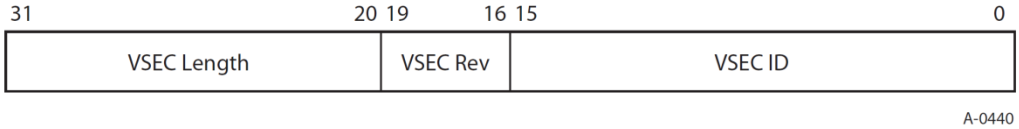


Figure 7-181: Vendor-Specific Header

Table 7-148: Vendor-Specific Header

Bit Location	Register Description	Attributes
15:0	**VSEC ID** – This field is a vendor-defined ID number that indicates the nature and format of the VSEC structure. Software must qualify the Vendor ID before interpreting this field.	RO
19:16	**VSEC Rev** – This field is a vendor-defined version number that indicates the version of the VSEC structure. Software must qualify the Vendor ID and VSEC ID before interpreting this field.	RO
31:20	**VSEC Length** – This field indicates the number of bytes in the entire VSEC structure, including the PCI Express Extended Capability header, the vendor-specific header, and the vendor-specific registers.	RO

7.9.6 Designated Vendor-Specific Extended Capability (DVSEC)

The PCI Express Designated Vendor-Specific Extended Capability (DVSEC) is an optional Extended Capability that is permitted to be implemented by any PCI Express Function or RCRB. This allows PCI Express component vendors to use the Extended Capability mechanism to expose vendor-specific registers that can be present in components by a variety of vendors.

A single PCI Express Function or RCRB is permitted to contain multiple DVSEC structures.

An example usage is a set of vendor-specific features that are intended to go into an on-going series of components from a collection of vendors. A DVSEC structure can tell vendor-specific software which features a particular component supports, including components developed after the software was released.

Figure 7-182 details allocation of register fields in the DVSEC structure. The structure of the PCI Express Extended Capability header and the Designated Vendor-Specific header is architected by this specification.

The DVSEC Vendor-Specific Register area begins at offset 0Ah.

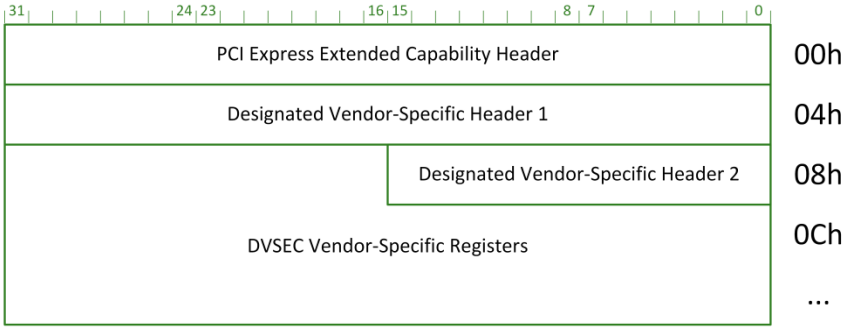


Figure 7-182: Designated Vendor-Specific Extended Capability

7.9.6.1 *Designated Vendor-Specific Extended Capability Header (Offset 00h)*

Figure 7-183 details allocation of register fields in the Designated Vendor-Specific Extended Capability header; Table 7-149 provides the respective bit definitions. Refer to Section 7.9.3 for a description of the PCI Express Extended Capability header. The Extended Capability ID for the Designated Vendor-Specific Extended Capability is 0023h.

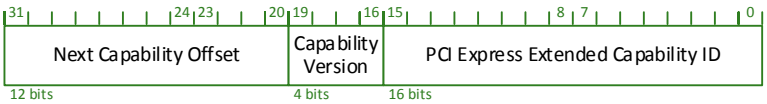


Figure 7-183: Designated Vendor-Specific Extended Capability Header

Table 7-149: Designated Vendor-Specific Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. Extended Capability ID for the Designated Vendor-Specific Extended Capability is 0023h.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh.	RO

7.9.6.2 *Designated Vendor-Specific Header 1 (Offset 04h)*

Figure 7-184 details allocation of register fields in the Designated Vendor-Specific Header 1; Table 7-150 provides the respective bit definitions.

Vendor-specific software must qualify the DVSEC Vendor ID before attempting to interpret the DVSEC Revision field.

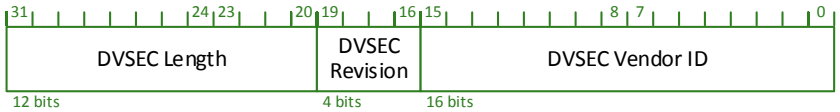


Figure 7-184: Designated Vendor-Specific Header 1

Table 7-150: Designated Vendor-Specific Header 1

Bit Location	Register Description	Attributes
15:0	**DVSEC Vendor ID** – This field is the Vendor ID associated with the vendor that defined the contents of this capability.	RO
19:16	**DVSEC Revision** – This field is a vendor-defined version number that indicates the version of the DVSEC structure. Software must qualify the DVSEC Vendor ID and DVSEC ID before interpreting this field.	RO
31:20	**DVSEC Length** – This field indicates the number of bytes in the entire DVSEC structure, including the PCI Express Extended Capability header, the DVSEC Header 1, DVSEC Header 2, and DVSEC vendor-specific registers.	RO

7.9.6.3 *Designated Vendor-Specific Header 2 (Offset 08h)*

Figure 7-185 details allocation of register fields in the Designated Vendor-Specific Header 2; Table 7-151 provides the respective bit definitions.

Vendor-specific software must qualify the DVSEC Vendor ID before attempting to interpret the DVSEC ID field.

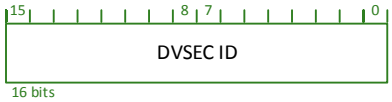


Figure 7-185: Designated Vendor-Specific Header 2

Table 7-151: Designated Vendor-Specific Header 2

Bit Location	Register Description	Attributes
15:0	**DVSEC ID** – This field is a vendor-defined ID that indicates the nature and format of the DVSEC structure. Software must qualify the DVSEC Vendor ID before interpreting this field.	RO

7.9.7 RCRB Header Capability

The PCI Express RCRB Header Capability is an optional Extended Capability that may be implemented in an RCRB to provide a Vendor ID and Device ID for the RCRB and to permit the management of parameters that affect the behavior of Root Complex functionality associated with the RCRB.

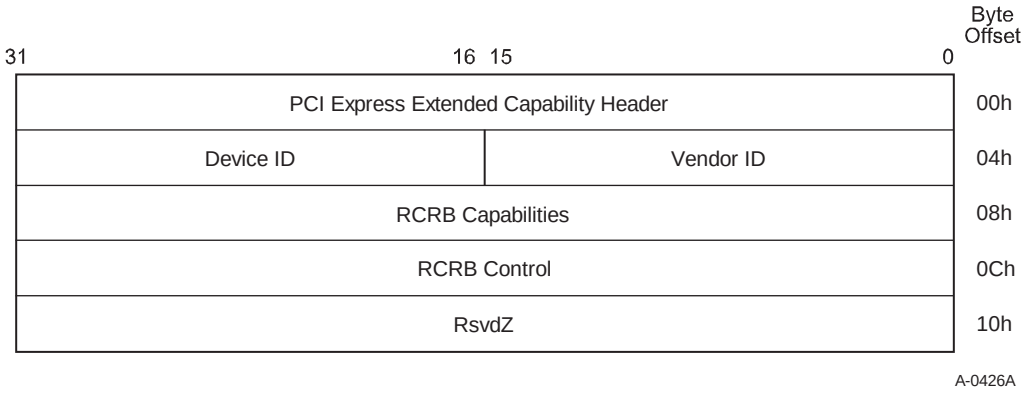


Figure 7-186: Root Complex Features Capability Structure

7.9.7.1 *RCRB Header Extended Capability Header (Offset 00h)*

Figure 7-187 details allocation of register fields in the RCRB Header Extended Capability header. Table 7-152 provides the respective bit definitions. Refer to Section 7.6.3 for a description of the PCI Express Enhanced Capabilities header. The Extended Capability ID for the RCRB Header Capability is 000Ah.

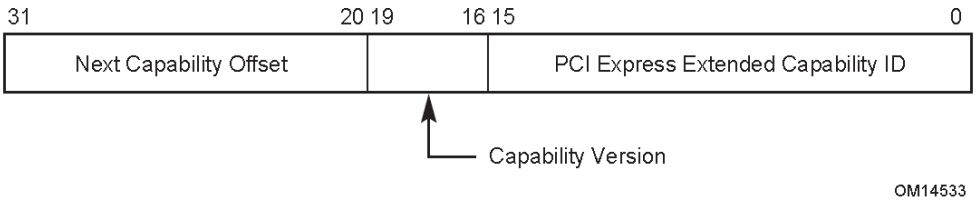


Figure 7-187: RCRB Header Extended Capability Header

Table 7-152: RCRB Header Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. Extended Capability ID for the RCRB Header Capability is 000Ah.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh.	RO

7.9.7.2 *Vendor ID (Offset 04h) and Device ID (Offset 06h)*

Figure 7-188 details allocation of register fields in the RCRB Capabilities register; Table 7-153 provides the respective bit definitions.

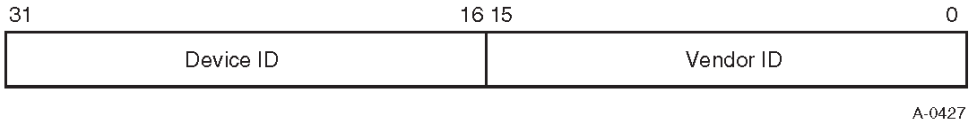


Figure 7-188: Vendor ID and Device ID

Table 7-153: Vendor ID and Device ID

Bit Location	Register Description	Attributes
15:0	**Vendor ID** – PCI-SIG assigned. Analogous to the equivalent field in PCI-compatible Configuration Space. This field provides a means to associate an RCRB with a particular vendor.	RO
31:16	**Device ID** – Vendor assigned. Analogous to the equivalent field in PCI-compatible Configuration Space. This field provides a	RO

	means for a vendor to classify a particular RCRB.	

7.9.7.3 RCRB Capabilities (Offset 08h)

Figure 7-189 details allocation of register fields in the RCRB Capabilities register; Table 7-154 provides the respective bit definitions.

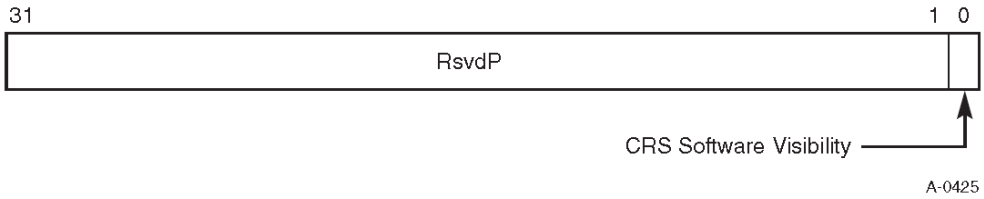


Figure 7-189: RCRB Capabilities

Table 7-154: RCRB Capabilities

Bit Location	Register Description	Attributes
0	**CRS Software Visibility** – When Set, this bit indicates that the Root Complex is capable of returning Configuration Request Retry Status (CRS) Completion Status to software for all Root Ports and integrated devices associated with this RCRB (see Section 2.3.1).	RO

7.9.7.4 RCRB Control (Offset 0Ch)

Figure 7-190 details allocation of register fields in the RCRB Control register; Table 7-155 provides the respective bit definitions.

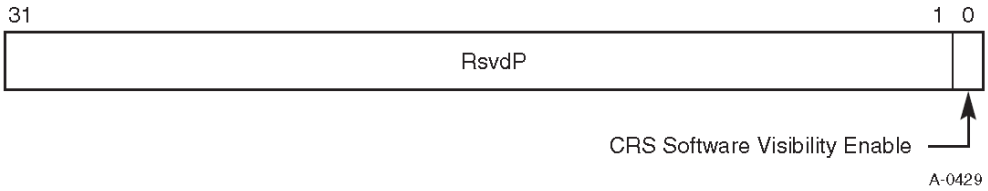


Figure 7-190: RCRB Control

Table 7-155: RCRB Control

Bit Location	Register Description	Attributes
0	**CRS Software Visibility Enable** – When Set, this bit enables the Root Complex to return Configuration Request Retry Status (CRS) Completion Status to software for all Root Ports and integrated devices associated with this RCRB (see Section 2.3.1). RCRBs that do not implement this capability must hardwire this bit to 0b. Default value of this bit is 0b.	RW

7.9.8 PCI Express Root Complex Link Declaration Capability

The PCI Express Root Complex Link Declaration Capability is an optional Capability that is permitted to be implemented by Root Ports, RCiEPs, or RCRBs to declare a Root Complex's internal topology.

A Root Complex consists of one or more following elements:

- PCI Express Root Port
- A default system Egress Port or an internal sink unit such as memory (represented by an RCRB)
- Internal Data Paths/Links (represented by an RCRB on either side of an internal Link)
- Integrated devices
- Functions

A Root Complex Component is a logical aggregation of the above described Root Complex elements. No single element can be part of more than one Root Complex Component. Each Root Complex Component must have a unique Component ID.

A Root Complex is represented either as an opaque Root Complex or as a collection of one or more Root Complex Components.

The PCI Express Root Complex Link Declaration Capability is permitted to be present in a Root Complex element's Configuration Space or RCRB. It declares Links from the respective element to other elements of the same Root Complex Component or to an element in another Root Complex Component. The Links are required to be declared bidirectional such that each valid data path from one element to another has corresponding Link entries in the Configuration Space (or RCRB) of both elements.

The PCI Express Root Complex Link Declaration Capability is permitted to also declare an association between a Configuration Space element (Root Port or RCiEP) and an RCRB Header Capability (see Section 7.9.7) contained in an RCRB that affects the behavior of the Configuration Space element. Note that an RCRB Header association is not declared bidirectional; the association is only declared by the Configuration Space element and not by the target RCRB.

IMPLEMENTATION NOTE

Topologies to Avoid

Topologies that create more than one data path between any two Root Complex elements (either directly or through other Root Complex elements) may not be able to support bandwidth allocation in a standard manner. The description of how traffic is routed through such a topology is implementation specific, meaning that general purpose-operating systems may not have enough information about such a topology to correctly support bandwidth allocation. In order to circumvent this problem, these operating systems may require that a single RCRB element (of type Internal Link) not declare more than one Link to a Root Complex Component other than the one containing the RCRB element itself.

The PCI Express Root Complex Link Declaration Capability, as shown in Figure 7-191, consists of the PCI Express Extended Capability header and Root Complex Element Self Description followed by one or more Root Complex Link Entries.

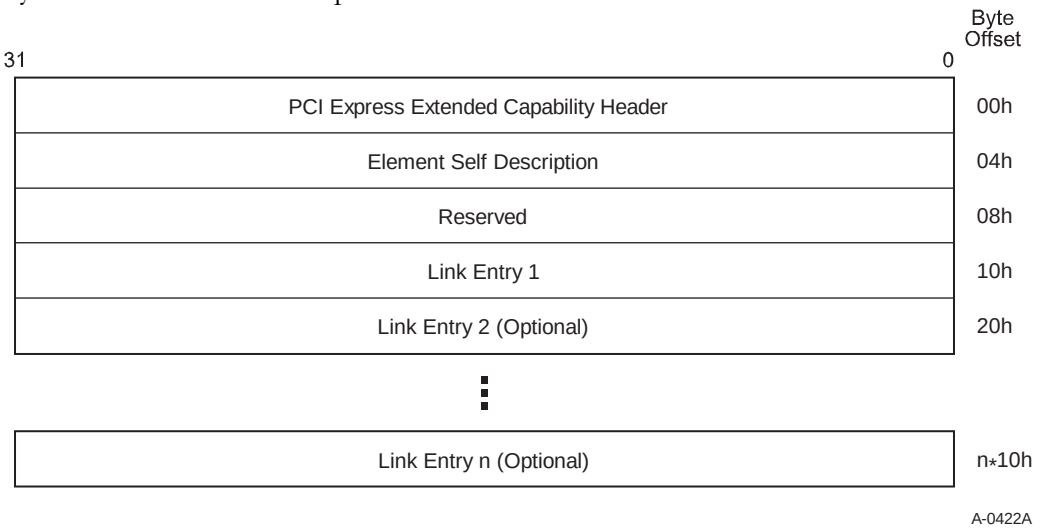


Figure 7-191: PCI Express Root Complex Link Declaration Capability

7.9.8.1 *Root Complex Link Declaration Extended Capability Header (Offset 00h)*

The Extended Capability ID for the Root Complex Link Declaration Capability is 0005h.

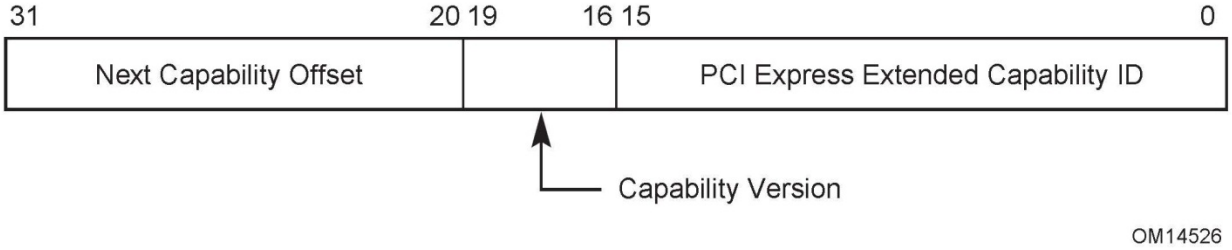


Figure 7-192: Root Complex Link Declaration Extended Capability Header

Table 7-156: Root Complex Link Declaration Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. The Extended Capability ID for the Root Complex Link Declaration Extended Capability is 0005h.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.	RO

Bit Location	Register Description	Attributes
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh. The bottom 2 bits of this offset are Reserved and must be implemented as 00b although software must mask them to allow for future uses of these bits.	RO

7.9.8.2 *Element Self Description (Offset 04h)*

The Element Self Description register provides information about the Root Complex element containing the Root Complex Link Declaration Capability.

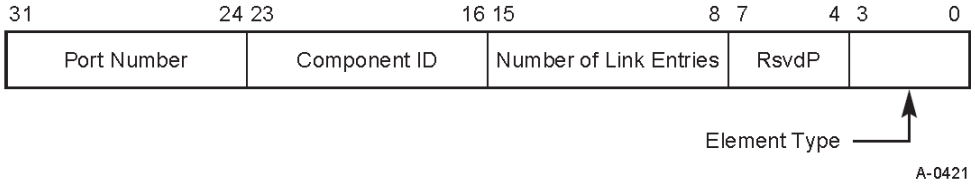


Figure 7-193: Element Self Description Register

Table 7-157: Element Self Description Register

Bit Location	Register Description	Attributes
3:0	**Element Type** – This field indicates the type of the Root Complex Element. Defined encodings are: 0h Configuration Space Element 1h System Egress Port or internal sink (memory) 2h Internal Root Complex Link 3h-Fh Reserved	RO
15:8	**Number of Link Entries** – This field indicates the number of Link entries following the Element Self Description. This field must report a value of 01h or higher.	HwInit
23:16	**Component ID** – This field identifies the Root Complex Component that contains this Root Complex Element. Component IDs must start at 01h, as a value of 00h is Reserved.	HwInit

Bit Location	Register Description	Attributes
31:24	**Port Number** – This field specifies the Port Number associated with this element with respect to the Root Complex Component that contains this element. An element with a Port Number of 00h indicates the default Egress Port to configuration software.	HwInit

7.9.8.3 *Link Entries*

Link Entries start at offset 10h of the PCI Express Root Complex Link Declaration Capability structure. Each Link Entry consists of a Link description followed by a 64-bit Link address at offset 08h from the start of Link entry identifying the target element for the declared Link. A Link Entry declares an internal Link to another Root Complex Element.

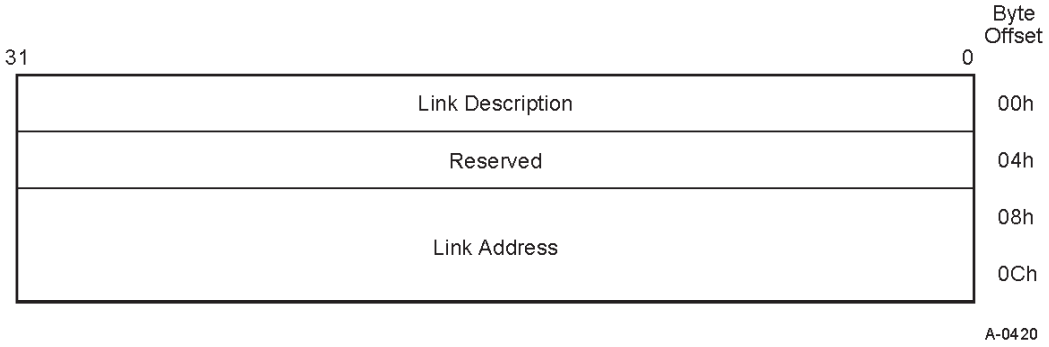


Figure 7-194: Link Entry

7.9.8.3.1 Link Description

The Link Description is located at offset 00h from the start of a Link Entry and is defined as follows:

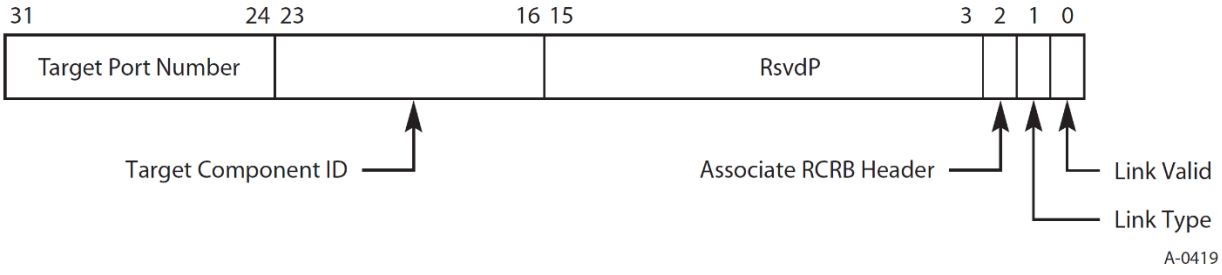


Figure 7-195: Link Description Register

Table 7-158: Link Description Register

Bit Location	Register Description	Attributes
0	**Link Valid** – When Set, this bit indicates that the Link Entry specifies a valid Link. Link entries that do not have either this bit Set or the Associate RCRB Header bit Set (or both) are ignored by software.	HwInit
1	**Link Type** – This bit indicates the target type of the Link and defines the format of the Link address field. Defined Link Type values are: 0b – Link points to memory-mapped space[140] (for RCRB). The Link address specifies the 64-bit base address of the target RCRB. 1b – Link points to Configuration Space (for a Root Port or RCiEP). The Link address specifies the configuration address (PCI Segment Group, Bus, Device, Function) of the target element.	HwInit
2	**Associate RCRB Header** – When Set, this bit indicates that the Link Entry associates the declaring element with an RCRB Header Capability in the target RCRB. Link entries that do not have either this bit Set or the Link Valid bit Set (or both) are ignored by software. The Link Type bit must be Clear when this bit is Set.	HwInit
23:16	**Target Component ID** – This field identifies the Root Complex Component that is targeted by this Link entry. Components IDs must start at 01h, as a value of 00h is Reserved	HwInit
31:24	**Target Port Number** – This field specifies the Port Number associated with the element targeted by this Link entry; the target Port Number is with respect to the Root Complex Component (identified by the Target Component ID) that contains the target element.	HwInit

7.9.8.3.2 Link Address

The Link address is a HwInit field located at offset 08h from the start of a Link Entry that identifies the target element for the Link entry. For a Link of Link Type 0 in its Link Description, the Link address specifies the memory-mapped base address of RCRB. For a Link of Link Type 1 in its Link Description, the Link address specifies the Configuration Space address of a PCI Express Root Port or an RCiEP.

[140] The memory-mapped space for accessing an RCRB is not the same as Memory Space, and must not overlap with Memory Space.

7.9.8.3.2.1 Link Address for Link Type 0

For a Link pointing to a memory-mapped RCRB (Link Type bit = 0), the first DWORD specifies the lower 32 bits of the RCRB base address of the target element as shown below; bits 11:0 are hardwired to 000h and Reserved for future use. The second DWORD specifies the high order 32 bits (63:32) of the RCRB base address of the target element.

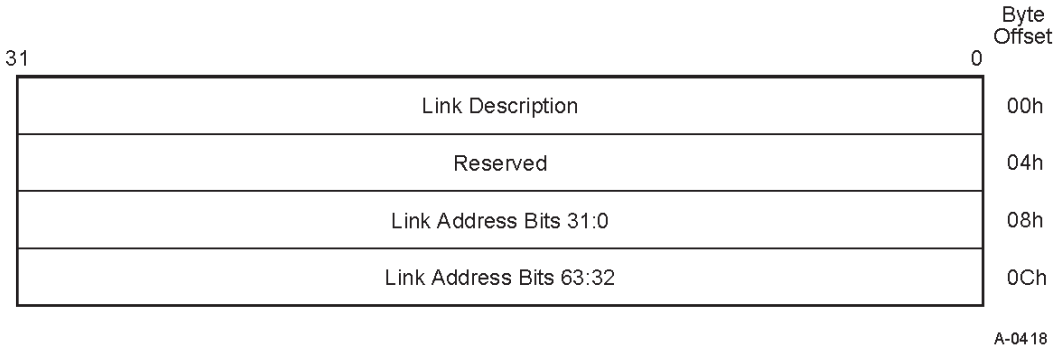


Figure 7-196: Link Address for Link Type 0

7.9.8.3.2.2 Link Address for Link Type 1

For a Link pointing to the Configuration Space of a Root Complex element (Link Type bit = 1), bits in the first DWORD specify the Bus, Device, and Function Number of the target element. As shown in Figure 7-197, bits 2:0 (N) encode the number of bits n associated with the Bus Number, with N = 000b specifying n = 8 and all other encodings specifying n = <value of N>. Bits 11:3 are Reserved and hardwired to 0. Bits 14:12 specify the Function Number, and bits 19:15 specify the Device Number. Bits (19 + n):20 specify the Bus Number, with $1 \leq n \leq 8$.

Bits 31:(20 + n) of the first DWORD together with the second DWORD optionally identify the target element's hierarchy for systems implementing the PCI Express Enhanced Configuration Access Mechanism by specifying bits 63:(20 + n) of the memory-mapped Configuration Space base address of the PCI Express hierarchy associated with the targeted element; single hierarchy systems that do not implement more than one memory mapped Configuration Space are allowed to report a value of zero to indicate default Configuration Space.

A Configuration Space base address [63:(20 + n)] equal to zero indicates that the Configuration Space address defined by bits (19 + n):12 (Bus Number, Device Number, and Function Number) exists in the default PCI Segment Group; any non-zero value indicates a separate Configuration Space base address.

Software must not use n outside the context of evaluating the Bus Number and memory-mapped Configuration Space base address for this specific target element. In particular, n does not necessarily indicate the maximum Bus Number supported by the associated PCI Segment Group.

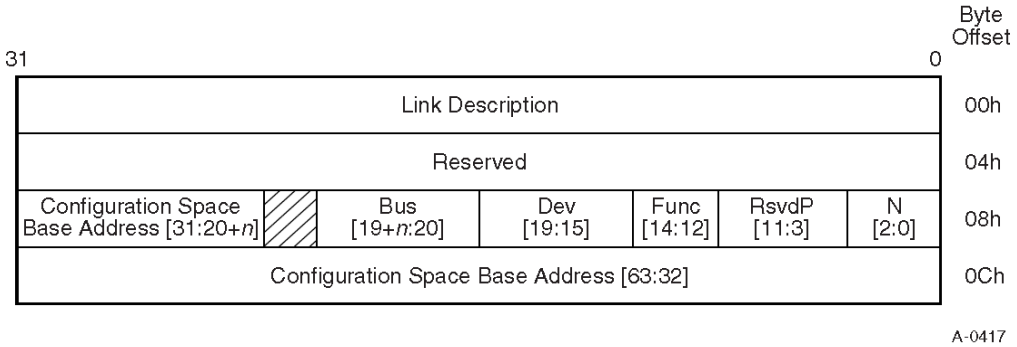


Figure 7-197: Link Address for Link Type 1

Table 7-159: Link Address for Link Type 1

Bit Location	Register Description	Attributes
2:0	**N** – Encoded number of Bus Number bits	HwInit
14:12	**Function Number**	HwInit
19:15	**Device Number**	HwInit
(19 + *n*):20	**Bus Number**	HwInit
63:(20 + *n*)	**PCI Express Configuration Space Base Address** (1 ≤ *n* ≤ 8) Note: A Root Complex that does not implement multiple Configuration Spaces is allowed to report this field as 0.	HwInit

7.9.9 PCI Express Root Complex Internal Link Control Capability

The PCI Express Root Complex Internal Link Control Capability is an optional Capability that controls an internal Root Complex Link between two distinct Root Complex Components. This Capability is valid for RCRBs that declare an Element Type field as Internal Root Complex Link in the Element Self-Description register of the Root Complex Link Declaration Capability structure.

The Root Complex Internal Link Control Capability structure is defined as shown in Figure 7-198.

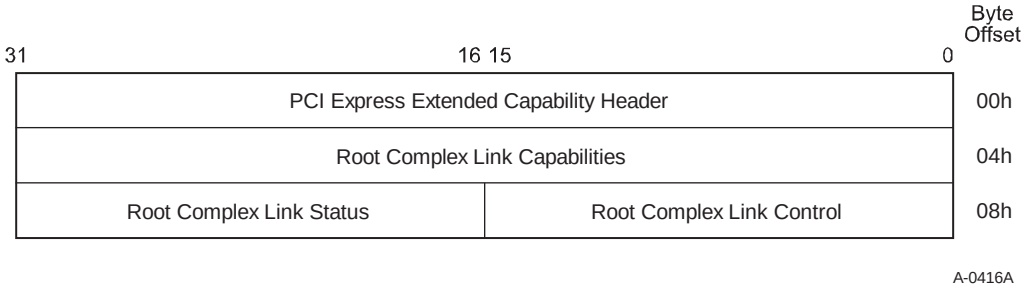


Figure 7-198: Root Complex Internal Link Control Capability

7.9.9.1 *Root Complex Internal Link Control Extended Capability Header (Offset 00h)*

The Extended Capability ID for the Root Complex Internal Link Control Capability is 0006h.

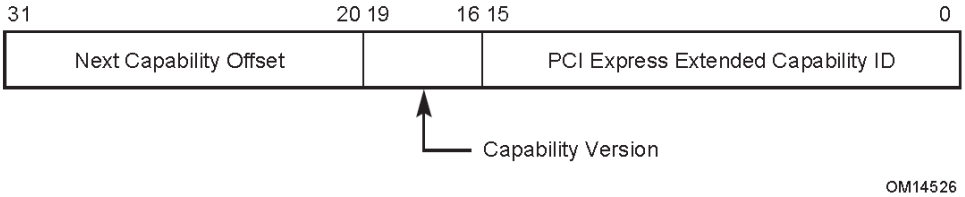


Figure 7-199: Root Internal Link Control Extended Capability Header

Table 7-160: Root Complex Internal Link Control Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. The Extended Capability ID for the Root Complex Internal Link Control Extended Capability is 0006h.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh. The bottom 2 bits of this offset are Reserved and must be implemented as 00b although software must mask them to allow for future uses of these bits.	RO

7.9.9.2 *Root Complex Link Capabilities Register (Offset 04h)*

The Root Complex Link Capabilities register identifies capabilities for this Link.

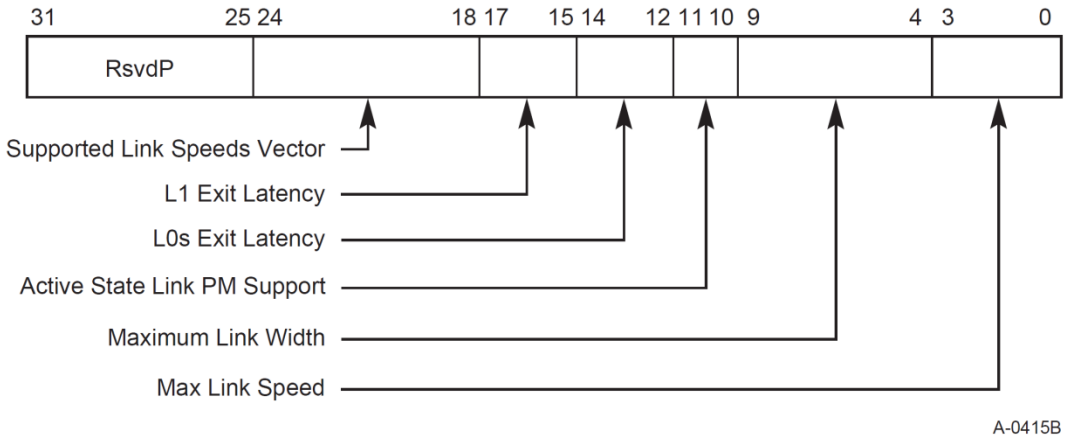


Figure 7-200: Root Complex Link Capabilities Register

Table 7-161: Root Complex Link Capabilities Register

Bit Location	Register Description	Attributes
3:0	**Max Link Speed** – This field indicates the maximum Link speed of the associated Link. The encoded value specifies a bit location in the Supported Link Speeds Vector (in the Root Complex Link Capabilities register) that corresponds to the maximum Link speed. Defined encodings are: 0001b Supported Link Speeds Vector field bit 0 0010b Supported Link Speeds Vector field bit 1 0011b Supported Link Speeds Vector field bit 2 0100b Supported Link Speeds Vector field bit 3 0101b Supported Link Speeds Vector field bit 4 0110b Supported Link Speeds Vector field bit 5 0111b Supported Link Speeds Vector field bit 6 All other encodings are reserved. A Root Complex that does not support this feature must report 0000b in this field.	RO

Bit Location	Register Description	Attributes
9:4	**Maximum Link Width** – This field indicates the maximum width of the given Link. Defined encodings are: 00 0001b x1 00 0010b x2 00 0100b x4 00 1000b x8 00 1100b x12 01 0000b x16 10 0000b x32 All other encodings are Reserved. A Root Complex that does not support this feature must report 00 0000b in this field.	RO
11:10	**Active State Power Management (ASPM) Support** – This field indicates the level of ASPM supported on the given Link. Defined encodings are: 00b No ASPM Support 01b L0s Supported 10b L1 Supported 11b L0s and L1 Supported	RO
14:12	**L0s Exit Latency** – This field indicates the L0s exit latency for the given Link. The value reported indicates the length of time this Port requires to complete transition from L0s to L0. If L0s is not supported, the value is undefined. Defined encodings are: 000b Less than 64 ns 001b 64 ns to less than 128 ns 010b 128 ns to less than 256 ns 011b 256 ns to less than 512 ns 100b 512 ns to less than 1 µs 101b 1 µs to less than 2 µs 110b 2 µs to 4 µs 111b More than 4 µs	RO

Bit Location	Register Description	Attributes
17:15	**L1 Exit Latency** – This field indicates the L1 exit latency for the given Link. The value reported indicates the length of time this Port requires to complete transition from ASPM L1 to L0. If ASPM L1 is not supported, the value is undefined. Defined encodings are: 000b Less than 1 µs 001b 1 µs to less than 2 µs 010b 2 µs to less than 4 µs 011b 4 µs to less than 8 µs 100b 8 µs to less than 16 µs 101b 16 µs to less than 32 µs 110b 32 µs to 64 µs 111b More than 64 µs	RO
24:18	Supported Link Speeds Vector – This field indicates the supported Link speed(s) of the associated Link. For each bit, a value of 1b indicates that the corresponding Link speed is supported; otherwise, the Link speed is not supported. See Section 8.2.1 for further requirements. Bit definitions within this field are: Bit 0 2.5 GT/s Bit 1 5.0 GT/s Bit 2 8.0 GT/s Bit 3 16.0 GT/s Bits 6:4 RsvdP	RO

IMPLEMENTATION NOTE

Supported Link Speeds With Earlier Hardware

Hardware components compliant to versions prior to the *PCI Express Base Specification, Revision 3.0* did not implement the Supported Link Speeds Vector field and instead returned 0000 000b in bits 24:18.

For software to determine the supported Link speeds for components where this field is contains 0000 000b, software can read bits 3:0 of the Root Complex Link Capabilities register (now defined to be the Max Link Speed field), and interpret the value as follows:

0001b2.5 GT/s Link speed supported

0010b5.0 GT/s and 2.5 GT/s Link speeds supported

For such components, the same encoding is also used for the values for the Current Link Speed field (in the Root Complex Link Status register).

IMPLEMENTATION NOTE

Software Management of Link Speeds With Future Hardware

It is strongly encouraged that software primarily utilize the Supported Link Speeds Vector instead of the Max Link Speed field, so that software can determine the exact set of supported speeds on current and future hardware. This can avoid software being confused if a future specification defines Links that do not require support for all slower speeds.

7.9.9.3 *Root Complex Link Control Register (Offset 08h)*

The Root Complex Link Control register controls parameters for this internal Link.

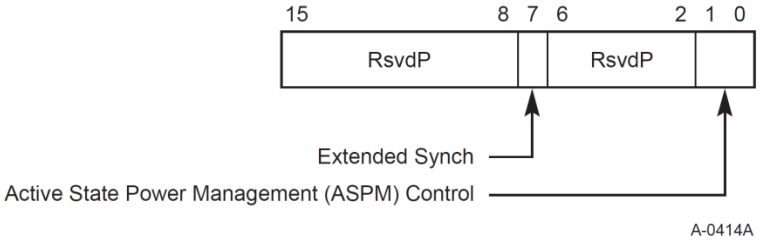


Figure 7-201: Root Complex Link Control Register

Table 7-162: Root Complex Link Control Register

Bit Location	Register Description	Attributes
1:0	**Active State Power Management (ASPM) Control** – This field controls the level of ASPM enabled on the given Link. Defined encodings are: 00b Disabled 01b L0s Entry Enabled 10b L1 Entry Enabled 11b L0s and L1 Entry Enabled Note: "L0s Entry Enabled" enables the Transmitter to enter L0s. If L0s is supported, the Receiver must be capable of entering L0s even when the Transmitter is disabled from entering L0s (00b or 10b). Default value of this field is implementation specific. Software must not enable L0s in either direction on a given Link unless components on both sides of the Link each support L0s, as indicated by their ASPM Support field values. Otherwise, the result is undefined. ASPM L1 must be enabled by software in the Upstream component on a Link prior to enabling ASPM L1 in the Downstream component on that Link. When disabling ASPM L1, software must disable ASPM L1 in the Downstream component on a Link prior to disabling ASPM L1 in the Upstream component on that Link. ASPM L1 must only be enabled on the Downstream component if both components on a Link support ASPM L1. A Root Complex that does not support this feature for the given internal Link must hardwire this field to 00b.	RW
7	**Extended Synch** – This bit when Set forces the transmission of additional Ordered Sets when exiting the L0s state (see Section 4.2.4.5) and when in the Recovery state (see Section 4.2.6.4.1). This mode provides external devices (e.g., logic analyzers) monitoring the Link time to achieve bit and Symbol lock before the Link enters the L0 state and resumes communication. A Root Complex that does not support this feature for the given internal Link must hardwire this bit to 0b. Default value for this bit is 0b.	RW

7.9.9.4 *Root Complex Link Status Register (Offset 0Ah)*

The Root Complex Link Status register provides information about Link specific parameters.

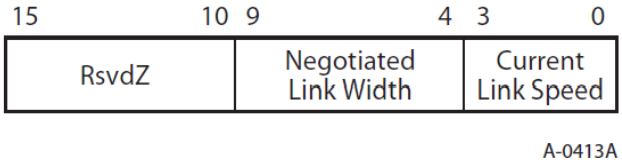


Figure 7-202: Root Complex Link Status Register

Table 7-163: Root Complex Link Status Register

Bit Location	Register Description	Attributes
3:0	**Current Link Speed** – This field indicates the negotiated Link speed of the given Link. The encoded value specifies a bit location in the Supported Link Speeds Vector (in the Root Complex Link Capabilities register) that corresponds to the current Link speed. Defined encodings are: 0001b Supported Link Speeds Vector field bit 0 0010b Supported Link Speeds Vector field bit 1 0011b Supported Link Speeds Vector field bit 2 0100b Supported Link Speeds Vector field bit 3 0101b Supported Link Speeds Vector field bit 4 0110b Supported Link Speeds Vector field bit 5 0111b Supported Link Speeds Vector field bit 6 All other encodings are Reserved. The value in this field is undefined when the Link is not up. A Root Complex that does not support this feature must report 0000b in this field.	RO

Bit Location	Register Description	Attributes
9:4	**Negotiated Link Width** – This field indicates the negotiated width of the given Link. Defined encodings are: 00 0001b x1 00 0010b x2 00 0100b x4 00 1000b x8 00 1100b x12 01 0000b x16 10 0000b x32 All other encodings are Reserved. The value in this field is undefined when the Link is not up. A Root Complex that does not support this feature must hardwire this field to 00 0000b.	RO

7.9.10 PCI Express Root Complex Event Collector Endpoint Association Capability

The PCI Express Root Complex Event Collector Endpoint Association Capability is implemented by Root Complex Event Collectors.

It declares the RCiEPs supported by the Root Complex Event Collector on the same Logical Bus on which the Root Complex Event Collector is located. A Root Complex Event Collector must implement the Root Complex Event Collector Endpoint Association Capability; no other PCI Express device Function is permitted to implement this Capability.

The PCI Express Root Complex Event Collector Endpoint Association Capability, as shown in Figure 7-203, consists of the PCI Express Extended Capability header followed by a DWORD bitmap enumerating RCiEPs associated with the Root Complex Event Collector.

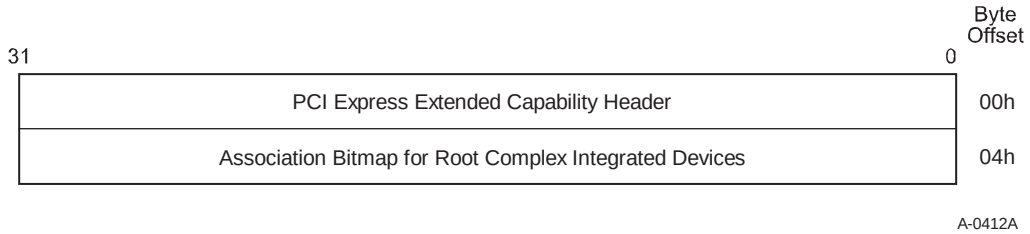


Figure 7-203: Root Complex Event Collector Endpoint Association Capability

7.9.10.1 *Root Complex Event Collector Endpoint Association Extended Capability Header (Offset 00h)*

The Extended Capability ID for the Root Complex Event Collector Endpoint Association Capability is 0007h. Figure 7-204 details allocation of fields in the Root Complex Event Collector

Endpoint Association Extended Capability header; Table 7-164 provides the respective bit definitions.

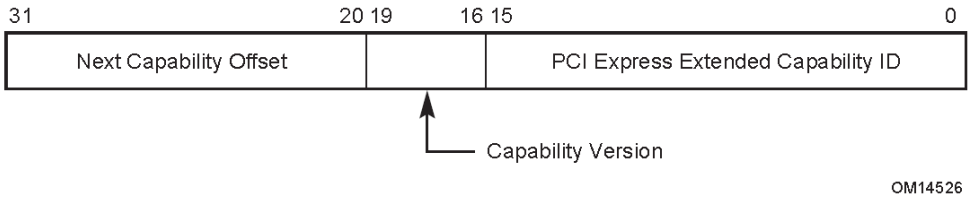


Figure 7-204: Root Complex Event Collector Endpoint Association Extended Capability Header

Table 7-164: Root Complex Event Collector Endpoint Association Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. The Extended Capability ID for the Root Complex Event Collector Endpoint Association Capability is 0007h.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh. The bottom 2 bits of this offset are Reserved and must be implemented as 00b although software must mask them to allow for future uses of these bits.	RO

7.9.10.2 *Association Bitmap for RCiEPs (Offset 04h)*

The Association Bitmap for RCiEPs is a read-only register that sets the bits corresponding to the Device Numbers of RCiEPs supported by the Root Complex Event Collector on the same Logical Bus as the Event Collector itself. The bit corresponding to the Device Number of the Root Complex Event Collector must always be Set.

7.9.11 Multicast Capability

Multicast is an optional normative functionality that is controlled by the Multicast Capability structure. The Multicast Capability is applicable to Root Ports, RCRBs, Switch Ports, Endpoint Functions, and RCiEPs. It is not applicable to PCI Express to PCI/PCI-X Bridges.

In the cases of a Switch or Root Complex or a component that contains multiple Functions, multiple copies of this Capability structure are required – one for each Endpoint Function, Switch Port, or Root Port that supports Multicast. To provide implementation efficiencies, certain fields within each of the Multicast Capability structures within a component must be programmed the same and results are indeterminate if this is not the case. The fields and registers that must be configured with the same values include MC_Enable, MC_Num_Group, MC_Base_Address and MC_Index_Position. These same fields in an Endpoint's Multicast Capability structure must match those configured into a Multicast Capability structure of the Switch or Root Complex above the Endpoint or in which the RCiEP is integrated.

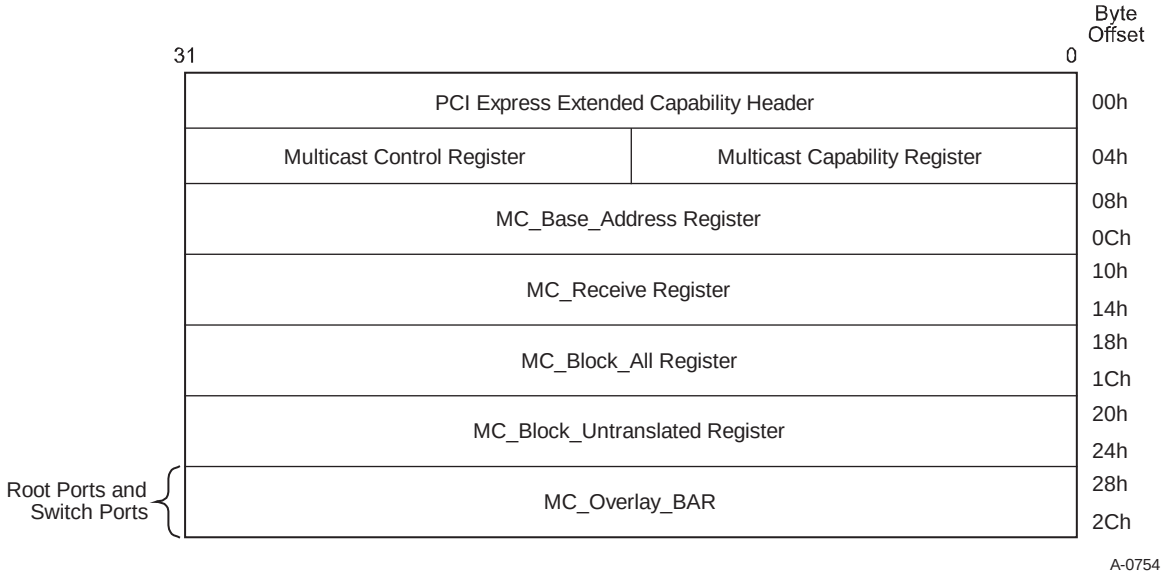


Figure 7-205: Multicast Extended Capability Structure

7.9.11.1 *Multicast Extended Capability Header (Offset 00h)*

Figure 7-206 details allocation of the fields in the Multicast Extended Capability Header and Table 7-165 provides the respective bit definitions.

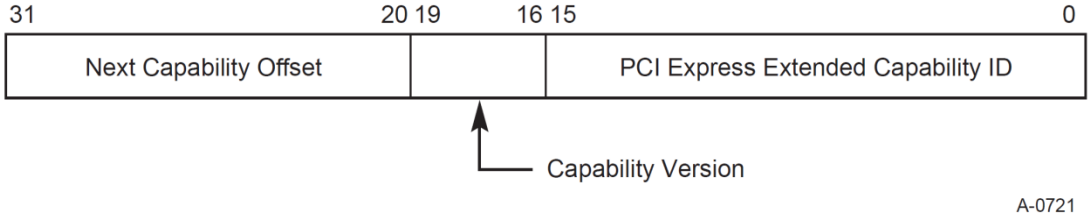


Figure 7-206: Multicast Extended Capability Header

Table 7-165: Multicast Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. PCI Express Extended Capability ID for the Multicast Capability is 0012h.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express Extended Capability structure or 000h if no other items exist in the linked list of Capabilities.	RO

7.9.11.2 *Multicast Capability Register (Offset 04h)*

Figure 7-207 details allocation of the fields in the Multicast Capability register and Table 7-166 provides the respective bit definitions.

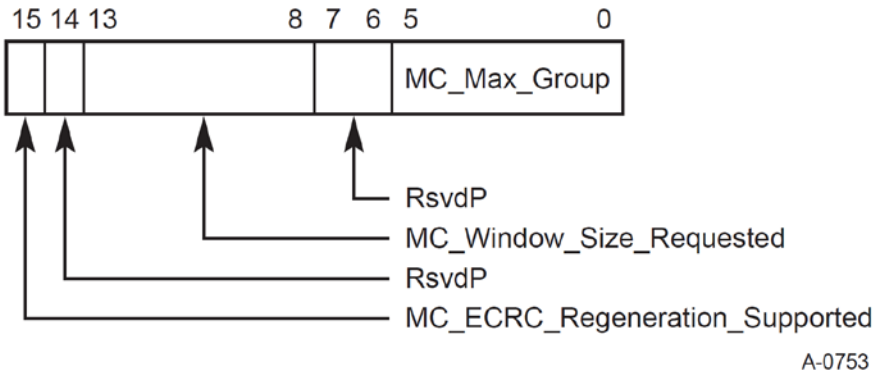


Figure 7-207: Multicast Capability Register

Table 7-166: Multicast Capability Register

Bit Location	Register Description	Attributes
5:0	**MC_Max_Group** – Value indicates the maximum number of Multicast Groups that the component supports, encoded as M-1. A value of 00h indicates that one Multicast Group is supported.	RO
13:8	**MC_Window_Size_Requested** – In Endpoints, the $\log_2$ of the Multicast Window size requested. RsvdP in Switch and Root Ports.	RO

Bit Location	Register Description	Attributes
15	**MC_ECRC_Regeneration_Supported** – If Set, indicates that ECRC regeneration is supported. This bit must not be Set unless the Function supports Advanced Error Reporting, and the ECRC Check Capable bit in the Advanced Error Capabilities and Control register is also Set. However, if ECRC regeneration is supported, its operation is not contingent upon the setting of the ECRC Check Enable bit in the Advanced Error Capabilities and Control register. This bit is applicable to Switch and Root Ports and is RsvdP in all other Functions.	RO/RsvdP

7.9.11.3 *Multicast Control Register (Offset 06h)*

Figure 7-208 details allocation of the fields in the Multicast Control register and Table 7-167 provides the respective bit definitions.

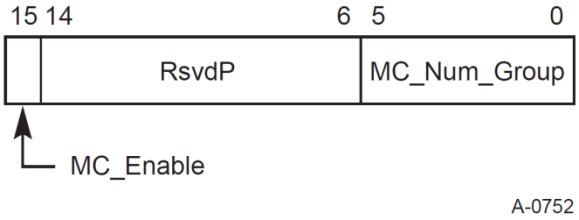


Figure 7-208: Multicast Control Register

Table 7-167: Multicast Control Register

Bit Location	Register Description	Attributes
5:0	**MC_Num_Group** – Value indicates the number of Multicast Groups configured for use, encoded as N-1. The default value of 00 0000b indicates that one Multicast Group is configured for use. Behavior is undefined if value exceeds MC_Max_Group. This parameter indirectly defines the upper limit of the Multicast address range. This field is ignored if MC_Enable is Clear. Default value is 00 0000b.	RW
15	**MC_Enable** – When Set, the Multicast mechanism is enabled for the component. Default value is 0b.	RW

7.9.11.4 *MC_Base_Address Register (Offset 08h)*

The MC_Base_Address register contains the MC_Base_Address and the MC_Index_Position. Figure 7-209 details allocation of the fields in the MC_Base_Address register and Table 7-168 provides the respective bit definitions.

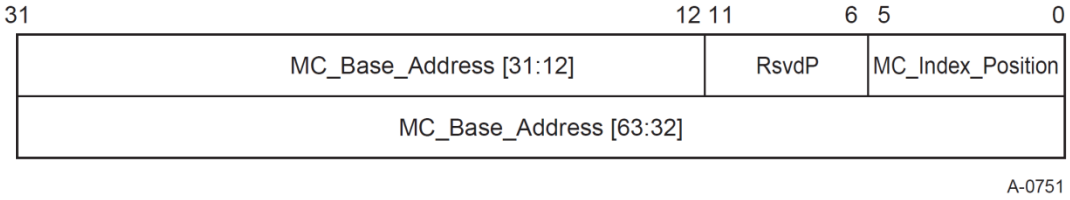


Figure 7-209: MC_Base_Address Register

Table 7-168: MC_Base Address Register

Bit Location	Register Description	Attributes
5:0	**MC_Index_Position** – The location of the LSB of the Multicast Group number within the address. Behavior is undefined if this value is less than 12 and MC_Enable is Set. Default is 0.	RW
63:12	**MC_Base_Address** – The base address of the Multicast address range. The behavior is undefined if MC_Enable is Set and bits in this field corresponding to address bits that contain the Multicast Group number or address bits less than MC_Index_Position are non-zero. Default is 0.	RW

7.9.11.5 *MC_Receive Register (Offset 10h)*

The MC_Receive register provides a bit vector denoting which Multicast groups the Function should accept, or in the case of Switch and Root Complex Ports, forward Multicast TLPs. This register is required in all Functions that implement the MC Capability structure.

Figure 7-210 details allocation of the fields in the MC_Receive register and Table 7-169 provides the respective bit definitions.

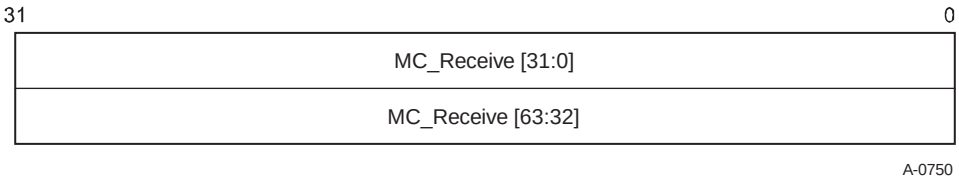


Figure 7-210: MC_Receive Register

Table 7-169: MC_Receive Register

Bit Location	Register Description	Attributes
MC_Max_Group:0	**MC_Receive** – For each bit that's Set, this Function gets a copy of any Multicast TLPs for the associated Multicast Group. Bits above MC_Num_Group are ignored by hardware. Default value of each bit is 0b.	RW
All other bits	Reserved	RsvdP

7.9.11.6 *MC_Block_All Register (Offset 18h)*

The MC_Block_All register provides a bit vector denoting which Multicast groups the Function should block. This register is required in all Functions that implement the MC Capability structure.

Figure 7-211 details allocation of the fields in the MC_Block_All register and Table 7-170 provides the respective bit definitions.

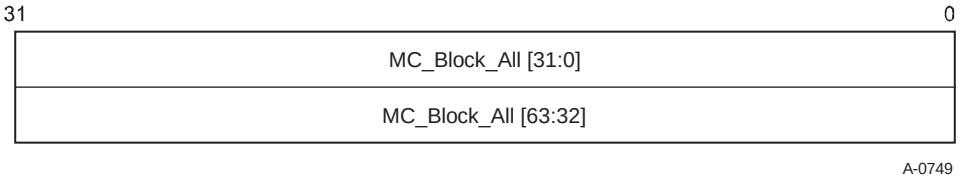


Figure 7-211: MC_Block_All Register

Table 7-170: MC_Block_All Register

Bit Location	Register Description	Attributes
MC_Max_Group:0	**MC_Block_All** – For each bit that is Set, this Function is blocked from sending TLPs to the associated Multicast Group. Bits above MC_Num_Group are ignored by hardware. Default value of each bit is 0b.	RW
All other bits	Reserved	RsvdP

7.9.11.7 *MC_Block_Untranslated Register (Offset 20h)*

The MC_Block_Untranslated register is used to determine whether or not a TLP that includes an Untranslated Address should be blocked. This register is required in all Functions that implement the MC Capability structure. However, an Endpoint Function that does not implement the ATS capability may implement this register as RsvdP.

Figure 7-212 details allocation of the fields in the MC_Block_Untranslated register and Table 7-171 provides the respective bit definitions.

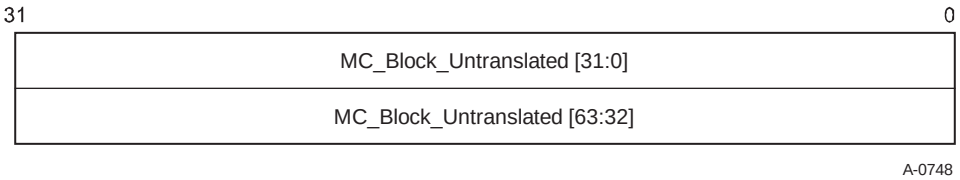


Figure 7-212: MC_Block_Untranslated Register

Table 7-171: MC_Block_Untranslated Register

Bit Location	Register Description	Attributes
MC_Max_Group:0	**MC_Block_Untranslated** – For each bit that is Set, this Function is blocked from sending TLPs containing Untranslated Addresses to the associated MCG. Bits above MC_Num_Group are ignored by hardware. Default value of each bit is 0b.	RW
All other bits	Reserved	RsvdP

7.9.11.8 *MC_Overlay_BAR (Offset 28h)*

The MC_Overlay_BAR is required in Switch and Root Complex Ports that support the Multicast Capability and not implemented in Endpoints. Software must interpret the Device/Port Type Field in the PCI Express Capabilities register to determine if the MC_Overlay_BAR is present in a Function.

The MC_Overlay_BAR specifies the base address of a window in unicast space onto which Multicast TLPs going out an Egress Port are overlaid by a process of address replacement. This allows a single BAR in an Endpoint attached to the Switch or Root Port to be used for both unicast and Multicast traffic. At a Switch Upstream Port, it allows the Multicast address range, or a portion of it, to be overlayed onto host memory.

Figure 7-213 details allocation of the fields in the MC_Overlay BAR and Table 7-172 provides the respective bit definitions.

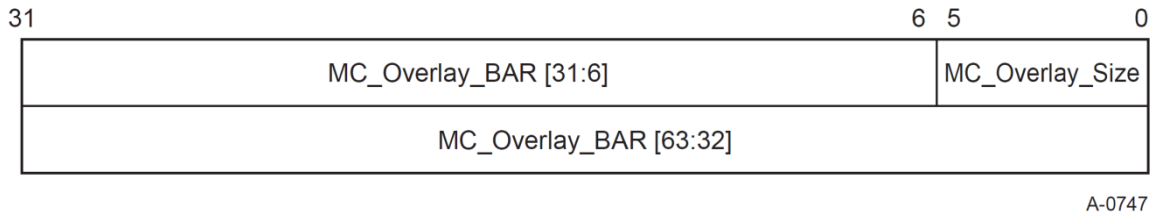


Figure 7-213: MC_Overlay_BAR

Table 7-172: MC_Overlay BAR

Bit Location	Register Description	Attributes
5:0	**MC_Overlay_Size** – If 6 or greater, specifies the size in bytes of the overlay aperture as a power of 2. If less than 6, disables the overlay mechanism. Default value is 00 0000b.	RW
63:6	**MC_Overlay_BAR** – Specifies the base address of the window onto which MC TLPs passing through this Function will be overlaid. Default value is 0.	RW

7.9.12 Dynamic Power Allocation (DPA) Capability

The DPA Capability structure is shown in Figure 7-214.

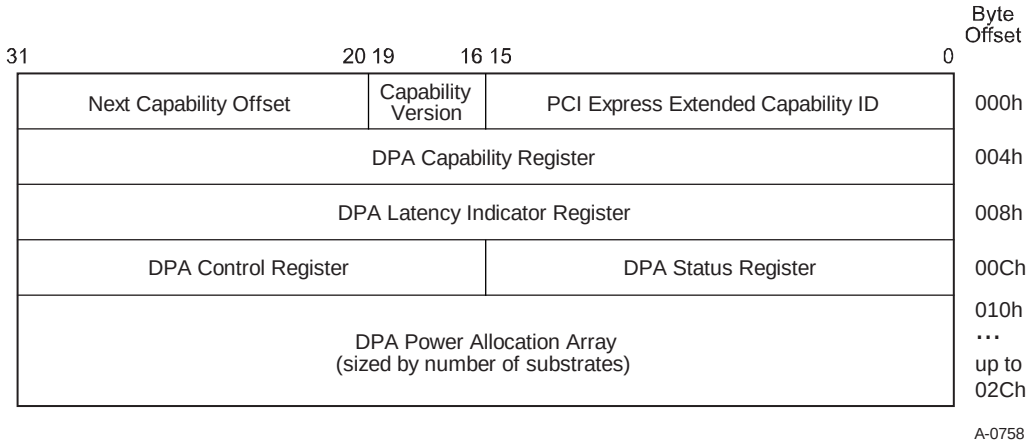


Figure 7-214: Dynamic Power Allocation Capability Structure

7.9.12.1 *DPA Extended Capability Header (Offset 00h)*

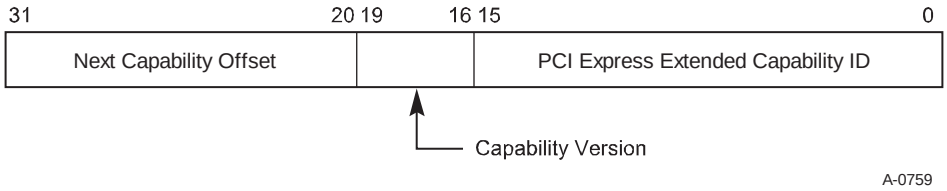


Figure 7-215: DPA Extended Capability Header

Table 7-173: DPA Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. PCI Express Extended Capability ID for the DPA Extended Capability is 0016h.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express Extended Capability structure or 000h if no other items exist in the linked list of Capabilities.	RO

7.9.12.2 *DPA Capability Register (Offset 04h)*

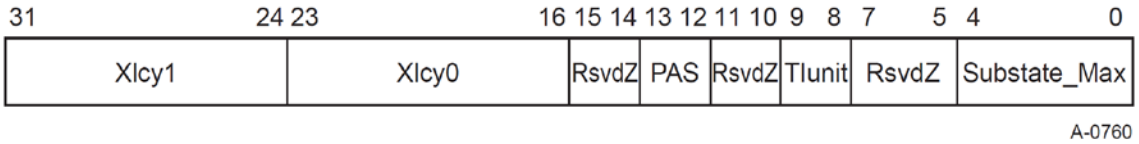


Figure 7-216: DPA Capability Register

Table 7-174: DPA Capability Register

Bit Location	Register Description	Attributes
4:0	**Substate_Max** – Value indicates the maximum substate number, which is the total number of supported substates minus one. A value of 0 0000b indicates support for one substate.	RO
9:8	**Transition Latency Unit (Tlunit)** – A substate's Transition Latency Value is multiplied by the Transition Latency Unit to determine the maximum Transition Latency for the substate. Defined encodings are 00b – 1 ms 01b – 10 ms 10b – 100 ms 11b – Reserved	RO
13:12	**Power Allocation Scale (PAS)** – The encodings provide the scale to determine power allocation per substate in Watts. The value corresponding to the substate in the Substate Power Allocation field is multiplied by this field to determine the power allocation for the substate. Defined encodings are 00b – 10.0x 01b – 1.0x 10b – 0.1x 11b – 0.01x	RO
23:16	**Transition Latency Value 0 (Xlcy0)** – This value is multiplied by the Transition Latency Unit to determine the maximum Transition Latency for the substate	RO
31:24	**Transition Latency Value 1 (Xlcy1)** – This value is multiplied by the Transition Latency Unit to determine the maximum Transition Latency for the substate.	RO

7.9.12.3 *DPA Latency Indicator Register (Offset 08h)*

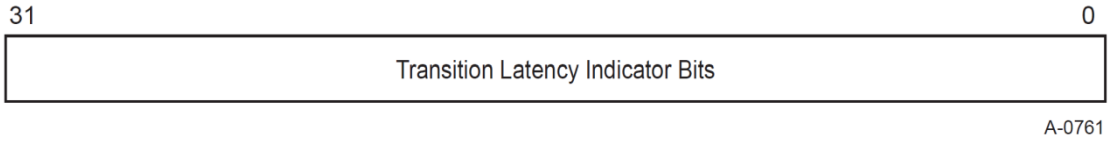


Figure 7-217: DPA Latency Indicator Register

Table 7-175: DPA Latency Indicator Register

Bit Location	Register Description	Attributes
Substate_Max:0	**Transition Latency Indicator Bits** – Each bit indicates which Transition Latency Value is associated with the corresponding substate. A value of 0b indicates Transition Latency Value 0; a value of 1b indicates Transition Latency Value 1.	RO
All other bits	Reserved	RsvdP

7.9.12.4 *DPA Status Register (Offset 0Ch)*

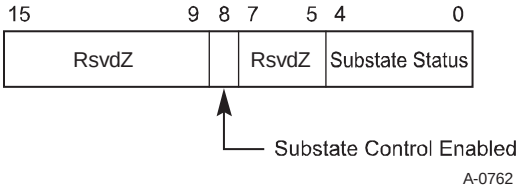


Figure 7-218: DPA Status Register

Table 7-176: DPA Status Register

Bit Location	Register Description	Attributes
4:0	**Substate Status** – Indicates current substate for this Function. Default is 0 0000b.	RO
8	**Substate Control Enabled** – Used by software to disable the Substate Control field in the DPA Control register. Hardware sets this bit following a Conventional Reset or FLR. Software clears this bit by writing a 1b to it. Software is unable to set this bit directly. When this bit is Set, the Substate Control field determines the current substate. When this bit is Clear, the Substate Control field has no effect on the current substate. Default value is 1b.	RW1C

7.9.12.5 *DPA Control Register (Offset 0Eh)*

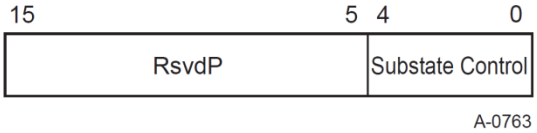


Figure 7-219: DPA Control Register

Table 7-177: DPA Control Register

Bit Location	Register Description	Attributes
4:0	**Substate Control** – Used by software to configure the Function substate. Software writes the substate value in this field to initiate a substate transition. When the Substate Control Enabled bit in the DPA Status register is Set, this field determines the Function substate. When the Substate Control Enabled bit in the DPA Status register is Clear, this field has no effect on the Function substate. Default value is 0 0000b.	RW

7.9.12.6 *DPA Power Allocation Array*

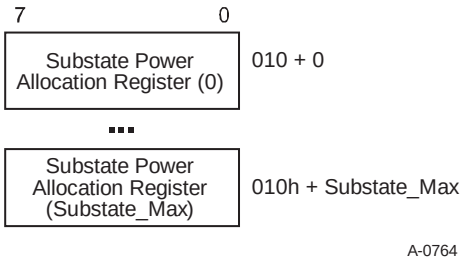


Figure 7-220: DPA Power Allocation Array

Each Substate Power Allocation register indicates the power allocation value for its associated substate. The number of Substate Power Allocation registers implemented must be equal to the number of substates supported by Function, which is Substate_Max plus one.

Table 7-178: Substate Power Allocation Register (0 to Substate_Max)

Bit Location	Register Description	Attributes
7:0	**Substate Power Allocation** – The value in this field is multiplied by the Power Allocation Scale to determine power allocation in Watts for the associated substate.	RO

7.9.13 TPH Requester Capability

The TPH Requester Capability structure is required for all Functions that are capable of generating Request TLPs with TPH. For a Multi-Function Device, this capability must be present in each Function that is capable of generating Request TLPs with TPH.

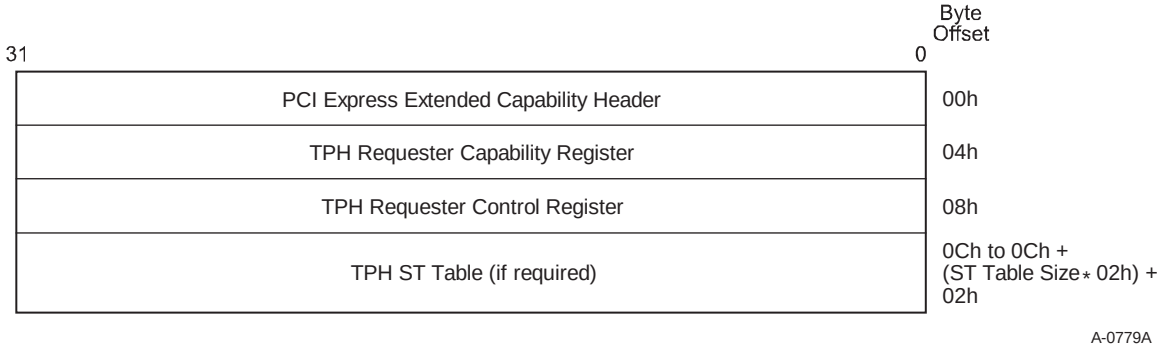


Figure 7-221: TPH Extended Capability Structure

7.9.13.1 *TPH Requester Extended Capability Header (Offset 00h)*

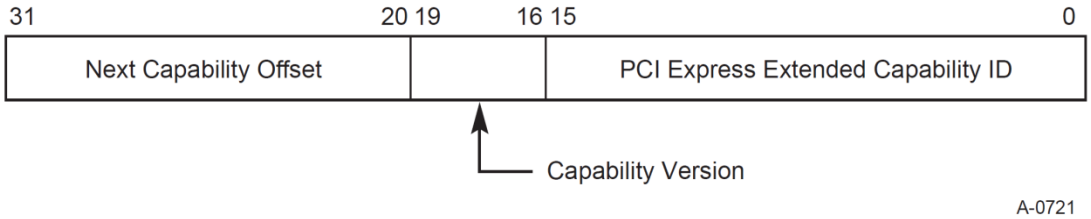


Figure 7-222: TPH Requester Extended Capability Header

Table 7-179: TPH Requester Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. PCI Express Extended Capability ID for the TPH Requester Capability is 0017h**.**	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express Extended Capability structure or 000h if no other items exist in the linked list of Capabilities.	RO

7.9.13.2 *TPH Requester Capability Register (Offset 04h)*

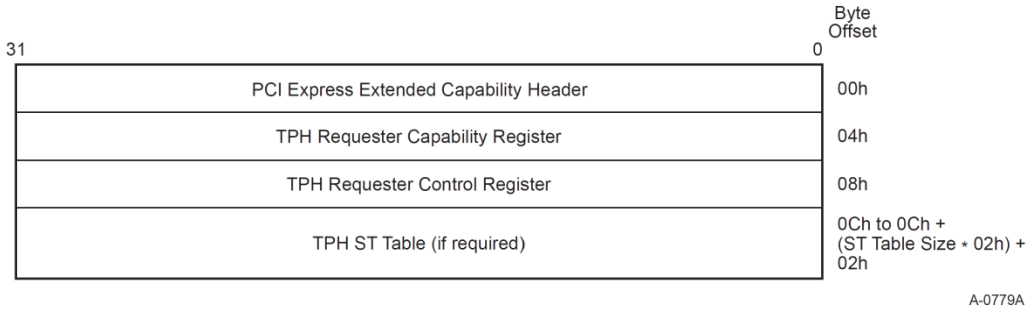


Figure 7-223: TPH Requester Capability Register

Table 7-180: TPH Requester Capability Register

Bit Location	Register Description	Attributes
0	**No ST Mode Supported** – If set indicates that the Function supports the No ST Mode of operation. This mode is required to be supported by all Functions that implement this Capability structure. This bit must have a value of 1b.	RO
1	**Interrupt Vector Mode Supported** – If set indicates that the Function supports the Interrupt Vector Mode of operation.	RO
2	**Device Specific Mode Supported** – If set indicates that the Function supports the Device Specific Mode of operation.	RO
8	**Extended TPH Requester Supported** – If Set indicates that the Function is capable of generating Requests with a TPH TLP Prefix. See Section 2.2.7.1 for additional details.	RO
10:9	**ST Table Location** – Value indicates if and where the ST Table is located. Defined Encodings are: 00b – ST Table is not present. 01b – ST Table is located in the TPH Requester Capability structure. 10b – ST Table is located in the MSI-X Table (see Section 7.7). 11b – Reserved A Function that only supports the No ST Mode of operation must have a value of 00b in this field. A Function may report a value of 10b only if it implements an MSI-X Capability.	RO

Bit Location	Register Description	Attributes
26:16	**ST Table Size** – Value indicates the maximum number of ST Table entries the Function may use. Software reads this field to determine the ST Table Size N, which is encoded as N-1. For example, a returned value of 000 0000 0011b indicates a table size of four entries. There is an upper limit of 64 entries when the ST Table is located in the TPH Requester Capability structure. When the ST Table is located in the MSI-X Table, this value is limited by the size of the MSI-X Table. This field is only applicable for Functions that implement an ST Table as indicated by the ST Table Location field. Otherwise, the value in this field is undefined.	RO

7.9.13.3 *TPH Requester Control Register (Offset 08h)*

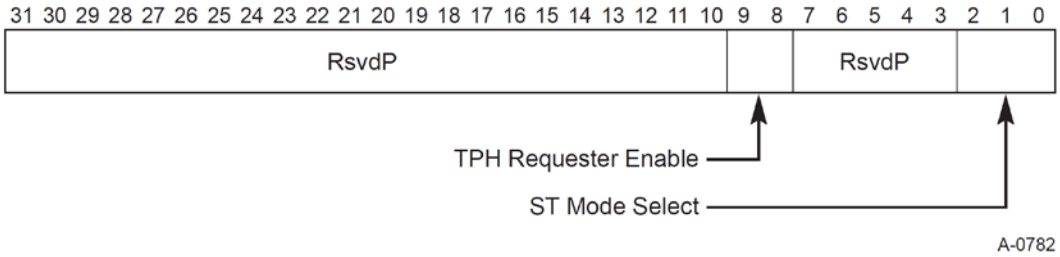


Figure 7-224: TPH Requester Control Register

Table 7-181: TPH Requester Control Register

Bit Location	Register Description	Attributes
2:0	**ST Mode Select** – selects the ST Mode of operation. Defined encodings are: 000b – No ST Mode 001b – Interrupt Vector Mode 010b – Device Specific Mode all other encodings – reserved for future use Functions that support only the No ST Mode of operation must hardwire this field to 000b. Function operation is undefined if software enables a mode of operation that does not correspond to a mode supported by the Function. The default value of this field is 000b. See Section 6.17.3 for details on ST modes of operation.	RW

Bit Location	Register Description	Attributes
9:8	**TPH Requester Enable** – Controls the ability to issue Request TLPs using either TPH or Extended TPH. Defined encodings are: 00b – Function operating as a Requester is not permitted to issue Requests with TPH or Extended TPH. 01b – Function operating as a Requester is permitted to issue Requests with TPH and is not permitted to issue Requests with Extended TPH. 10b – Reserved. 11b – Function operating as a Requester is permitted to issue Requests with TPH and Extended TPH. Functions that advertise that they do not support Extended TPH are permitted to hardwire bit 9 of this field to 0b. The default value of this field is 00b.	RW

7.9.13.4 *TPH ST Table (Starting from Offset 0Ch)*

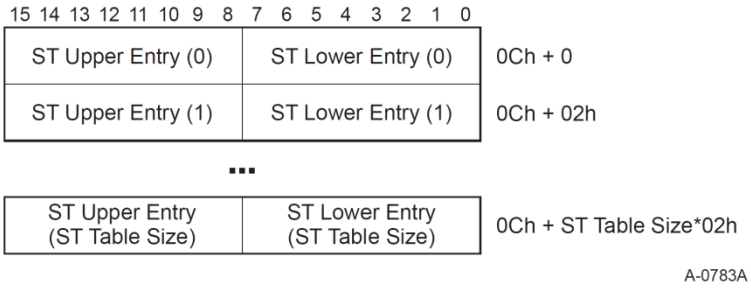


Figure 7-225: TPH ST Table

The ST Table must be implemented in the TPH Requester Capability structure if the value of the ST Table Location field is 01b. For all other values, the ST Entry registers must not be implemented. Each implemented ST Entry is 16 bits. The number of ST Entry registers implemented must be equal to the number of ST Table entries supported by the Function, which is the value of the ST Table Size field plus one.

Table 7-182: TPH ST Table

Bit Location	Register Description	Attributes
7:0	**ST Lower** – This field contains the lower 8 bits of a Steering Tag. Default value of this field is 00h.	RW
15:8	**ST Upper** – If the Function's Extended TPH Requester Supported bit is Set, then this field contains the upper 8 bits of a Steering Tag. Otherwise, this field is RsvdP. Default value of this field is 00h.	RW

7.9.14 LNR Extended Capability

The LN Requester (LNR) Extended Capability is an optional normative capability for Endpoints. All Endpoints that support LN protocol as a Requester must implement this capability. See Section 6.21. This capability may be implemented by any type of Endpoint, but not by any other Function type.

31	0	Byte Offset
PCI Express Extended Capability Header		000h
LNR Control Register	LNR Capability Register	004h

Figure 7-226: LNR Extended Capability

7.9.14.1 *LNR Extended Capability Header (Offset 00h)*

31 20	19 16	15 0
Next Capability Offset		PCI Express Extended Capability ID

Capability Version

Figure 7-227: LNR Extended Capability Header

Table 7-183: LNR Extended Capability Header

Bit Location	Register Description	Attributes
15:0	PCI Express Extended Capability ID – This field is a PCI-SIG defined ID number that indicates the nature and format of the extended capability. PCI Express Extended Capability ID for the LNR Extended Capability is 001Ch.	RO
19:16	Capability Version – This field is a PCI-SIG defined version number that indicates the version of the capability structure present. Must be 1h for this version of the specification.	RO
31:20	Next Capability Offset – This field contains the offset to the next PCI Express Extended Capability structure or 000h if no other items exist in the linked list of capabilities.	RO

7.9.14.2 *LNR Capability Register (Offset 04h)*

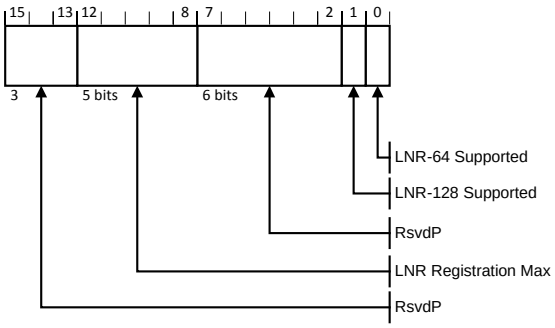


Figure 7-228: LNR Capability Register

Table 7-184: LNR Capability Register

Bit Location	Register Description	Attributes
0	LNR-64 Supported – This bit must be 1b if the Endpoint supports LN protocol for 64-byte cachelines as a Requester; otherwise, must be 0b. See Section 6.21.4 for additional details.	RO
1	LNR-128 Supported – This bit must be 1b if the Endpoint supports LN protocol for 128-byte cachelines as a Requester; otherwise, must be 0b.	RO
12:8	LNR Registration Max – This field, encoded as a power of 2, indicates the maximum number of cachelines that this LN Requester is capable of registering concurrently. For example, a value of 00101b indicates that the LN Requester might be capable of registering up to 32 cachelines (2^5) concurrently, and is capable of registering greater than 16.	RO

7.9.14.3 *LNR Control Register (Offset 06h)*

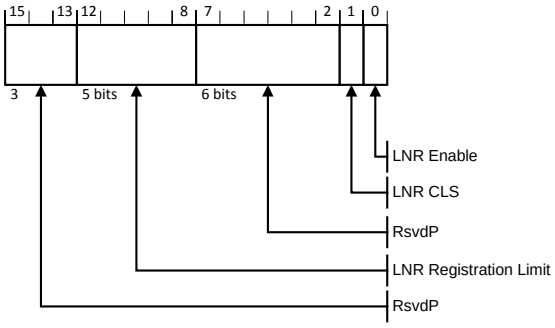


Figure 7-229: LNR Control Register

Table 7-185: LNR Control Register

Bit Location	Register Description	Attributes
0	LNR Enable – When this bit is Set, the Endpoint is enabled to operate as an LN Requester. Software is permitted to Clear this bit at any time. See Section 6.21.4 for requirements regarding the LNR's internal registration state. Default value of this bit is 0b.	RW
1	LNR CLS – This bit controls or indicates the cacheline size used with LN protocol by this Requester. See Section 6.21.4 for restrictions on setting and modifying this bit. If this bit is Clear, the cacheline size is 64 bytes. If this bit is Set, the cacheline size is 128 bytes. If this LN Requester supports only one cacheline size, this bit is permitted to be hardwired to indicate that size. Otherwise, the default value of this bit is 0b.	RW
12:8	LNR Registration Limit – This field, encoded as a power of 2, imposes a limit on the number of cachelines that this LN Requester is permitted to register concurrently. For example, a value of 00100b indicates that the LN Requester must not register more than 16 cachelines (2^4) concurrently. See Section 6.21.4 for restrictions on modifying this field. The default value of this field is 11111b.	RW

7.9.15 DPC Extended Capability

The Downstream Port Containment (DPC) Extended Capability is an optional normative capability that provides a mechanism for Downstream Ports to contain uncorrectable errors and enable software to recover from them. See Section 6.2.10. This capability may be implemented by a Root Port or a Switch Downstream Port. It is not applicable to any other Device/Port type.

If a Downstream Port implements the DPC Extended Capability, that Port must also be capable of reporting the DL_Active state, and indicate so by Setting the Data Link Layer Link Active Reporting Capable bit in the Link Capabilities register. See Section 7.5.3.6.

The various RP PIO registers must be implemented only by Root Ports that support RP Extensions for DPC, as indicated by the RP Extensions for DPC bit in the DPC Capability register.

31	0	Byte Offset
PCI Express Extended Capability Header		000h
DPC Control Register	DPC Capability Register	004h
DPC Error Source ID Register	DPC Status Register	008h
RP PIO Status Register		00Ch
RP PIO Mask Register		010h
RP PIO Severity Register		014h
RP PIO SysError Register		018h
RP PIO Exception Register		01Ch
RP PIO Header Log Register		020h
RP PIO ImpSpec Log Register (Optional)		030h
RP PIO TLP Prefix Log Register (Optional) (Variable Size)		034h

Figure 7-230: DPC Extended Capability

7.9.15.1 *DPC Extended Capability Header (Offset 00h)*

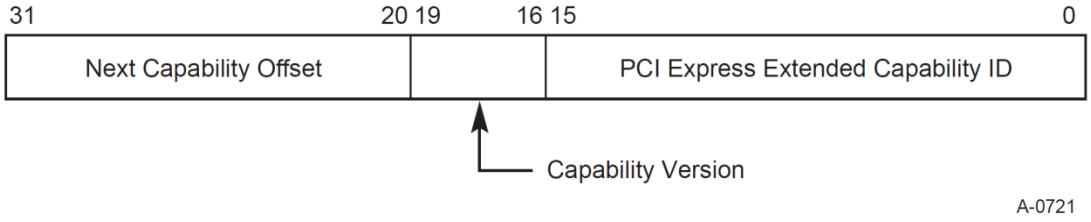


Figure 7-231: DPC Extended Capability Header

Table 7-186: DPC Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the extended capability. PCI Express Extended Capability ID for the DPC Extended Capability is 001Dh.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the capability structure present. Must be 1h for this version of the specification.	RO

Bit Location	Register Description	Attributes
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express Extended Capability structure or 000h if no other items exist in the linked list of capabilities.	RO

7.9.15.2 *DPC Capability Register (Offset 04h)*

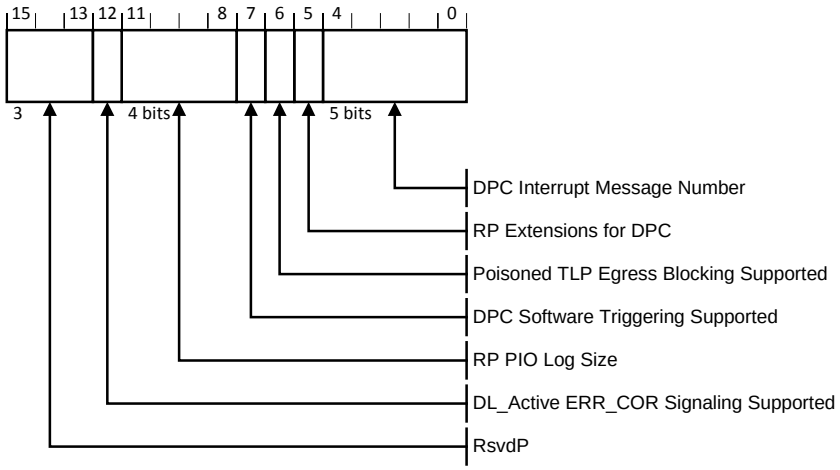


Figure 7-232: DPC Capability Register

Table 7-187: DPC Capability Register

Bit Location	Register Description	Attributes
4:0	**DPC Interrupt Message Number** – This field indicates which MSI/MSI-X vector is used for the interrupt message generated in association with the DPC Capability structure. For MSI, the value in this field indicates the offset between the base Message Data and the interrupt message that is generated. Hardware is required to update this field so that it is correct if the number of MSI Messages assigned to the Function changes when software writes to the Multiple Message Enable field in the MSI Message Control register. For MSI-X, the value in this field indicates which MSI-X Table entry is used to generate the interrupt message. The entry must be one of the first 32 entries even if the Function implements more than 32 entries. For a given MSI-X implementation, the entry must remain constant. If both MSI and MSI-X are implemented, they are permitted to use different vectors, though software is permitted to enable only one mechanism at a time. If MSI-X is enabled, the value in this field must indicate the vector for MSI-X. If MSI is enabled or neither is enabled, the value in this field must indicate the vector for MSI. If software enables both MSI and MSI-X at the same time, the value in this field is undefined.	RO

Bit Location	Register Description	Attributes
5	**RP Extensions for DPC** – If Set, this bit indicates that a Root Port supports a defined set of DPC Extensions that are specific to Root Ports. Switch Downstream Ports must not Set this bit.	RO
6	**Poisoned TLP Egress Blocking Supported** – If Set, this bit indicates that the Root Port or Switch Downstream Port supports the ability to block the transmission of a poisoned TLP from its Egress Port. Root Ports that support RP Extensions for DPC must Set this bit.	RO
7	**DPC Software Triggering Supported** – If Set, this bit indicates that a Root Port or Switch Downstream Port supports the ability for software to trigger DPC. Root Ports that support RP Extensions for DPC must Set this bit.	RO
11:8	**RP PIO Log Size** – This field indicates how many DWORDs are allocated for the RP PIO log registers, comprised by the RP PIO Header Log, the RP PIO ImpSpec Log, and RP PIO TLP Prefix Log. If the Root Port supports RP Extensions for DPC, the value of this field must be 4 or greater; otherwise, the value of this field must be 0. See Sections 7.9.15.11, 7.9.15.12, and 7.9.15.13.	RO
12	**DL_Active ERR_COR Signaling Supported –** If Set, this bit indicates that the Root Port or Switch Downstream Port supports the ability to signal with ERR_COR when the Link transitions to the DL_Active state. Root Ports that support RP Extensions for DPC must Set this bit.	RO

7.9.15.3 *DPC Control Register (Offset 06h)*

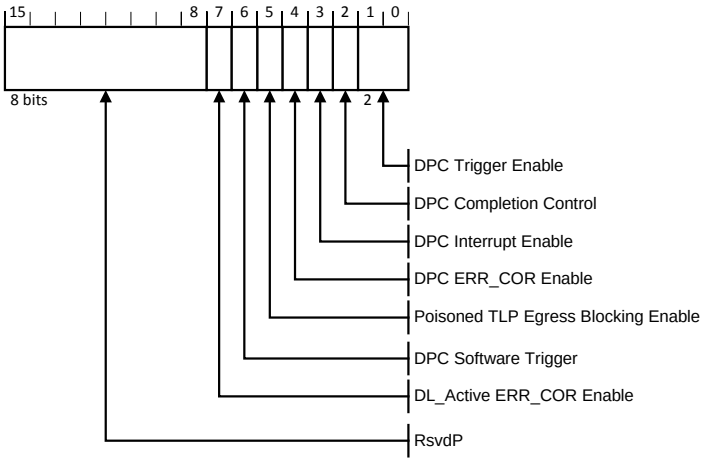


Figure 7-233: DPC Control Register

Table 7-188: DPC Control Register

Bit Location	Register Description	Attributes
1:0	**DPC Trigger Enable** – This field enables DPC and controls the conditions that cause DPC to be triggered. Defined encodings are: 00bDPC is disabled 01bDPC is enabled and is triggered when the Downstream Port detects an unmasked uncorrectable error or when the Downstream Port receives an ERR_FATAL Message. 10bDPC is enabled and is triggered when the Downstream Port detects an unmasked uncorrectable error or when the Downstream Port receives an ERR_NONFATAL or ERR_FATAL Message. 11b Reserved Default value of this field is 00b.	RW
2	**DPC Completion Control** – This bit controls the Completion Status for Completions formed during DPC. See Section 2.9.3. Defined encodings are: 0bCompleter Abort (CA) Completion Status 1bUnsupported Request (UR) Completion Status Default value of this bit is 0b.	RW
3	**DPC Interrupt Enable** – When Set, this bit enables the generation of an interrupt to indicate that DPC has been triggered. See Section 6.2.10.1. Default value of this bit is 0b.	RW
4	**DPC ERR_COR Enable** – When Set, this bit enables the sending of an ERR_COR Message to indicate that DPC has been triggered. See Section 6.2.10.2. Default value of this bit is 0b.	RW
5	**Poisoned TLP Egress Blocking Enable** – This bit must be RW if the Poisoned TLP Egress Blocking Supported bit is Set; otherwise, it is permitted to be hardwired to 0b. Software must not Set this bit unless the Poisoned TLP Egress Blocking Supported bit is Set. When Set, this bit enables the associated Egress Port to block the transmission of poisoned TLPs. See Section 2.7.2.2. Default value of this bit is 0b.	RW / RO
6	**DPC Software Trigger** – This bit must be RW if the DPC Software Triggering Supported bit is Set; otherwise, it is permitted to be hardwired to 0b. If DPC is enabled and the DPC Trigger Status bit is Clear, when software writes 1b to this bit, DPC is triggered. Otherwise, software writing a 1b to this bit has no effect. It is permitted to write 1b to this bit while simultaneously writing updated values to other fields in this register, notably the DPC Trigger Enable field. For this case, the DPC Software Trigger semantics are based on the updated value of the DPC Trigger Enable field. This bit always returns 0b when read.	RW / RO

Bit Location	Register Description	Attributes
7	**DL_Active ERR_COR Enable** – This bit must be RW if the DL_Active ERR_COR Signaling Supported bit is Set; otherwise, it is permitted to be hardwired to 0b. Software must not Set this bit unless the DL_Active ERR_COR Signaling Supported bit is Set. When Set, this bit enables the associated Downstream Port to signal with ERR_COR when the Link transitions to the DL_Active state. See Section 6.2.10.5. Default value of this bit is 0b.	RW / RO

7.9.15.4 *DPC Status Register (Offset 08h)*

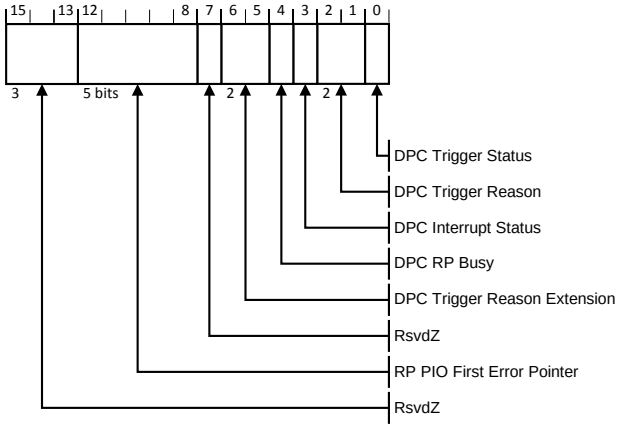


Figure 7-234: DPC Status Register

Table 7-189: DPC Status Register

Bit Location	Register Description	Attributes
0	**DPC Trigger Status** – When Set, this bit indicates that DPC has been triggered, and by definition the Port is “in DPC”. DPC is event triggered. While this bit is Set, hardware must direct the LTSSM to the Disabled State. This bit must be cleared before the LTSSM can be released from the Disabled State, after which the Port is no longer in DPC, and the LTSSM must transition to the Detect State. See Section 6.2.10 for requirements on how long software must leave the Downstream Port in DPC. Once these requirements are met, software is permitted to clear this bit regardless of the state of other status bits associated with the triggering event. After clearing this bit, software must honor timing requirements defined in Section 6.6.1 with respect to the first Configuration Read following a Conventional Reset. Default value of this bit is 0b.	RW1CS

Bit Location	Register Description	Attributes
2:1	**DPC Trigger Reason** – This field indicates why DPC has been triggered. Defined encodings are: 00bDPC was triggered due to an unmasked uncorrectable error. 01bDPC was triggered due to receiving an ERR_NONFATAL. 10bDPC was triggered due to receiving an ERR_FATAL. 11bDPC was triggered due to a reason that is indicated by the DPC Trigger Reason Extension field. This field is valid only when the DPC Trigger Status bit is Set; otherwise the value of this field is undefined.	ROS
3	**DPC Interrupt Status** – This bit is Set if DPC is triggered while the DPC Interrupt Enable bit is Set. This may cause the generation of an interrupt. See Section 6.2.10.1. Default value of this bit is 0b.	RW1CS
4	**DPC RP Busy** – When the DPC Trigger Status bit is Set and this bit is Set, the Root Port is busy with internal activity that must complete before software is permitted to Clear the DPC Trigger Status bit. If software Clears the DPC Trigger Status bit while this bit is Set, the behavior is undefined. This field is valid only when the DPC Trigger Status bit is Set; otherwise the value of this field is undefined. This bit is applicable only for Root Ports that support RP Extensions for DPC, and is Reserved for Switch Downstream Ports. Default value of this bit is undefined.	RO / RsvdZ
6:5	**DPC Trigger Reason Extension** – This field serves as an extension to the DPC Trigger Reason field. When that field is valid and has a value of 11b, this field indicates why DPC has been triggered. Defined encodings are: 00bDPC was triggered due to an RP PIO error. 01bDPC was triggered due to the DPC Software Trigger bit. 10b Reserved 11b Reserved This field is valid only when the DPC Trigger Status bit is Set and the value of the DPC Trigger Reason field is 11b; otherwise the value of this field is undefined.	ROS
12:8	**RP PIO First Error Pointer** – The value of this field identifies a bit position in the RP PIO Status register, and this field is considered valid when that bit is Set. When this field is valid, and software writes a 1b to the indicated RP PIO Status bit (thus clearing it), this field must revert to its default value. This field is applicable only for Root Ports that support RP Extensions for DPC, and otherwise is Reserved. If this field is not Reserved, its default value is 11111b, indicating a permanently Reserved RP PIO Status bit, thus guaranteeing that this field is not considered valid.	ROS / RsvdZ

7.9.15.5 *DPC Error Source ID Register (Offset 0Ah)*

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DPC Error Source ID															

Figure 7-235: DPC Error Source ID Register

Table 7-190: DPC Error Source ID Register

Bit Location	Register Description	Attributes
15:0	**DPC Error Source ID** – When the DPC Trigger Reason field indicates that DPC was triggered due to the reception of an ERR_NONFATAL or ERR_FATAL, this register contains the Requester ID of the received Message. Otherwise, the value of this register is undefined.	ROS

7.9.15.6 *RP PIO Status Register (Offset 0Ch)*

This register is present only in Root Ports that support RP Extensions for DPC. See Section 6.2.10.3.

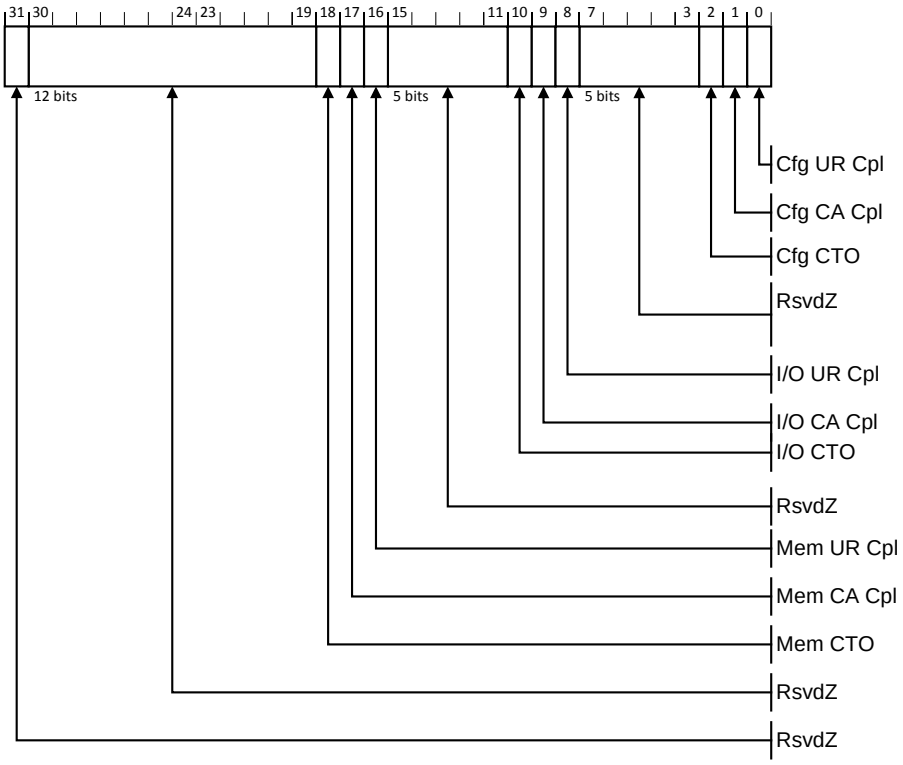


Figure 7-236: RP PIO Status Register

Table 7-191: RP PIO Status Register

Bit Location	Register Description	Attributes	Default
0	Cfg UR Cpl – Configuration Request received UR Completion	RW1CS	0b
1	Cfg CA Cpl – Configuration Request received CA Completion	RW1CS	0b
2	Cfg CTO – Configuration Request Completion Timeout	RW1CS	0b
8	I/O UR Cpl – I/O Request received UR Completion	RW1CS	0b
9	I/O CA Cpl – I/O Request received CA Completion	RW1CS	0b
10	I/O CTO – I/O Request Completion Timeout	RW1CS	0b
16	Mem UR Cpl – Memory Request received UR Completion	RW1CS	0b
17	Mem CA Cpl – Memory Request received CA Completion	RW1CS	0b
18	Mem CTO – Memory Request Completion Timeout	RW1CS	0b
31	Permanently Reserved, since the default RP PIO First Error Pointer field value points to it.	RsvdZ	0b

7.9.15.7 *RP PIO Mask Register (Offset 10h)*

This register is present only in Root Ports that support RP Extensions for DPC. See Section 6.2.10.3.

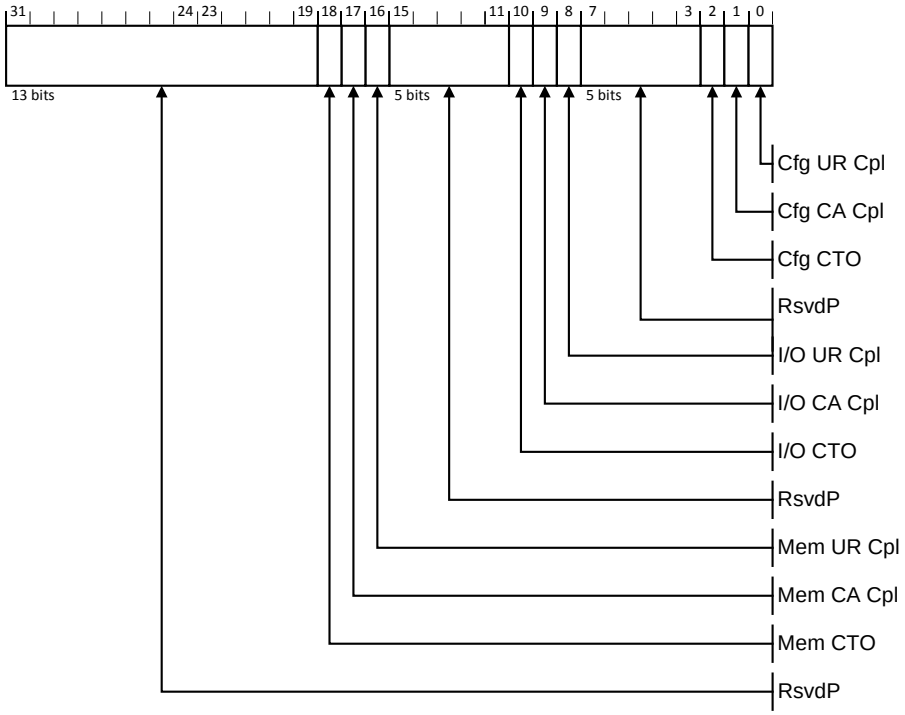


Figure 7-237: RP PIO Mask Register

Table 7-192: RP PIO Mask Register

Bit Location	Register Description	Attributes	Default
0	Cfg UR Cpl – Configuration Request received UR Completion	RWS	1b
1	Cfg CA Cpl – Configuration Request received CA Completion	RWS	1b
2	Cfg CTO – Configuration Request Completion Timeout	RWS	1b
8	I/O UR Cpl – I/O Request received UR Completion	RWS	1b
9	I/O CA Cpl – I/O Request received CA Completion	RWS	1b
10	I/O CTO – I/O Request Completion Timeout	RWS	1b
16	Mem UR Cpl – Memory Request received UR Completion	RWS	1b
17	Mem CA Cpl – Memory Request received CA Completion	RWS	1b
18	Mem CTO – Memory Request Completion Timeout	RWS	1b

7.9.15.8 *RP PIO Severity Register (Offset 14h)*

This register is present only in Root Ports that support RP Extensions for DPC. See Section 6.2.10.3.

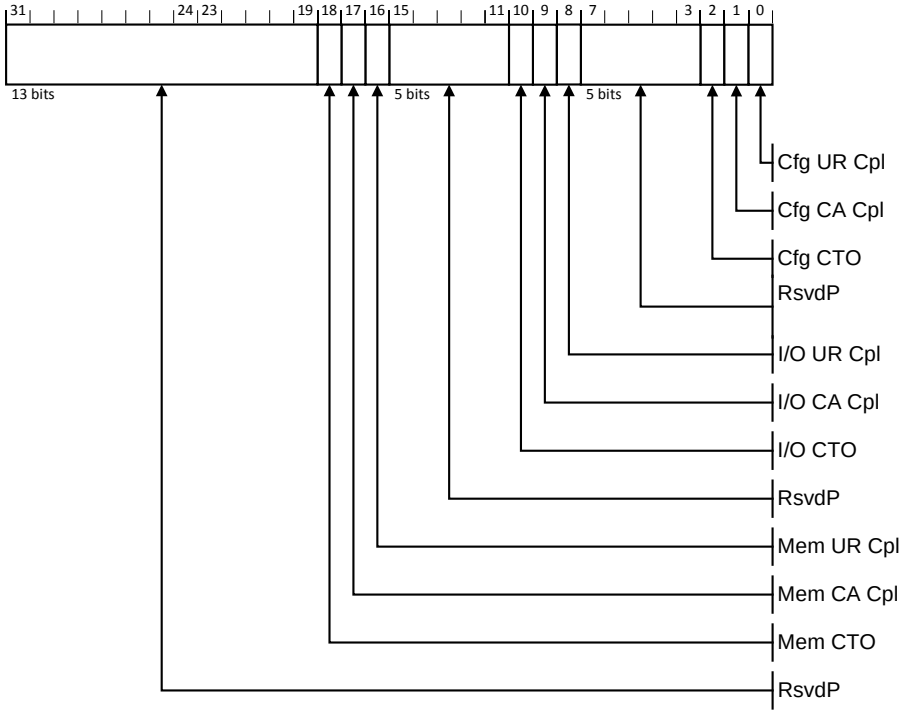


Figure 7-238: RP PIO Severity Register

Table 7-193: RP PIO Severity Register

Bit Location	Register Description	Attributes	Default
0	Cfg UR Cpl – Configuration Request received UR Completion	RWS	0b
1	Cfg CA Cpl – Configuration Request received CA Completion	RWS	0b
2	Cfg CTO – Configuration Request Completion Timeout	RWS	0b
8	I/O UR Cpl – I/O Request received UR Completion	RWS	0b
9	I/O CA Cpl – I/O Request received CA Completion	RWS	0b
10	I/O CTO – I/O Request Completion Timeout	RWS	0b
16	Mem UR Cpl – Memory Request received UR Completion	RWS	0b
17	Mem CA Cpl – Memory Request received CA Completion	RWS	0b
18	Mem CTO – Memory Request Completion Timeout	RWS	0b

7.9.15.9 *RP PIO SysError Register (Offset 18h)*

This register is present only in Root Ports that support RP Extensions for DPC. See Section 6.2.10.3.

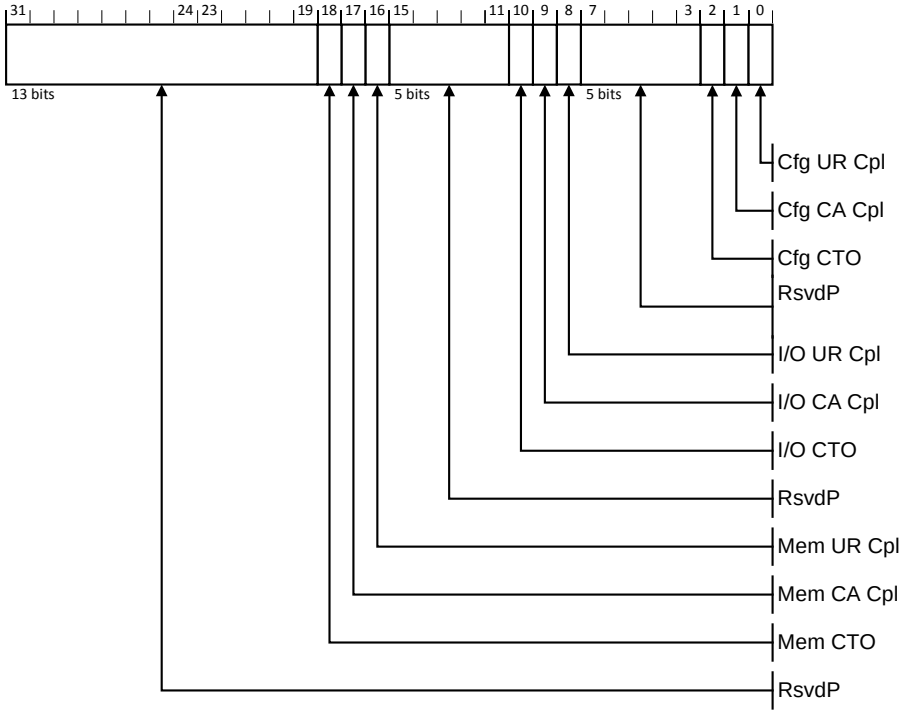


Figure 7-239: RP PIO SysError Register

Table 7-194: RP PIO SysError Register

Bit Location	Register Description	Attributes	Default
0	Cfg UR Cpl – Configuration Request received UR Completion	RWS	0b
1	Cfg CA Cpl – Configuration Request received CA Completion	RWS	0b

Bit Location	Register Description	Attributes	Default
2	Cfg CTO – Configuration Request Completion Timeout	RWS	0b
8	I/O UR Cpl – I/O Request received UR Completion	RWS	0b
9	I/O CA Cpl – I/O Request received CA Completion	RWS	0b
10	I/O CTO – I/O Request Completion Timeout	RWS	0b
16	Mem UR Cpl – Memory Request received UR Completion	RWS	0b
17	Mem CA Cpl – Memory Request received CA Completion	RWS	0b
18	Mem CTO – Memory Request Completion Timeout	RWS	0b

7.9.15.10 *RP PIO Exception Register (Offset 1Ch)*

This register is present only in Root Ports that support RP Extensions for DPC. See Section 6.2.10.3.

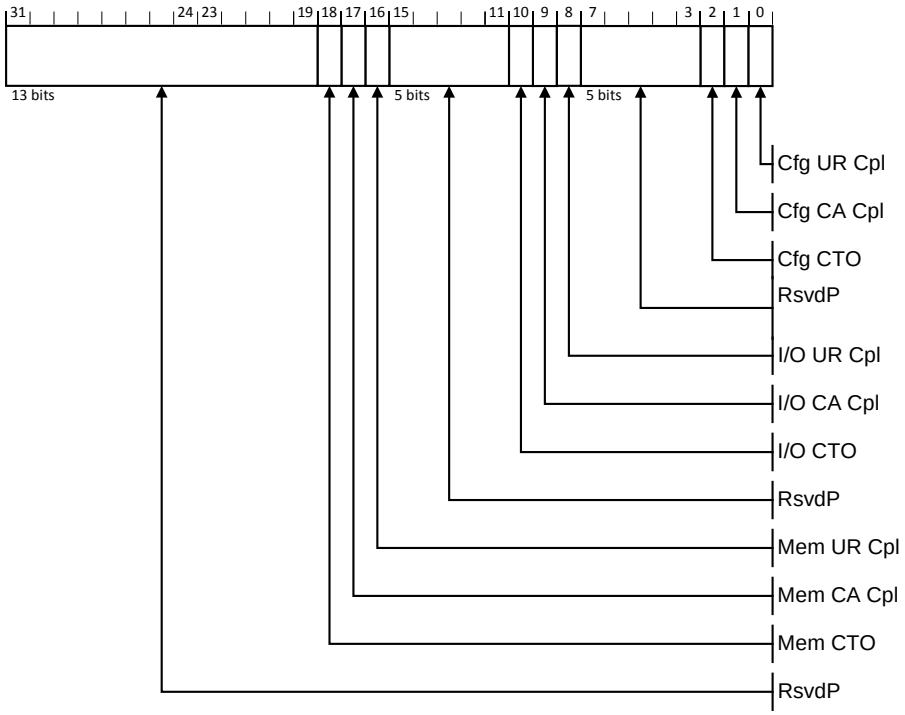


Figure 7-240: RP PIO Exception Register

Table 7-195: RP PIO Exception Register

Bit Location	Register Description	Attributes	Default
0	Cfg UR Cpl – Configuration Request received UR Completion	RWS	0b
1	Cfg CA Cpl – Configuration Request received CA Completion	RWS	0b
2	Cfg CTO – Configuration Request Completion Timeout	RWS	0b
8	I/O UR Cpl – I/O Request received UR Completion	RWS	0b
9	I/O CA Cpl – I/O Request received CA Completion	RWS	0b
10	I/O CTO – I/O Request Completion Timeout	RWS	0b
16	Mem UR Cpl – Memory Request received UR Completion	RWS	0b
17	Mem CA Cpl – Memory Request received CA Completion	RWS	0b
18	Mem CTO – Memory Request Completion Timeout	RWS	0b

7.9.15.11 *RP PIO Header Log Register (Offset 20h)*

This register is implemented only in Root Ports that support RP Extensions for DPC. The RP PIO Header Log register contains the header from the Request TLP associated with a recorded RP PIO error. Refer to Section 6.2.10.3 for further details. This register is 16 bytes and is formatted identically to the Header Log register in AER. See Section 7.8.4.8.

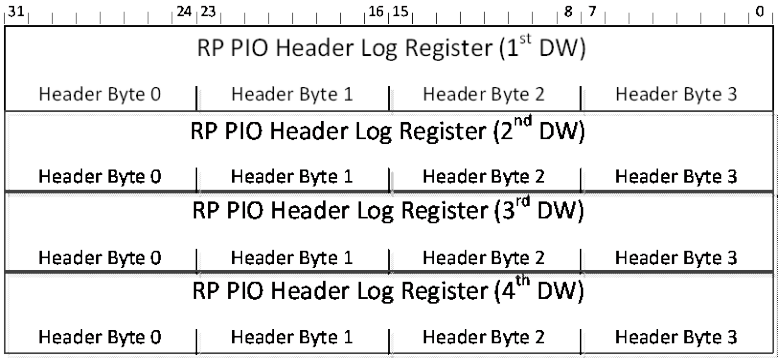


Figure 7-241: RP PIO Header Log Register

Table 7-196: RP PIO Header Log Register

Bit Location	Register Description	Attributes	Default
127:0	Header of TLP associated with error	ROS	0

7.9.15.12 *RP PIO ImpSpec Log Register (Offset 30h)*

This register is permitted to be implemented only in Root Ports that support RP Extensions for DPC. The RP PIO ImpSpec Log register, if implemented, contains implementation-specific

information associated with the recorded error, e.g., indicating the source of the Request TLP. Space is allocated for this register if the value of the RP PIO Log Size field is 5 or greater. If space is allocated for the register, but the register is not implemented, the bits must be hardwired to 0b.

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
RP PIO ImpSpec Log

Figure 7-242: RP PIO ImpSpec Log Register

Table 7-197: RP PIO ImpSpec Log Register

Bit Location	Register Description	Attributes	Default
31:0	RP PIO ImpSpec Log	ROS	0

7.9.15.13 *RP PIO TLP Prefix Log Register (Offset 34h)*

This register is permitted to be implemented only in Root Ports that support RP Extensions for DPC. The RP PIO TLP Prefix Log register contains any End-End TLP Prefixes from the TLP corresponding to a recorded RP PIO error. Refer to Section 6.2.10.3 for further details.

If the Root Port supports tracking Non-Posted Requests that contain End-End TLP Prefixes, this register must be implemented, and must be of sufficient size to record the maximum number of End-End TLP Prefixes for any tracked Request. See Section 2.9.3. The allocated size in DWORDs of the RP PIO TLP Prefix Log register is the RP PIO Log Size minus 5 if the RP PIO Log Size is 9 or less, or 4 if the RP PIO Log Size is greater than 9. The implemented size of the TLP Prefix Log must be less than or equal to the Root Port's Max End-End TLP Prefixes field value. For the case where the Root Port never transmits Non-Posted Requests containing End-End TLP Prefixes, the allocated and implemented size of the TLP Prefix Log is permitted to be 0. Any DWORDs allocated but not implemented must be hardwired to zero.

This register is formatted identically to the TLP Prefix Log register in AER, although this register's allocated size is variable, whereas the register in AER is always 4 DWORDs. See Section 7.8.4.12. The First TLP Prefix Log register contains the first End-End TLP Prefix from the TLP, the Second TLP Prefix Log register contains the second End-End TLP Prefix, and so forth. If the TLP contains fewer TLP Prefixes than this register accommodates, any remaining TLP Prefix Log registers must contain zero.

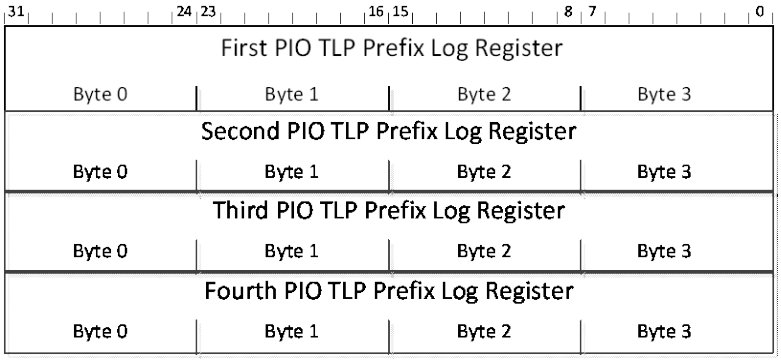


Figure 7-243: RP PIO TLP Prefix Log Register

Table 7-198: RP PIO TLP Prefix Log Register

Bit Location	Register Description	Attributes	Default
127:0	RP PIO TLP Prefix Log	ROS	0

7.9.16 Precision Time Management (PTM) Capability

The Precision Time Measurement (PTM) Capability is an optional Extended Capability for discovering and controlling the distribution of a PTM Hierarchy. For Root Complexes, this Capability is required in any Root Port, RCiEP, or RCRB that supports PTM. For Endpoints and Switch Upstream Ports that support PTM, this Capability is required in exactly one Function of the Upstream Port and that Capability controls the PTM behavior of all PTM capable Functions associated with that Upstream Port. For Switch Downstream Ports, PTM behavior is controlled by the same PTM Capability that controls the associated Switch Upstream Port. The PTM Capability is not permitted in Bridges, Switch Downstream Ports, and Root Complex Event Collectors.

For Switches, a single instance of this Capability controls behavior for the entire Switch. If the Upstream Port of the Switch is associated with an MFD, it is not required that the controlling Function be the Function corresponding to the Switch Upstream Port. For a given Switch, if this Capability is present, all Downstream Ports of the Switch must implement the requirements defined in Section 6.22.3.2.

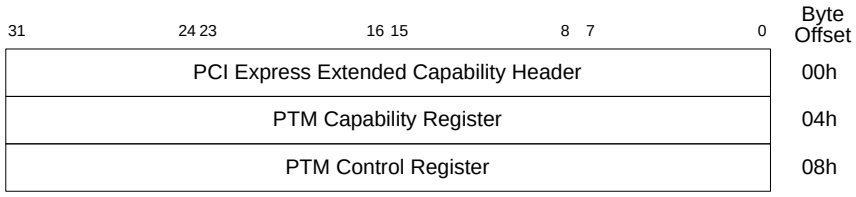


Figure 7-244: PTM Extended Capability Structure

7.9.16.1 *PTM Extended Capability Header (Offset 00h)*

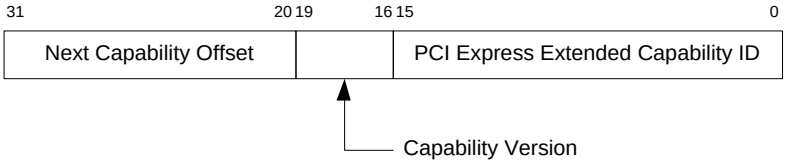


Figure 7-245: PTM Extended Capability Header

Table 7-199: PTM Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. PCI Express Extended Capability ID for the Precision Time Measurement Capability is 001Fh.	HwInit
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.	HwInit
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express Extended Capability structure or 000h if no other items exist in the linked list of Capabilities.	HwInit

7.9.16.2 *PTM Capability Register (Offset 04h)*

This register describes a Function's support for Precision Time Measurement. Not all fields within this register apply to all Functions capable of implementing PTM.

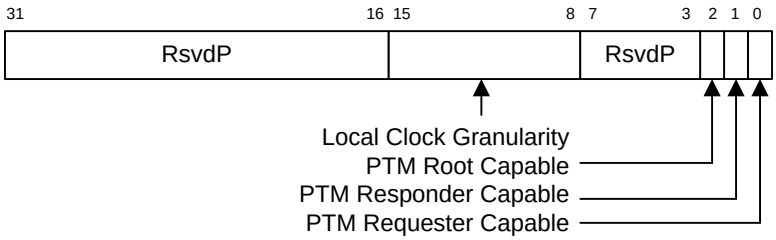


Figure 7-246: PTM Capability Register

Table 7-200: PTM Capability Register

Bit Location	Register Description	Attributes
0	**PTM Requester Capable** – Endpoints and RCiEPs are permitted to, and Switches supporting PTM must, set this bit to 1b to indicate they implement the PTM Requester role (see Section 6.22.3.1).	HwInit
1	**PTM Responder Capable** – Root Ports and RCRB's are permitted to, and Switches supporting PTM must, set this bit to 1b to indicate they implement the PTM Responder role (see Section 6.22.3.2). If PTM Root Capable is Set, this bit must be Set to 1b.	HwInit
2	**PTM Root Capable** – Root Ports, RCRB's and Switches are permitted to set this bit to 1b, if they implement a PTM Time Source Role and are capable of serving as the PTM Root.	HwInit
7:3	Reserved	RsvdP

15:8	Local Clock Granularity – 0000 0000b – Time Source does not implement a local clock. It simply propagates timing information obtained from further Upstream in the PTM Hierarchy when responding to PTM Request messages. 0000 0001b – 1111 1110b: Indicates the period of this Time Source's local clock in ns. 1111 1111b: Indicates the period of this Time Source's local clock is greater than 254 ns. If the PTM Root Select bit is Set, this local clock is used to provide PTM Master Time. Otherwise, the Time Source uses this local clock to locally track PTM Master Time received from further Upstream within a PTM Hierarchy. This field is reserved for Functions that do not implement the PTM Time Source role.	HwInit/RsvdP
31:16	Reserved	RsvdP

7.9.16.3 *PTM Control Register (Offset 08h)*

This register controls a Function's participation in the Precision Time Measurement mechanism. Not all fields within this register apply to all Functions capable of implementing PTM.

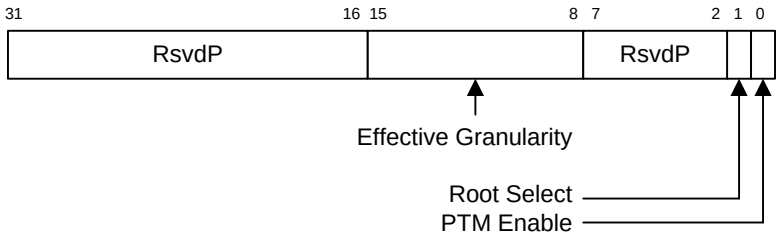


Figure 7-247: PTM Control Register

Table 7-201: PTM Control Register

Bit Location	Register Description	Attributes
0	**PTM Enable** – When Set, this Function is permitted to participate in the PTM mechanism according to its selected role(s) (see Section 6.22.2). Default value is 0b.	RW
1	**Root Select** – When Set, if the PTM Enable bit is also Set, this Time Source is the PTM Root. Within each PTM Hierarchy, it is recommended that system software select only the furthest Upstream Time Source to be the PTM Root. Default value is 0b.If the value of the PTM Root Capable bit is 0b, this bit is permitted to be hardwired to 0b.	RW/RO
7:2	Reserved	RsvdP

Bit Location	Register Description	Attributes
15:8	**Effective Granularity** – For Functions implementing the PTM Requester Role, this field provides information relating to the expected accuracy of the PTM clock, but does not otherwise affect the PTM mechanism. For Endpoints, system software must program this field to the value representing the maximum Local Clock Granularity reported by the PTM Root and all intervening PTM Time Sources. For RCiEPs, system software must set this field to the value reported in the Local Clock Granularity field by the associated PTM Time Source. Permitted values: 0000 0000b: Unknown PTM granularity – one or more Switches between this Function and the PTM Root reported a Local Clock Granularity value of 0000 0000b. 0000 0001b – 1111 1110b: Indicates the effective PTM granularity in ns. 1111 1111b: Indicates the effective PTM granularity is greater than 254 ns. Default value is 00000b. If PTM Requester Capable is Clear, this field is permitted to be hardwired to 0000 0000b.	RW/RO
31:16	Reserved	RsvdP

7.9.17 Readiness Time Reporting Extended Capability

The Readiness Time Reporting Extended Capability provides an optional mechanism for describing the time required for a Device or Function to become Configuration-ready. In the indicated situations, software is permitted to issue Requests to the Device or Function after waiting for the time advertised in this capability and need not wait for the (longer) times required elsewhere.

Software is permitted to issue requests upon the earliest of:

- Receiving a Readiness Notifications message (see Section 6.23).
- Waiting the appropriate time as specified in this document or in applicable specifications including the *PCI Local Bus Specification* and the *PCI Bus Power Management Interface Specification*.
- Waiting the time indicated in the associated field of this capability.
- Waiting the time defined by system software or firmware[141].

Software is permitted to cache values from this capability and to use those cached values when the device topology has not changed.

[141] For example, using ACPI tables to provide the equivalent of this capability.

This capability is permitted to be implemented in all Functions.

A Function must be Configuration-Ready if:

- The Immediate Readiness bit is Clear and at least **Reset Time** has elapsed after the completion of Conventional Reset
  - If the Immediate Readiness bit is Set, **Reset Time** does not apply, and is Reserved
- The Function is associated with an Upstream Port and at least **DL_Up Time** has elapsed after the Downstream Port above that Function reported Data Link Layer Link Active (see Section 7.5.3.8).
- The Function supports Function Level Reset and at least **FLR Time** has elapsed after that Function was issued a Function Level Reset.
- Immediate_Readiness_on_Return_to_D0 is Clear and at least **$D3_{hot}$ to D0 Time** has elapsed after that Function was directed to the D0 state from $D3_{hot}$.
  - If the Immediate_Readiness_on_Return_to_D0 bit is Set, **$D3_{hot}$ to D0 Time** does not apply, and is Reserved

When Immediate_Readiness_on_Return_to_D0 is Clear, a Function must be Configuration-Ready when at least **$D3_{hot}$ to D0 Time** has elapsed after the Function was directed to the D0 state from $D3_{hot}$. In addition, the Function must be in either the $D0_{uninitialized}$ or $D0_{active}$ state, depending on the value of the No_Soft_Reset bit.

For VFs additional behavior is defined in Chapter 9.

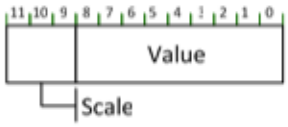
If the above conditions do not apply, Function behavior is not determined by the Readiness Time Reporting Extended Capability, and the Function must respond as defined elsewhere (including, for example, no response or a response with Configuration Retry Status).

The time values reported are determined by implementation-specific mechanisms. A valid bit is defined in this capability to permit a device to defer reporting time values, for example to allow hardware initialization through driver-based mechanisms. If the Valid bit remains Clear and 1 minute has elapsed after device driver(s) have started, software is permitted to assume that no values will be reported.

Registers and fields in the Readiness Time Reporting Extended Capability are shown in Figure 7-248. Time values are encoded in floating point as shown in Figure 7-249. The actual time value is $Value \times Multiplier[Scale]$. For example, the value A1Eh represents about 1 second (actually 1.006 sec) and the value 80Ah represents about 10 ms (actually 10.240 ms).

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	Byte Offset
PCI Express Extended Capability Header	00h
Readiness Time Reporting 1	04h
Readiness Time Reporting 2	08h

Figure 7-248: Readiness Time Reporting Extended Capability



Scale	Multiplier
0	1 ns
1	32 ns
2	1,024 ns
3	32,768 ns
4	1,048,576 ns
5	33,554,432 ns
6	Reserved
7	Reserved

Multiplier = 32^{scale} ns

Figure 7-249: Readiness Time Encoding

7.9.17.1 *Readiness Time Reporting Extended Capability Header (Offset 00h)*

Figure 7-250 and Table 7-202 detail allocation of fields in the Extended Capability header.

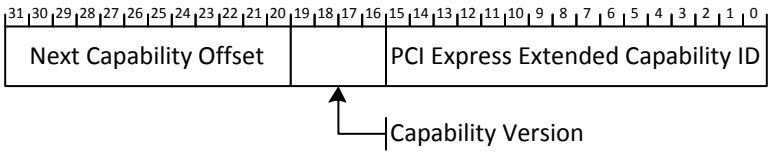


Figure 7-250: Readiness Time Reporting Extended Capability Header

Table 7-202: Readiness Time Reporting Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. Extended Capability ID for the Readiness Time Reporting Extended Capability is 0022h.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in	RO

	the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh.	

7.9.17.2 *Readiness Time Reporting 1 (Offset 04h)*

Figure 7-251 and Table 7-203 detail allocation of fields in the Readiness Time Reporting 1 Register.

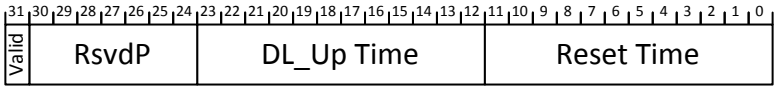


Figure 7-251: Readiness Time Reporting 1 Register

Table 7-203: Readiness Time Reporting 1 Register

Bit Location	Register Description	Attributes
11:0	**Reset Time** – is the time the Function requires to become Configuration-Ready after the completion of Conventional Reset. This field is RsvdP if the Immediate Readiness bit is Set. This field is undefined when the Valid bit is Clear. This field must be less than or equal to the encoded value A1Eh.	HwInit/RsvdP
23:12	**DL_Up Time** – is the time the Function requires to become Configuration-Ready after the Downstream Port above the Function reports Data Link Layer Link Active. This field is RsvdP in Functions that are not associated with an Upstream Port. This field is undefined when the Valid bit is Clear. This field must be less than or equal to the encoded value A1Eh.	HwInit/RsvdP
30:24	Reserved	RsvdP
31	**Valid** – If Set, indicates that all time values in this capability are valid. If Clear, indicates that the time values in this capability are not yet available. Time values may depend on device configuration. Device specific mechanisms, possibly involving the device driver(s), could be involved in determining time values. If this bit remains Clear and 1 minute has elapsed after all associated device driver(s) have started, software is permitted to assume that this bit will never be set.	HwInit

7.9.17.3 *Readiness Time Reporting 2 (Offset 08h)*

Figure 7-252 and Table 7-204 detail allocation of fields in the Readiness Time Reporting 2 Register.

31:30	29:28	27:26	25:24	23:22	21:20	19:18	17:16	15:14	13:12	11:10	9:8	7:6	5:4	3:2	1:0
RsvdP				D3hot to D0 Time						FLR Time					

Figure 7-252: Readiness Time Reporting 2 Register

Table 7-204: Readiness Time Reporting 2 Register

Bit Location	Register Description	Attributes
11:0	**FLR Time** – is the time that the Function requires to become Configuration-Ready after it was issued an FLR. This field is RsvdP when the Function Level Reset Capability bit is Clear (see Section 7.5.3.3). This field is undefined when the Valid bit is Clear. This field must be less than or equal to the encoded value A1Eh.	HwInit/RsvdP
23:12	**$D3_{hot}$ to D0 Time** –If Immediate_Readiness_on_Return_to_D0 is Clear, $D3_{hot}$ to D0 Time is the time that the Function requires after it is directed from $D3_{hot}$ to D0 before it is Configuration-Ready and has returned to either $D0_{unitialized}$ or $D0_{active}$ state (see the *PCI Bus Power Management Interface Specification*). This field is RsvdP if the Immediate_Readiness_on_Return_to_D0 bit is Set. This field is undefined when the Valid bit is Clear. This field must be less than or equal to the encoded value 80Ah.	HwInit/RsvdP
31:24	Reserved	RsvdP

7.9.18 Hierarchy ID Extended Capability

The Hierarchy ID Extended Capability provides an optional mechanism for passing a unique identifier to Functions within a Hierarchy. At most one instance of this capability is permitted in a Function. This capability is not applicable to Bridges, Root Complex Event Collectors, and RCRBs.

This capability takes three forms:

In Upstream Ports:

- This capability is permitted any Function associated with an Upstream Port.
- This capability is optional in Switch Upstream Ports. Support in Switch Upstream and Downstream Ports is independently optional.
- This capability is mandatory in Functions that use the Hierarchy ID Message. This includes use by the Function's driver.

- Functions, other than VFs, that have Hierarchy ID Writeable Clear, must report the Message Routing ID, Hierarchy ID, System GUID Authority ID, and System GUID fields from the most recently received Hierarchy ID Message.
- All VFs that have Hierarchy ID Writeable Clear, must report the same Hierarchy ID Valid, Message Routing ID, Hierarchy ID, System GUID Authority ID, and System GUID values as their associated PF.
- PFs must implement this capability if any of their VFs implement this capability.
- Functions that have Hierarchy ID Writeable Set must report the Hierarchy ID Valid, Message Routing ID, Hierarchy ID, System GUID Authority ID, and System GUID values programmed by software.

In Downstream Ports:

- This capability is permitted in any Downstream Port. It is recommended that it be implemented in Root Ports.
- When present in a Switch Downstream Port, this capability must be implemented in all Downstream Ports of the Switch. Support in Switch Upstream and Downstream Ports is independently optional.
- In Downstream Ports, the Hierarchy ID, System GUID Authority ID, and System GUID fields are Read / Write and contain the values to send in the Hierarchy ID Message.
- A Hierarchy ID capability is not affected by Hierarchy ID Messages forwarded through the associated Downstream Port.

In RCiEPs:

- VFs that have Hierarchy ID Writeable Clear must report the same Message Routing ID, Hierarchy ID, System GUID Authority ID, and System GUID values as their associated PF.
- PFs must implement this capability if any of their VFs implement this capability.
- Functions, other than VFs, that have Hierarchy ID Writeable Clear, must report the same Hierarchy ID Valid, Message Routing ID, Hierarchy ID, System GUID Authority ID, and System GUID values. The source of this information is outside the scope of this specification.
- Functions that have Hierarchy ID Writeable Set must report the Hierarchy ID Valid, Message Routing ID, Hierarchy ID, System GUID Authority ID, and System GUID values programmed by software.

Figure 7-253 details the layout of the Hierarchy ID Extended Capability.

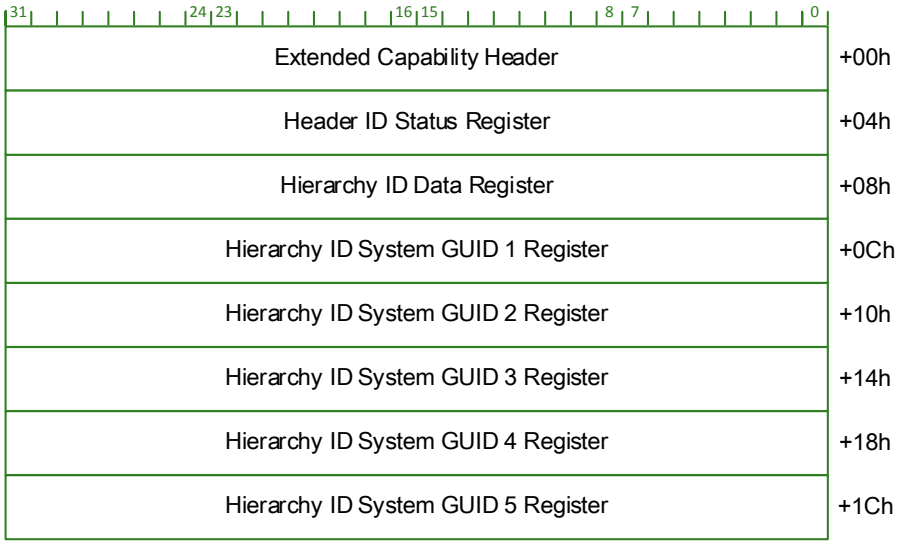


Figure 7-253: Hierarchy ID Extended Capability

7.9.18.1 *Hierarchy ID Extended Capability Header*

Figure Figure 7-254 and Table 7-205 detail allocation of fields in the Hierarchy ID Extended Capability Header.

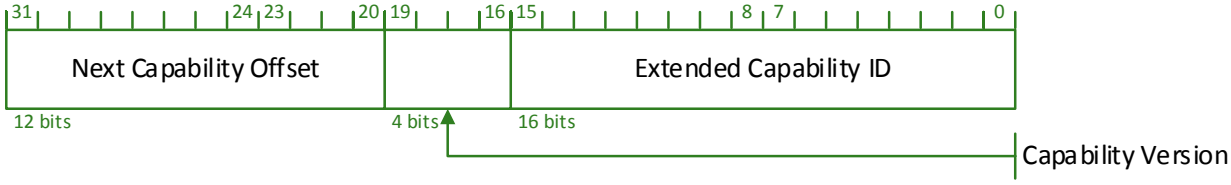


Figure 7-254: Hierarchy ID Extended Capability Header

Table 7-205: Hierarchy ID Extended Capability Header

Bit Location	Description	Attributes
15:0	**Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. PCI Express Extended Capability ID for the Hierarchy ID Capability is 0028h.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express Extended Capability structure or 000h if	RO

	no other items exist in the linked list of Capabilities. For Extended Capabilities in configuration space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating the list of Capabilities) or greater than 0FFh.	

7.9.18.2 *Hierachy ID Status Register*

Figure 7-255 and Table 7-206 detail allocation of fields in the Hierarchy ID Status Register.

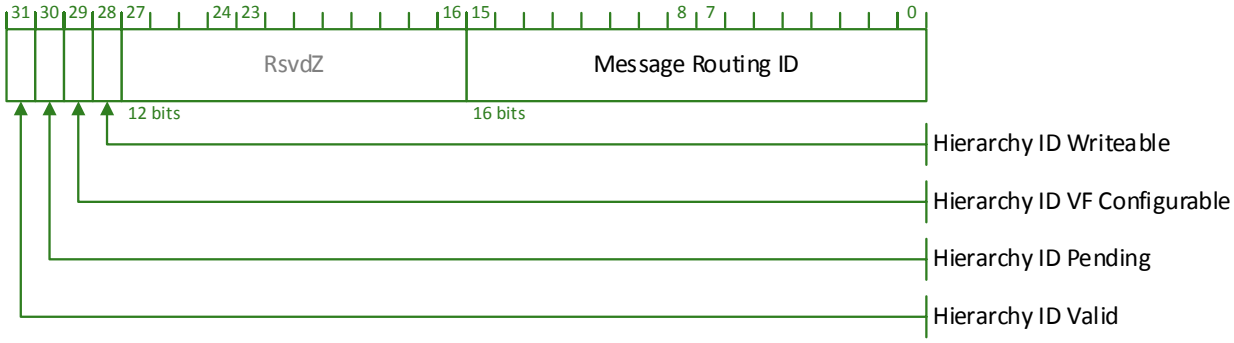


Figure 7-255: Hierarchy ID Status Register

Table 7-206: Hierarchy ID Status Register

Bit Location	Description	Attributes
15:0	**Message Routing ID** – In an Upstream Port, this field contains the Routing ID from the most recently received Hierarchy ID Message. This field is meaningful only if Hierarchy ID Valid is 1b. In a Downstream Port, this field is RsvdZ. For RCiEPs, this field is RsvdZ. This field defaults to 0000h.	RO/RsvdZ
29:16	RsvdZ	RsvdZ

Bit Location	Description	Attributes
28	**Hierarchy ID Writeable** – This bit is Set to indicate that the Hierarchy ID Data and GUID registers are read/write. This bit is Clear to indicate that the Hierarchy ID and GUID registers are read only. In Downstream Ports this bit is hardwired to 1b. In Upstream Ports, Functions that are not VFs must hardwire this bit to 0b. RCiEPs that are not VFs, must hardwire this bit to either 0b or 1b. VFs in an Upstream Port and Root Complex Integrated VFs are permitted to either: ❑ hardwire this bit to 0b or ❑ implement this bit as read / write with a default value of 0b.	RW/RO
29	**Hierarchy ID VF Configurable** – This bit indicates that Hierarchy ID Writeable can be configured. If Hierarchy ID Writeable is implemented as read / write, this bit is 1b. Otherwise this bit is 0b.	RO
30	**Hierarchy ID Pending** – In Downstream Ports this requests the transmittion of a Hierarchy ID Message. Setting it requests transmission of a message based on the Hierarchy Data and GUID registers in this capability. This bit is cleared when either the transmit request is satisfied or the Link enters DL_Down. Behavior is undefined if the Hierarchy Data or GUID registers in this capability are written while this bit is Set. In Downstream Ports, this bit is Read / Write defaulting to 0b. In all other Functions, this bit is RsvdZ.	RW/RsvdZ
31	**Hierarchy ID Valid** – This bit indicates that the remaining fields in this capability are meaningful. In Downstream Ports, this bit is hardwired to 1b. In all other Functions, the following rules apply: ❑ If Hierarchy ID Writeable is Set, this bit is read/write, default 0b. ❑ If Hierarchy ID Writeable is Clear, this bit is read only, default 0b. o In VFs, this bit contains the same value as the associated PF. o In Functions other than VFs that are associated with an Upstream Port, this bit is Set when a Hierarchy ID Message is received, and Cleared when the Link is DL_Down. o In RCiEPs other than VFs, this bit contains a system provided value. The mechanism for determining this value is outside the scope of this specification.	RW/RO

7.9.18.3 *Hierarchy ID Data Register*

Figure 7-256 and Table 7-207 detail allocation of fields in the Hierarchy ID Data Register.

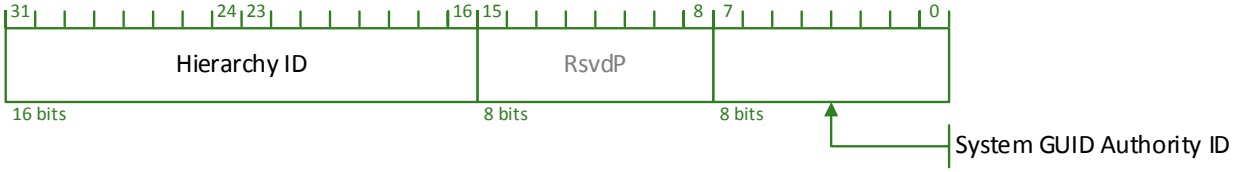


Figure 7-256: Hierarchy ID Data Register

Table 7-207: Hierarchy ID Data Register

Bit Location	Description	Attributes
7:0	**System GUID Authority ID** – This field corresponds to the System GUID Authority ID field in the Hierarchy ID Message. See Section 6.26 for details. This field is meaningful only if Hierarchy ID Valid is 1b. If Hierarchy ID Writeable is Set, this field is read-write and contains the value programmed by software. If Hierarchy ID Writeable is Clear, this field is read only. The value is determined using the rules defined in Section 7.9.18. This field defaults to 00h.	RO/RW
15:8	RsvdP	RsvdP
31:16	**Hierarchy ID** – This field corresponds to the Hierarchy ID field in the Hierarchy ID Message. See Section 6.26 for details. This field is meaningful only if Hierarchy ID Valid is 1b. If Hierarchy ID Writeable is Set, this field is read-write and contains the value programmed by software. If Hierarchy ID Writeable is Clear, this field is read only. The value is determined using the rules defined in Section 7.9.18. This field defaults to 0000h.	RO/RW

7.9.18.4 *Hierarchy ID GUID 1 Register*

Figure 7-257 and Table 7-208 detail allocation of fields in the Hierarchy ID GUID 1 Register.

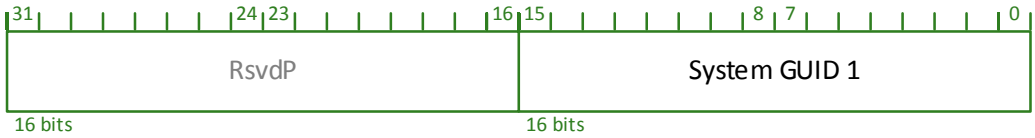


Figure 7-257: Hierarchy ID GUID 1 Register

Table 7-208: Hierarchy ID GUID 1 Register

Bit Location	Description	Attributes
15:0	**System GUID 1** – This field corresponds to bits [143:128] of the System GUID in the Hierarchy ID Message. See Section 6.26 for details. This field is meaningful only if Hierarchy ID Valid is 1b. If Hierarchy ID Writeable is Set, this field is read-write and contains the value programmed by software. If Hierarchy ID Writeable is Clear, this field is read only. The value is determined using the rules defined in Section 7.9.18. This field defaults to 0000h.	RO/RW
31:0	RsvdP	RsvdP

7.9.18.5 *Hierarchy ID GUID 2 Register*

Figure 7-258 and Table 7-209 detail allocation of fields in the Hierarchy ID GUID 2 Register.

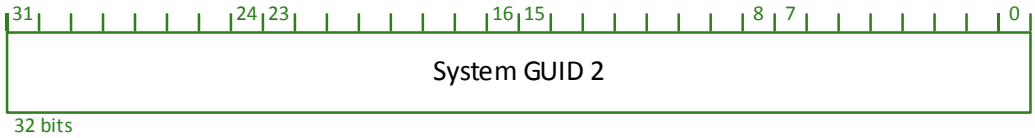


Figure 7-258: Hierarchy ID GUID 2 Register

Table 7-209: Hierarchy ID GUID 2 Register

Bit Location	Description	Attributes
31:0	**System GUID 2** – This field corresponds to bits [127:96] of the System GUID field in the Hierarchy ID Message. See Section 6.26 for details. This field is meaningful only if Hierarchy ID Valid is 1b. If Hierarchy ID Writeable is Set, this field is read-write and contains the value programmed by software. If Hierarchy ID Writeable is Clear, this field is read only. The value is determined using the rules defined in Section 7.9.18. This field defaults to 0000 0000h.	RO/RW

7.9.18.6 *Hierarchy ID GUID 3 Register*

Figure 7-259 and Table 7-210 detail allocation of fields in the Hierarchy ID GUID 3 Register.

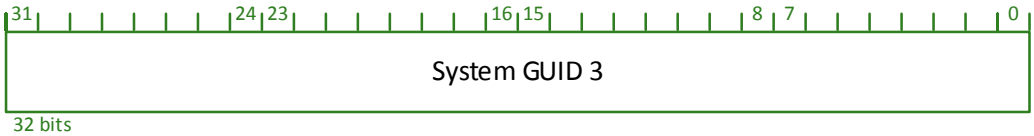


Figure 7-259: Hierarchy ID GUID 3 Register

Table 7-210: Hierarchy ID GUID 3 Register

Bit Location	Description	Attributes
31:0	**System GUID 3** – This field corresponds to bits [95:64] of the System GUID field in the Hierarchy ID Message. See Section 6.26 for details. This field is meaningful only if Hierarchy ID Valid is 1b. If Hierarchy ID Writeable is Set, this field is read-write and contains the value programmed by software. If Hierarchy ID Writeable is Clear, this field is read only. The value is determined using the rules defined in Section 7.9.18. This field defaults to 0000 0000h.	RO/RW

7.9.18.7 *Hierarchy ID GUID 4 Register*

Figure 7-260 and Table 7-211 detail allocation of fields in the Hierarchy ID GUID 4 Register.

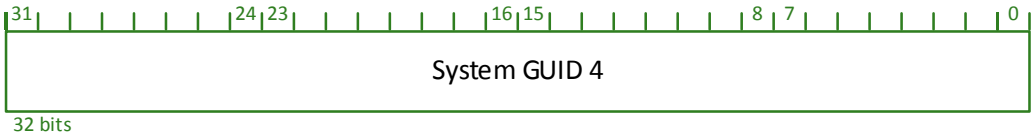


Figure 7-260: Hierarchy ID GUID 4 Register

Table 7-211: Hierarchy ID GUID 4 Register

Bit Location	Description	Attributes
31:0	**System GUID 4** – This field corresponds to bits [63:32] of the System GUID field in the Hierarchy ID Message. See Section 6.26 for details. This field is meaningful only if Hierarchy ID Valid is 1b. If Hierarchy ID Writeable is Set, this field is read-write and contains the value programmed by software. If Hierarchy ID Writeable is Clear, this field is read only. The value is determined using the rules defined in Section 7.9.18. This field defaults to 0000 0000h.	RO/RW

7.9.18.8 *Hierarchy ID GUID 5 Register*

Figure 7-261 and Table 7-212 detail allocation of fields in the Hierarchy ID GUID 5 Register.

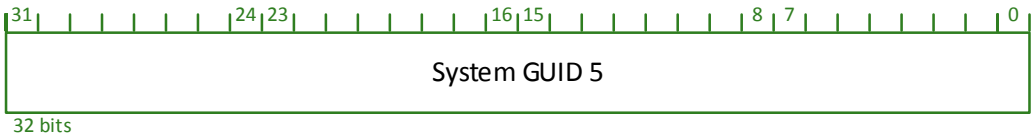


Figure 7-261: Hierarchy ID GUID 5 Register

Table 7-212: Hierarchy ID GUID 5 Register

Bit Location	Description	Attributes
31:0	**System GUID 5** – This field corresponds to bits [31:0] of the System GUID field in the Hierarchy ID Message. See Section 6.26 for details. This field is meaningful only if Hierarchy ID Valid is 1b. If Hierarchy ID Writeable is Set, this field is read-write and contains the value programmed by software. If Hierarchy ID Writeable is Clear, this field is read only. The value is determined using the rules defined in Section 7.9.18. This field defaults to 0000 0000h.	RO/RW

7.9.19 VPD Capability

Support of VPD is optional. All Functions are permitted to contain the capability. This includes all Functions of a Multi-Function Device associated with an Upstream Port as well as RCiEPs.

Vital Product Data (VPD) is information that uniquely identifies hardware and, potentially, software elements of a system. The VPD can provide the system with information on various Field Replaceable Units such as part number, serial number, and other detailed information. The objective from a system point of view is to make this information available to the system owner and service personnel. VPD typically resides in a storage device (for example, a serial EEPROM) associated with the Function.

Details of the VPD data is defined in Section 6.28.

Access to the VPD is provided using the Capabilities List in Configuration Space. The VPD capability structure is shown in Figure 7-262.

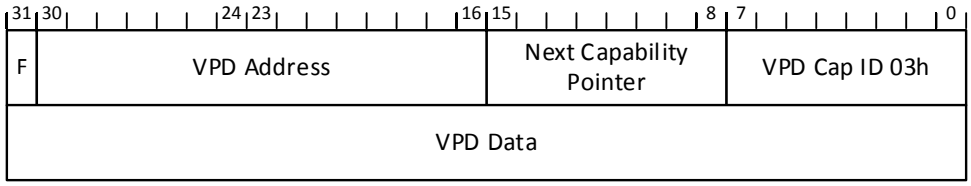


Figure 7-262: VPD Capability Structure

Table 7-213: VPD Register Fields

Bit Location	Description	Attributes
7:0 1st DW	ID – Value 03h to indicate the Vital Product Data Capability.	RO
15:8 1st DW	Next Capability Pointer – Pointer to the next capability structure, or 00h if this is the last structure in the Capability List.	RO
30:16 1st DW	VPD Address – DWORD-aligned byte address of the VPD to be accessed. Behavior is undefined if the low 2 bits of this field are non-zero. Default is indeterminate.	RW
31 1st DW	F – A flag used to indicate when the transfer of data between the VPD Data register and the storage component is completed. The flag register is written when the VPD Address register is written. To read VPD information, a zero is written to the flag register when the address is written to the VPD Address register. The hardware device will set the flag to a one when 4 bytes of data from the storage component have been transferred to the VPD Data register. Software can monitor the flag and, after it is set to a one, read the VPD information from the VPD Data register. If either the VPD Address or VPD Data register is written, prior to the flag bit being set to a one, the results of the original read operation are unpredictable. To write VPD information, to the read/write portion of the VPD space, write the data to the VPD Data register. Then write the address of where the VPD data is to be stored into the VPD Address register and write the flag bit to a one (at the time the address is written). The software then monitors the flag bit and when it is set to zero (by device hardware), the VPD data (all 4 bytes) has been transferred from the VPD Data register to the storage component. If either the VPD Address or VPD Data register is written, prior to the flag bit being set to a zero, the results of the write operation to the storage component are unpredictable.	RW
31:0 2nd DW	VPD Data – VPD data can be read through this register. The least significant byte of this register (at offset 4 in this capability structure) corresponds to the byte of VPD at the address specified by the VPD Address register. Behavior is undefined for any read or write of this register with Byte Enables other than 1111b. Reading or writing data outside of the VPD space in the storage component is not allowed. The VPD (both the read only items and the read/write fields) is stored information and will have no direct control of any device operations. The initial value of this register at power up is indeterminate.	RW

7.9.20 Native PCIe Enclosure Management Extended Capability

The Native PCIe Enclosure Management Extended (NPEM) Capability is an optional extended capability that is permitted to be implemented by Root Ports, Switch Downstream Ports, and Endpoints.

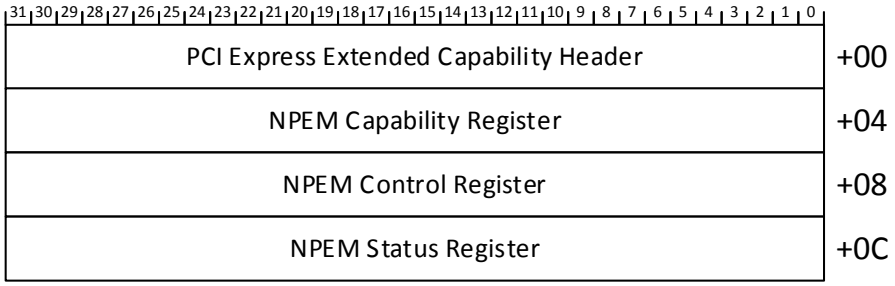


Figure 7-263: NPEM Extended Capability

7.9.20.1 *Native PCIe Enclosure Management Extended Capability Header (Offset 00h)*

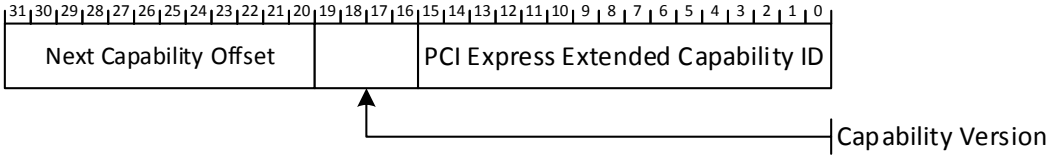


Figure 7-264: NPEM Extended Capability Header

Table 7-214: NPEM Extended Capability Header

Bit Location	Register Description	Attributes
15:0	PCI Express Extended Capability ID – This field is a PCI-SIG defined ID number that indicates the nature and format of the extended capability. PCI Express Extended Capability ID for the NPEM Extended Capability is 0029h.	RO
19:16	Capability Version – This field is a PCI-SIG defined version number that indicates the version of the capability structure present. Must be 1h for this version of the specification.	RO
31:20	Next Capability Offset – This field contains the offset to the next PCI Express Extended Capability structure or 000h if no other items exist in the linked list of capabilities.	RO

7.9.20.2 *Native PCIe Enclosure Management Capability Register (Offset 04h)*

The NPEM Capability Register contains an overall NPEM capable bit and a bit map of states supported in the implementation. Implementations are required to support OK, Locate, Fail, and Rebuild states if NPEM capable bit is Set. All other states are optional.

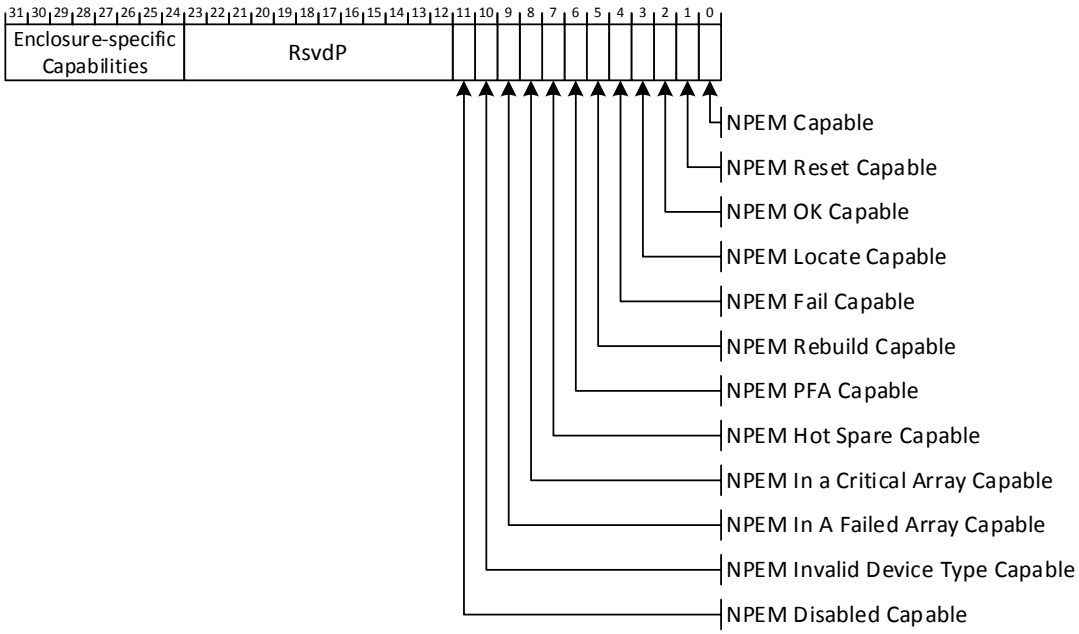


Figure 7-265: NPEM Capability Register

Figure 7-266: NPEM Capability Register

Bit Location	Register Description	Attributes
0	**NPEM Capable** - When Set, this bit indicates that the enclosure has NPEM functionality.	HwInit
1	**NPEM Reset Capable** - A value of 1b indicates support for the optional NPEM Reset mechanism described in Section 6.29. This capability is independently optional.	HwInit
2	**NPEM OK Capable** - When Set, this bit indicates that enclosure has the ability to indicate the NPEM OK state. This bit must be Set if NPEM Capable is also Set.	HwInit
3	**NPEM Locate Capable** - When Set, this bit indicates that enclosure has the ability to indicate the NPEM Locate state. This bit must be Set if NPEM Capable is also Set.	HwInit
4	**NPEM Fail Capable** - When Set, this bit indicates that enclosure has the ability to indicate the NPEM Fail state. This bit must be Set if NPEM Capable is also Set.	HwInit
5	**NPEM Rebuild Capable** - When Set, this bit indicates that enclosure has the ability to indicate the NPEM Rebuild state. This bit must be Set if NPEM Capable is also Set.	HwInit

Bit Location	Register Description	Attributes
6	**NPEM PFA Capable** - When Set, this bit indicates that enclosure has the ability to indicate the NPEM PFA state. This capability is independently optional.	HwInit
7	**NPEM Hot Spare Capable** - When Set, this bit indicates that enclosure has the ability to indicate the NPEM Hot Spare state. This capability is independently optional.	HwInit
8	**NPEM In A Critical Array Capable** - When Set, this bit indicates that enclosure has the ability to indicate the NPEM In A Critical Array state. This capability is independently optional.	HwInit
9	**NPEM In A Failed Array Capable** - When Set, this bit indicates that enclosure has the ability to indicate the NPEM In A Failed Array state. This capability is independently optional.	HwInit
10	**NPEM Invalid Device Type Capable** - When Set, this bit indicates that enclosure has the ability to indicate the NPEM_Invalid_ Device_Type state. This capability is independently optional.	HwInit
11	**NPEM Disabled Capable** - When Set, this bit indicates that enclosure has the ability to indicate the NPEM_Disabled state. This capability is independently optional.	HwInit
31:24	**Enclosure-specific Capabilities** - The definition of enclosure-specific bits is outside the scope of this specification.	HwInit

7.9.20.3 *Native PCIe Enclosure Management Control Register (Offset 08h)*

The NPEM Control Register contains an overall NPEM enable bit and a bit map of states that software controls.

Use of Enclosure-specific bits is outside the scope of this specification.

All writes to this register, including writes that do not change the register value, are NPEM commands and should eventually result in a command completion indication in the NPEM Status register.

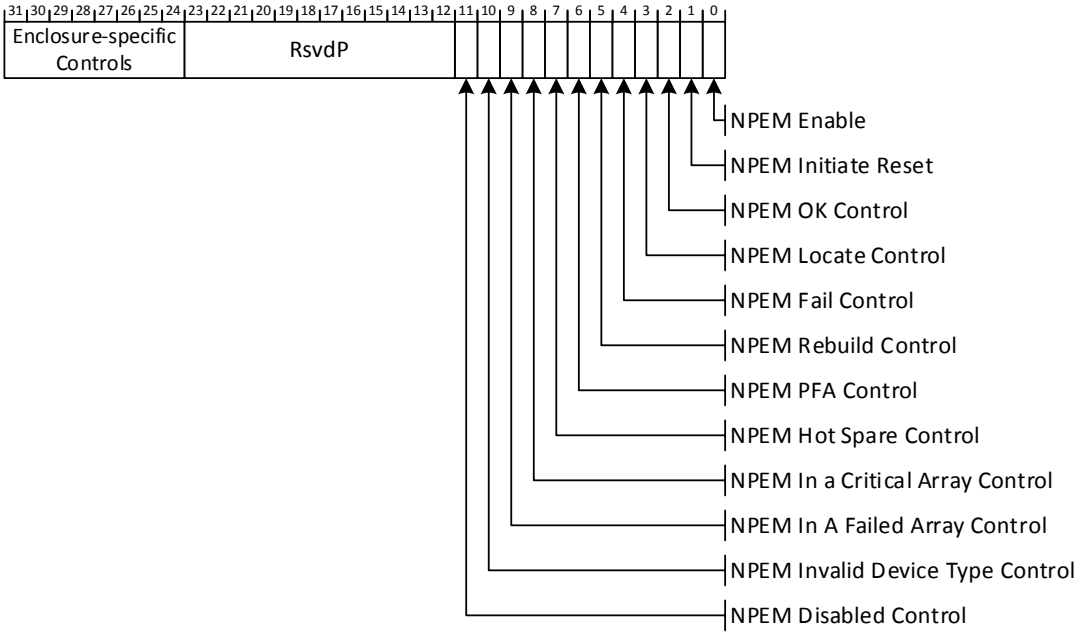


Figure 7-267: NPEM Control Register

Table 7-215: NPEM Control Register

Bit Location	Register Description	Attributes
0	**NPEM Enable** - When Set, this bit enables the NPEM capability. When Clear, this bit disables the NPEM capability. Default value of this bit is 0b. When enabled, this capability operates as defined in this specification. When disabled, the other bits in this capability have no effect and any associated indications are outside the scope of this specification.	RW
1	**NPEM Initiate Reset** – If NPEM Reset Capable bit is 1b, then a write of 1b to this bit initiates NPEM Reset. If NPEM Reset Capable bit is 0b, then this bit is permitted to be read-only with a value of 0b. The value read by software from this bit must always be 0b.	RW/RO
2	**NPEM OK Control** – When Set, this bit specifies that the NPEM OK indication be turned ON. When Clear, this bit specifies that the NPEM OK indication be turned OFF. If NPEM OK Capable bit in NPEM Capability register is 0b, this bit is permitted to be read-only with a value of 0b. Default value of this bit is 0b	RW/RO
3	**NPEM Locate Control** - When Set, this bit specifies that the NPEM Locate indication be turned ON. When Clear, this bit specifies that the NPEM Locate indication be turned OFF. If NPEM Locate Capable bit in the NPEM Capability register is 0b, this bit is permitted to be read-only with a value of 0b. Default value of this bit is 0b	RW/RO

4	**NPEM Fail Control** - When Set, this bit specifies that the NPEM Fail indication be turned ON. When Clear, this bit specifies that the NPEM Fail indication be turned OFF. If NPEM Fail Capable bit in the NPEM Capability register is 0b, this bit is permitted to be read-only with a value of 0b. Default value of this bit is 0b	RW/RO
5	**NPEM Rebuild Control** - When Set, this bit specifies that the NPEM Rebuild indication be turned ON. When Clear, this bit specifies that the NPEM Rebuild indication be turned OFF. If NPEM Rebuild Capable bit in NPEM Capability register is 0b, this bit is permitted to be read-only with a value of 0b. Default value of this bit is 0b	RW/RO
6	**NPEM PFA Control** - When Set, this bit specifies that the NPEM PFA indication be turned ON. When Clear, this bit specifies that the NPEM PFA indication be turned OFF. If NPEM PFA Capable bit in NPEM Capability register is 0b, this bit is permitted to be read-only with a value of 0b. Default value of this bit is 0b	RW/RO
7	**NPEM Hot Spare Control** - When Set, this bit specifies that the NPEM Hot Spare indication be turned ON. When Clear, this bit specifies that the NPEM Hot Spare indication be turned OFF. If NPEM Hot Spare Capable bit in NPEM Capability register is 0b, this bit is permitted to be read-only with a value of 0b. Default value of this bit is 0b	RW/RO
8	**NPEM In A Critical Array Control** - When Set, this bit specifies that the NPEM In A Critical Array indication be turned ON. When Clear, this bit specifies that the NPEM In A Ciritical Array indication be turned OFF. If NPEM In A Critical Array Capable bit in NPEM Capability register is 0b, this bit is permitted to be read-only with a value of 0b. Default value of this bit is 0b	RW/RO
9	**NPEM In A Failed Array Control** - When Set, this bit specifies that the NPEM In A Failed Array indication be turned ON. When Clear, this bit specifies that the NPEM In A Failed Array indication be turned OFF. If NPEM In A Failed Array Capable bit in NPEM Capability register is 0b, this bit is permitted to be read-only with a value of 0b. Default value of this bit is 0b	RW/RO
10	**NPEM Invalid Device Type Control** - When Set, this bit specifies that the NPEM Invaild Device Type indication be turned ON. When Clear, this bit specifies that the NPEM Invalid Device Type indication be turned OFF. If NPEM Invalid Device Type Capable bit in NPEM Capability register is 0b, this bit is permitted to be read-only with a value of 0b. Default value of this bit is 0b	RW/RO

11	**NPEM Disabled Control** - When Set, this bit specifies that the NPEM Disabled indication be turned ON. When Clear, this bit specifies that the NPEM Disabled indication be turned OFF. If NPEM Disabled Capable bit in NPEM Capability register is 0b, this bit is permitted to be read-only with a value of 0b. Default value of this bit is 0b	RW/RO
31:24	**Enclosure-specific Controls** - The definition of enclosure-specific bits is outside the scope of this specification. Enclosure-specific software is permitted to change the value of this field. Other software must preserve the existing value when writing this register. Default value of this field is 00h	RW/RO

7.9.20.4 *Native PCIe Enclosure Management Status Register (Offset 0Ch)*

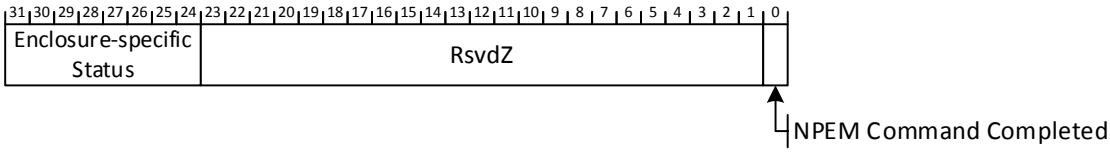


Figure 7-268: NPEM Status Register

Table 7-216: NPEM Status Register

Bit Location	Register Description	Attributes
0	NPEM Command Completed - This bit is Set as an indication to host software that an NPEM command has completed successfully. Default value of this bit is 0b. Software must wait for an NPEM command to complete before issuing the next NPEM command. However, if this bit is not set within 1 second limit on command execution, software is permitted to repeat the NPEM command or issue the next NPEM command. If software issues a write before the Port has completed processing of the previous command and before the 1 second time limit has expired, the Port is permitted to either accept or discard the write. Such a write is considered a programming error, and could result in a discrepancy between the NPEM Control register and the enclosure element state. To recover from such a programming error and return the enclosure to a consistent state, software must issue a write to the NPEM Control register which conforms to the NPEM command completion rules.	RW1C
31:24	Enclosure-specific Status - The definition of enclosure specific bits is outside the scope of this specification. Enclosure specific software is permitted to write non-zero values to this field. Other software must write 00h to this field. The default value of this field is enclosure-specific. This field is permitted to be hardwired to 00h.	RsvdZ / RO / RW1C

8

8 Electrical Sub-Block

8.1 Electrical Specification Introduction

Key attributes of the Electrical Specification Include:

- Support for 2.5, 5,0, 8.0, and 16.0 GT/s data rates
- Support for common and separate independent reference clock architectures
- Support for Spread Spectrum clocking
- Reduced swing mode for lower power Link operation
- In-band receiver detection and electrical idle detection
- Channel compliance methodology
- Adaptive transmitter equalization and reference receiver equalization allowing closed eye channel support at 8.0 and 16.0 GT/s
- Lane margining
- AC coupled channel

The 4.0 version of the PCI Express electrical specification is organized into separate sections for the Transmitter, Receiver, the channel, and the Refclk. In this version most parameters have been regularized such that a common set of parameters is used to define compliance at all four data rates.

8.2 Interoperability Criteria

8.2.1 Data Rates

A PCIe device may operate with the following data rates: 2.5 GT/s only, 2.5 and 5.0 GT/s, 2.5, 5.0, and 8.0 GT/s, or 2.5, 5.0, 8.0, and 16.0 GT/s. In other words, a device must support 2.5 GT/s and is not permitted to skip any data rates between 2.5 GT/s and the highest supported.

8.2.2 Refclk Architectures

PCIe supports two Refclk data architectures: Common Refclk, and Independent Refclk. These are described in detail in section 8.6. A PCIe device may support one or more of these architectures.

8.3 Transmitter Specification

8.3.1 Measurement Setup for Characterizing Transmitters

The PCI Express electrical specification references all measurements to the device's pin. However, the pin of a device under test (DUT) is not generally accessible, and the closest accessible point is usually a pair of microwave-type coaxial connectors separated from the DUT pins by several inches of PCB trace, called the breakout channel on a silicon test board. On a test board with many Lanes the minimum breakout channel length is constrained by the need to route to a large number of coaxial connectors. Typically, this limitation holds true for both the Tx and the Rx pins. Figure 8-2 illustrates a typical test connection to a DUT, showing a single Tx Lane breakout.

A low jitter Refclk source is used when the silicon supports using an external reference clock in order that the jitter measurements for the DUT include only contributions from the Transmitter.

When testing a Transmitter it is desirable to have as many other PCI Express Lanes sending or receiving the compliance pattern as is feasible. Similarly, if the device supports other I/O it should also be sending or receiving on these interfaces. The goal is to have the Tx test environment replicate that found in a real system as closely as possible.

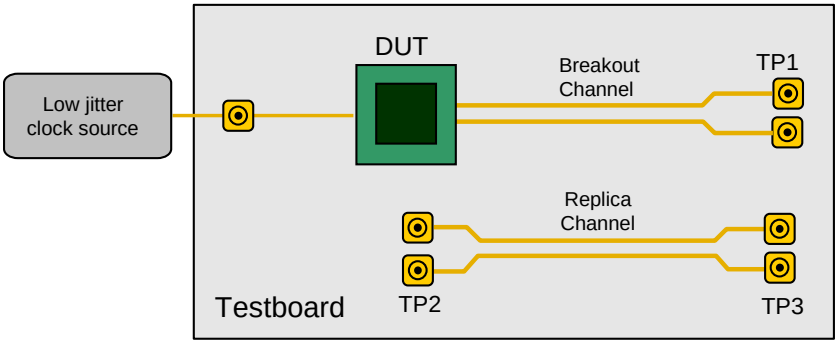


Figure 8-1: Tx Test Board for Non-Embedded Refclk

The 4.0 version of the Tx specification also includes explicit support for Transmitters utilizing embedded Refclks. In this case the Tx under test is not driven with a low jitter clock source, and both the Tx data and Tx Refclk out must be sampled simultaneously by means of a 2-port measurement. For more details consult Section 8.3.5.3. When an implementation is tested that is configured for the independent reference clock architecture only the data is sampled for both the Non-Embedded and Embedded reference clock cases.

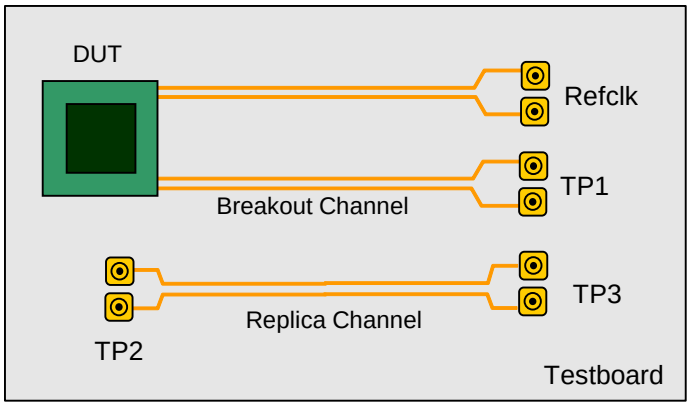


Figure 8-2: Tx Test board for Embedded Refclk

8.3.1.1 *Breakout and Replica Channels*

In order to specify a Transmitter with a uniform set of Tx parameters it is necessary to establish a one-to-one correspondence between what is measurable at TP1 and the corresponding Tx voltage or jitter at the pin. This may be achieved by means of a breakout channel and a replica channel. The replica channel reproduces the electrical characteristics of the breakout channel as closely as possible, matching its length, layer transitions, etc, making it possible to de-embed Tx measurements to the pin of the DUT. All voltage parameters are de-embedded to the pins unless otherwise specified. While the specification does not define precise electrical characteristics for the replica and breakout channels, it is advisable to adhere to the following guidelines:

- Breakout channels should be as short as feasible, less than 6 inches if possible. The longer the channel the more HF information will be lost.
- Breakout channels should be the same length for each Lane and routed on as few layers as possible, thereby reducing the number of replica channels that need to be built and measured.
- Each routing layer on a test board should have a separate breakout channel where the via and pad structures of the breakout and replica channels on respective layers match as closely as possible.
- Break-out channels should be designed to have an insertion loss of less than 2 dB (4 dB if the maximum signaling rate is 16.0 GT/s) and a return loss of greater than 15 dB at 8 GHz, which may require use of low loss dielectric, wide signal traces and back-drilling of break-out vias or use of micro-via technology.
- The impedance targets for the breakout channel are 100 Ω differential and 50 Ω single-ended. For best accuracy the actual breakout channel impedance should be within ±5% of these values. For larger deviations a more complex de-embedding technique may be required.

8.3.2 Voltage Level Definitions

A differential voltage is defined by taking the voltage difference between two conductors. In this specification, a differential signal or differential pair is comprised of a voltage on a positive conductor, V_{D+}, and a negative conductor, V_{D-}. The differential voltage (V_{DIFF}) is defined as the

difference of the positive conductor voltage and the negative conductor voltage ($V_{DIFF} = V_{D+} - V_{D-}$). The Common Mode Voltage (V_{CM}) is defined as the average or mean voltage present on the same differential pair ($V_{CM} = [V_{D+} + V_{D-}]/2$). This document's electrical specifications often refer to common mode peak-to-peak measurements or peak measurements, which are defined by the following equations.

- $V_{DIFFp\text{-}p} = (2*\max|V_{D+} - V_{D-}|)$ (This applies to a symmetric differential swing)

Equation 8-1

- $V_{TX\text{-}AC\text{-}CM\text{-}PP} = \max(V_{D+} + V_{D-})/2 - \min(V_{D+} + V_{D-})/2$

Equation 8-2

Note: The maximum value is calculated on a per unit interval evaluation with unit interval boundaries determined by the behavioral CDR. The maximum function as described is implicit for all peak-to-peak and peak common mode equations throughout the rest of this chapter.

In this section, DC is defined as all frequency components below F_{DC} = 30 kHz. AC is defined as all frequency components at or above F_{DC} = 30 kHz. These definitions pertain to all voltage and current specifications.

An example waveform is shown in Figure 8-3. In this waveform the differential voltage (defined as D+ - D-) is approximately 800 mVPP, and the single-ended voltage for both D+ and D- is approximately 400 mVPP for each. Note that while the center crossing point for both D+ and D- is nominally at 200 mV, the corresponding crossover point for the differential voltage is at 0.0 V.

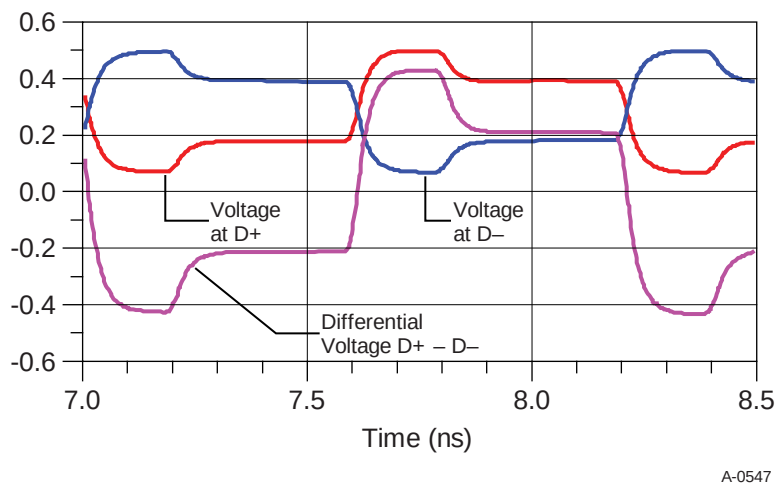


Figure 8-3: Single-ended and Differential Levels

8.3.3 Tx Voltage Parameters

Tx voltage parameters include equalization coefficients, equalization presets, and min/max voltage swings.

8.3.3.1 *2.5 and 5.0 GT/s Transmitter Equalization*

Tx equalization at 2.5 and 5.0 GT/s is only de-emphasis. Tx equalization de-emphasis values at 2.5 and 5.0 GT/s are measured using the average ratio of transition to non-transition average eye amplitude at the 0.5 UI location using 500 repetitions of the compliance pattern.

8.3.3.2 *8.0 and 16.0 GT/s Transmitter Equalization*

Tx voltage swing and equalization presets at 8.0 and 16.0 GT/s are measured by means of a low frequency pattern within the compliance pattern. Consisting of a sequence of 64 zeroes followed by 64 ones, this pattern permits an accurate measurement of voltage since ISI effects will have decayed and the signal will have approached a steady state. 8.0 and 16.0 GT/s Transmitter must implement a coefficient-based equalization mode in order to support fine grained control over Tx equalization resolution. Additionally, a Transmitter must support a specified number of presets that give a coarser control over Tx equalization resolution. Both coefficient space and preset space are controllable via messaging from the Receiver via an equalization procedure. The equalization procedure operates on the same physical path as normal signaling and is implemented via extensions to the existing protocol Link layer.

All 8.0 and 16.0 GT/s Transmitters must implement support for the equalization procedure, whereas 8.0 GT/s and 16.0 GT/s Receivers may optionally make use of requests for the Transmitter on the Link partner to update Transmitter equalization. Details of the equalization procedure may be found in the Physical Layer Logical Block chapter.

Tx equalization coefficients are based on the following FIR filter relationship as shown in Figure 8-4. Equalization coefficients are subject to constraints limiting their max swing to ±unity with c_{-1} and c_{+1} being zero or negative. The inclusion of the unity condition means that only two of the three coefficients need to be specified to fully define v_out_n. In this specification the two coefficients so specified are c_{-1} and c_{+1}, where c_0 is implied. Note that the coefficient magnitude is not the same as the Tx voltage swing magnitude.

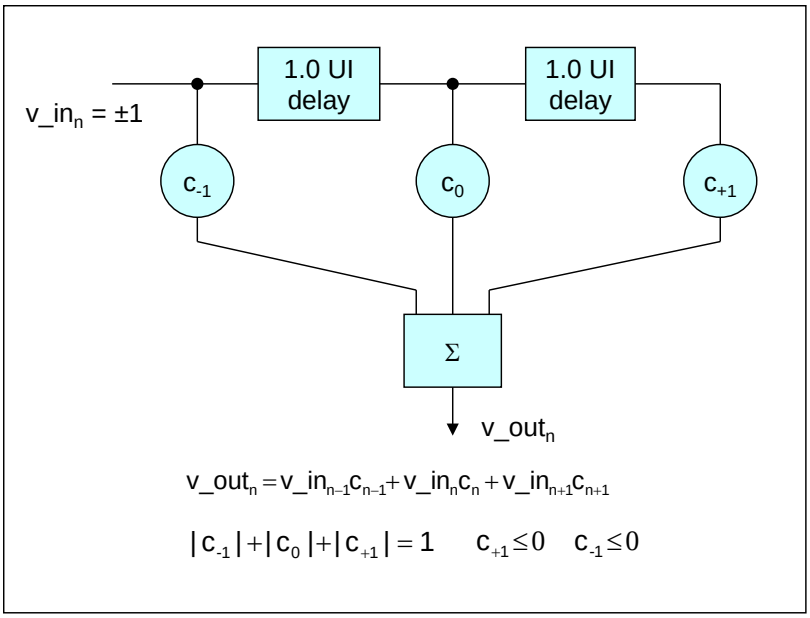
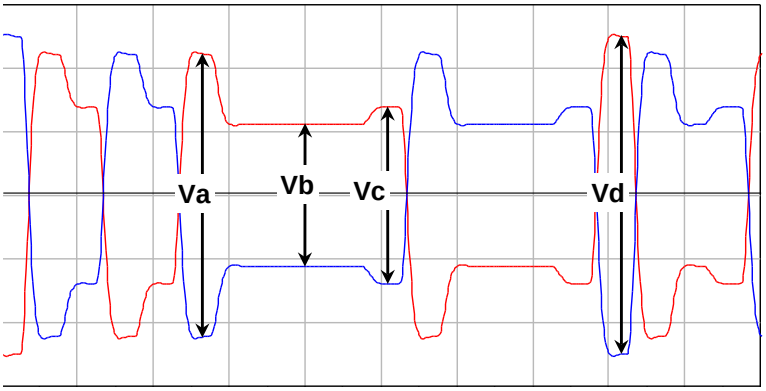


Figure 8-4: Tx Equalization FIR Representation

8.3.3.3 *Tx Equalization Presets*

8.0 GT/s and 16.0 GT/s PCIe signaling must support the full range of presets given in Table 8-1. Presets are defined in terms of ratios, relating the pre-cursor and post-cursor equalization voltages. The pre-cursor (Vc) is referred to as pre-shoot, while the post-cursor (Vb) is referred to as de-emphasis. This convention permits the specification to retain the existing 2.5 GTs/s/5.0 GT/s definitions for Tx equalization, where only de-emphasis is defined, and it allows pre-shoot and de-emphasis to be defined such that each is independent of the other. The tolerances in Table 8-1 also appy to 2.5 and 5.0 GT/s de-emphasis. The maximum swing, Vd, is also shown to illustrate that, when both c_{+1} and c_{-1} are non-zero, the swing of Va does not reach the maximum as defined by Vd. Figure 8-5 is shown as an example of transmitter equalization, but it is not intended to represent the signal as it would appear for measurement purposes. The high frequency nature of PCIe signaling makes measurement of single UI pulse heights impractical. Consequently all amplitude measurements are made with low frequency waveforms, as shown in Figure 8-6 and Figure 8-7.

The presets defined in Table 8-1 and Table 8-2 are numbered to match the designations in the Physical Layer Logical Block chapter. Note that the coefficients in the Physical Layer Logical Block chapter are defined as integers (using upper case "C"), while in this section they are defined as floating point (using lower case "c").



De-emphasis = 20 $\log_{10}$Vb/Va
Preshoot = 20$\log_{10}$Vc/Vb
Boost = 20$\log_{10}$ Vd/Vb

Figure 8-5: Definition of Tx Voltage Levels and Equalization Ratios

Table 8-1 lists the values for presets; at 8.0 GT/s and 16.0 GT/s all preset values must be supported for full swing signaling.

Table 8-1. Tx Preset Ratios and Corresponding Coefficient Values

Preset #	Preshoot (dB)	De-emphasis (dB)	c-1	c+1	Va/Vd	Vb/Vd	Vc/Vd
P4	0.0	0.0	0.000	0.000	1.000	1.000	1.000
P1	0.0	-3.5 ± 1 dB	0.000	-0.167	1.000	0.668	0.668
P0	0.0	-6.0 ± 1.5 dB	0.000	-0.250	1.000	0.500	0.500
P9	3.5 ± 1 dB	0.0	-0.166	0.000	0.668	0.668	1.000
P8	3.5 ± 1 dB	-3.5 ± 1 dB	-0.125	-0.125	0.750	0.500	0.750
P7	3.5 ± 1 dB	-6.0 ± 1.5 dB	-0.100	-0.200	0.800	0.400	0.600
P5	1.9 ± 1 dB	0.0	-0.100	0.000	0.800	0.800	1.000
P6	2.5 ± 1 dB	0.0	-0.125	0.000	0.750	0.750	1.000
P3	0.0	-2.5 ± 1 dB	0.000	-0.125	1.000	0.750	0.750
P2	0.0	-4.4 ± 1.5 dB	0.000	-0.200	1.000	0.600	0.600
P10	0.0	Note 2.	0.000	Note 2.	1.000	Note 2.	Note 2.

Notes:

1. Reduced swing signaling must implement presets P4, P1, P9, P5, P6, and P3. Full swing signaling must implement all the above presets.
2. P10 boost limits are not fixed, since its de-emphasis level is a function of the LF level that the Tx advertises during training. P10 is used for testing the boost limit of Transmitter at full swing. P1 is used for testing the boost limit of Transmitter at reduced swing.

8.3.3.4 *Measuring Tx Equalization for 2.5 GT/s and 5.0 GT/s*

Tx equalization de-emphasis values at 2.5 and 5.0 GT/s are measured using the average ratio of transition to non-transition eye heights at the 0.5 UI location using 500 repetitions of the compliance pattern.

8.3.3.5 *Measuring Presets at 8.0 GT/s and 16.0 GT/s*

Figure 8-6 and Figure 8-7 illustrate the waveforms observed when measuring presets using the 64-zeroes/64-ones sequence that is part of the compliance pattern. Depending on whether the preset implements de-emphasis, preshoot, or both, the corresponding waveshape will differ. The two cases illustrated below show preshoot and de-emphasis as can be observed by noting whether the equalization occurs before or after an edge transition. For the case where both de-emphasis and preshoot are present, boost occurs both before and after each edge transition. In all cases the voltage of interest occurs during the flat portion (at Vb) of the waveform, where it can be accurately measured independent of high frequency effects. Measurements of Vb are made using the average voltage from UI 57 to 62 in all the 64 zeroes and 64 ones sequences over 500 repetitions of the compliance pattern.

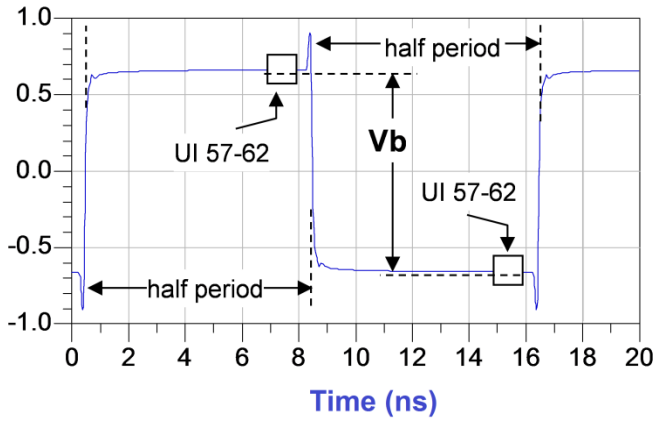


Figure 8-6: Waveform Measurement Points for Pre-shoot

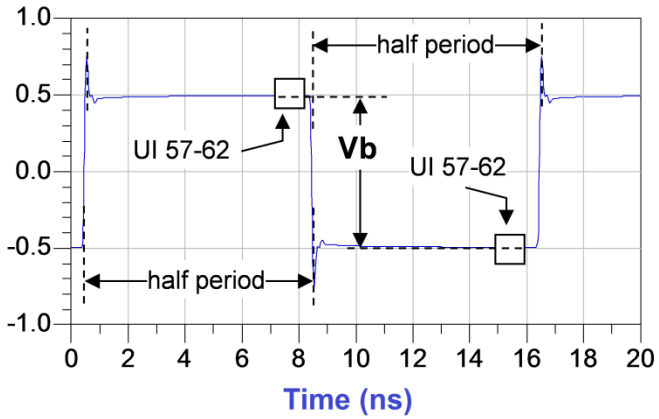


Figure 8-7: Waveform Measurement Points for De-emphasis

With the exception of P4 (for which both pre-shoot and de-emphasis are 0.0 dB) it is not possible to obtain a direct measurement of Va and Vc, because these portions of the waveform are 1 UI wide and therefore subject to attenuation by the package and the breakout channel. Instead the Va and Vc values are obtained by setting the DUT to a different preset value where the former's Va or Vc voltage occurs during the latter's Vb interval. Table 8-2 lists the preset values required to measure

each of the Va and Vc values from which their preshoot or de-emphasis values may be derived from the ratio of Vb values for the indicated presets.

Table 8-2. Preset Measurement Cross Reference Table

Preset Number	De-emphasis (dB) 20log10 (Vb(i)/Vb(j))	Preshoot (dB)
20log10 (Vb(i)/Vb(j))		
P4	N/A	N/A
P1	P1/P4	N/A
P0	P0/P4	N/A
P9	N/A	P4/P9
P8	P8/P6	P3/P8
P7	P7/P5	P2/P7
P5	N/A	P4/P5
P6	N/A	P4/P6
P3	P3/P4	N/A
P2	P2/P4	N/A
P10	P10/P4	N/A

8.3.3.6 *Method for Measuring $V_{TX\text{-}DIFF\text{-}PP}$ at 2.5 GT/s and 5.0 GT/s*

$V_{TX\text{-}DIFF\text{-}PP}$ ($V_{TX\text{-}DIFF\text{-}PP\text{-}LOW}$ for reduced swing) at 2.5 GTs and 5.0 GT/s are measured using the average transition eye amplitude at the 0.5 UI location using 500 repetitions of the compliance pattern.

8.3.3.7 *Method for Measuring $V_{TX\text{-}DIFF\text{-}PP}$ at 8.0 GT/s and 16.0 GT/s*

The range for a Transmitter's output voltage swing, (specified by Vd) with no equalization is defined by $V_{TX\text{-}DIFF\text{-}PP}$ ($V_{TX\text{-}DIFF\text{-}PP\text{-}LOW}$ for reduced swing), and is obtained by setting c_{-1} and c_{+1} to zero and measuring the peak-peak voltage on the 64-ones/64-zeroes segment of the compliance pattern. The resulting signal effectively measures at the die pad, minus any low frequency package loss. ISI and switching effects are minimized by restricting the portion of the curve over which voltage is measured to the last few UI of each half cycle, as illustrated in Figure 8-8. High frequency noise is mitigated by averaging over 500 repetitions of the compliance pattern.

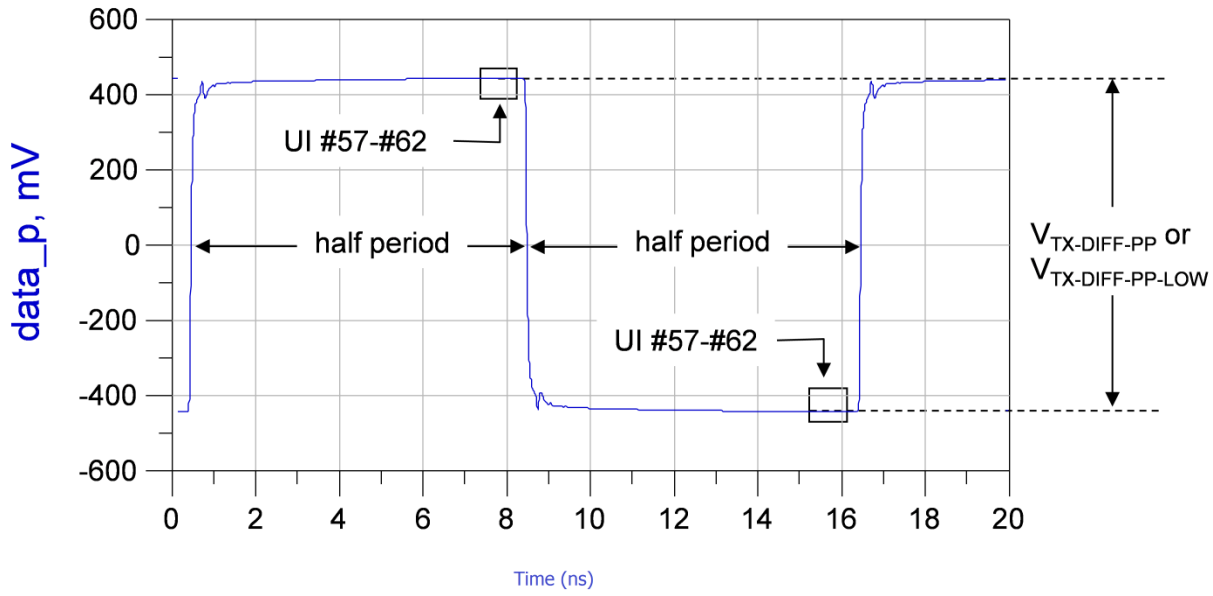


Figure 8-8: $V_{TX\text{-}DIFF\text{-}PP}$ and $V_{TX\text{-}DIFF\text{-}PP\text{-}LOW}$ Measurement

8.3.3.8 *Coefficient Range and Tolerance*

8.0 GT/s and 16.0 GT/s Transmitters are required to inform the Receiver of their coefficient range and tolerance. Coefficient range and tolerance are constrained by the following requirements.

- Coefficients must support all eleven presets and their respective tolerances as defined in Table 8-2
- All Transmitters must meet the full swing signaling $V_{TX\text{-}EIEOS\text{-}FS}$ limits.
- Transmitters may optionally support reduced swing, and if they do, they must meet the $V_{TX\text{-}EIEOS\text{-}RS}$ limits.
- The coefficients must meet the boost and resolution ($V_{TX\text{-}BOOST\text{-}FS}$, $V_{TX\text{-}BOOST\text{-}RS}$ and $EQ_{TX\text{-}COEFF\text{-}RES}$) limits defined in Table 8-7:.

When the above constraints are applied the resulting coefficient space may be mapped onto a triangular matrix, an example of which is shown in Table 8-3. The matrix may be interpreted as follows: Pre-shoot and de-emphasis coefficients are mapped onto the Y-axis and X-axes, respectively. In both cases the maximum granularity of 1/24 is assumed. Each matrix cell, corresponding to a valid combination of preshoot and de-emphasis coefficients, has three entries corresponding to preshoot (PS), de-emphasis (DE), and boost (as shown in the upper left hand corner). Diagonal elements are defined by the maximum boost ratio. Those cells highlighted in blue are presets required for reduced swing, while cells in either blue or orange represent presets required for full swing signaling. Note that this figure is informative only and is not intended to imply any particular Tx implementation or to alter requirements for nominal preset equalization values and allowed ranges.

Table 8-3: Transmit Equalization Coefficient Space Triangular Matrix Example

Min Reduced Swing Limit

PS DE		C_{+1}								
BOOST		0/24	1/24	2/24	3/24	4/24	5/24	6/24	7/24	8/24
C_{-1}	0/24	0.0 0.0 0.0	0.0 -0.8 0.8	0.0 -1.6 1.6	0.0 -2.5 2.5	0.0 -3.5 3.5	0.0 -4.7 4.7	0.0 -6.0 6.0	0.0 -7.6 7.6	0.0 -9.5 9.5
	1/24	0.8 0.0 0.8	0.8 -0.8 1.6	0.9 -1.7 2.5	1.0 -2.8 3.5	1.2 -3.9 4.7	1.3 -5.3 6.0	1.6 -6.8 7.6	1.9 -8.8 9.5	
	2/24	1.6 0.0 1.6	1.7 -0.9 2.5	1.9 -1.9 3.5	2.2 -3.1 4.7	2.5 -4.4 6.0	2.9 -6.0 7.6	3.5 -8.0 9.5		
	3/24	2.5 0.0 2.5	2.8 -1.0 3.5	3.1 -2.2 4.7	3.5 -3.5 6.0	4.1 -5.1 7.6	4.9 -7.0 9.5			
	4/24	3.5 0.0 3.5	3.9 -1.2 4.7	4.4 -2.5 6.0	5.1 -4.1 7.6	6.0 -6.0 9.5				
	5/24	4.7 0.0 4.7	5.3 -1.3 6.0	6.0 -2.9 7.6	7.0 -4.9 9.5					
	6/24	6.0 0.0 6.0	6.8 -1.6 7.6	8.0 -3.5 9.5						

Full swing Limit or
Max reduced swing limit

8.3.3.9 *EIEOS and $V_{TX\text{-}EIEOS\text{-}FS}$ and $V_{TX\text{-}EIEOS\text{-}RS}$ Limits*

EIEOS signaling is defined for 5.0 GT/s, 8.0 GT/s and 16.0 GT/s only. At 5.0 GT/s the K28.7 Symbol is used. $V_{TX\text{-}EIEOS\text{-}FS}$ and $V_{TX\text{-}EIEOS\text{-}RS}$ are measured using the EIEOS sequence contained within the compliance pattern for 8.0 GT/s and 16.0 GT/s. At 8.0 GT/s the EIEOS pattern consists of eight consecutive ones followed by the same number of consecutive zeroes, where the pattern is repeated for a total of 128 UI. At 16.0 GT/s the EIEOS patterns consists of 16 consecutive ones followed by the same number of consecutive zeroes, where the pattern is repeated for a total of 128 UI.

A transmitter sends an EIEOS to cause an exit of Electrical Idle at the Receiver. This pattern guarantees the Receiver will properly detect the EI Exit condition with its squelch exit detect circuit, something not otherwise guaranteed by scrambled data. The Tx EIEOS launch voltage is defined by $V_{TX\text{-}EIEOS\text{-}FS}$ for full swing signaling and by $V_{TX\text{-}EIEOS\text{-}RS}$ for reduced swing signaling. $V_{TX\text{-}EIEOS\text{-}RS}$ is smaller than $V_{TX\text{-}EIEOS\text{-}FS}$ to reflect the fact that reduced swing is typically supported only for lower loss channels where there is less attenuation at the EIEOS signaling rate.

For full swing signaling $V_{TX\text{-}EIEOS\text{-}FS}$ is measured with a preset number P10. This is equivalent to a maximum nominal boost of 9.5 dB and represents the maximum boost attainable in coefficient space. When a tolerance of ±1.5 dB is factored in this yields the minimum boost limit of 8.0 dB appearing in Table 8-7:. For reduced swing signaling $V_{TX\text{-}EIEOS\text{-}RS}$ is measured with preset P1.

A Transmitter is not always permitted to generate the maximum boost level noted above. In particular, a Transmitter that cannot drive significantly more than 800 mVPP is limited by the need to meet $V_{TX\text{-}EIEOS\text{-}FS}$. The Tx must reject any adjustments to its presets or coefficients that would violate the $V_{TX\text{-}EIEOS\text{-}FS}$ or $V_{TX\text{-}EIEOS\text{-}RS}$ limits. The EIEOS voltage limits are imposed to guarantee the EIEOS threshold of 175 mVPP at the Rx pin.

Figure 8-9 illustrates the de-emphasis peak as observed at the pin of a Tx for $V_{TX\text{-}EIEOS\text{-}FS}$. At the far end of a lossy channel the de-emphasis peak will be attenuated; this is why the measurement interval includes only the middle five UI at 8.0 GT/s and UI number 5-14 at 16.0 GT/s. The voltage is

averaged over this interval for both the negative and positive halves of the waveform over 500 repetitions of the compliance pattern.
$V_{TX\text{-}EIEOS\text{-}FS}$ and $V_{TX\text{-}EIEOS\text{-}RS}$ are defined as the difference between the negative and positive waveform segment averages. UI boundaries are defined with respect to the edge of the recovered data clock.

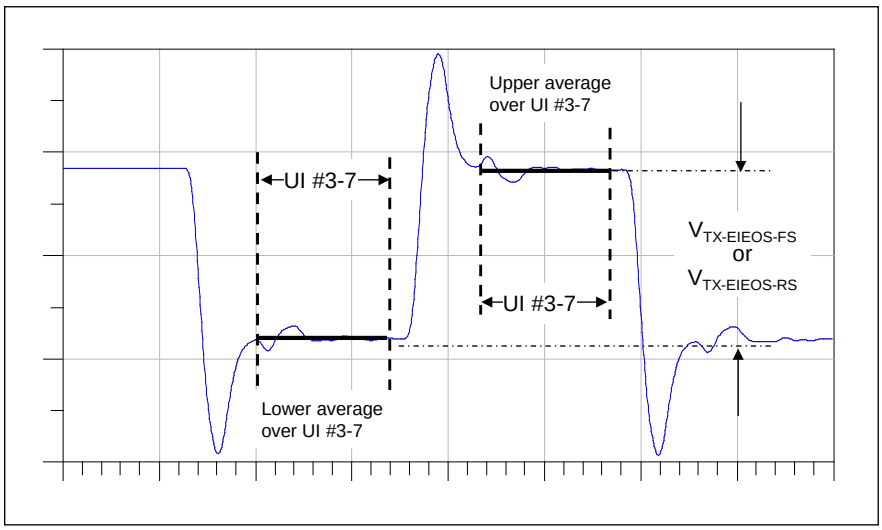


Figure 8-9: Measuring $V_{TX\text{-}EIEOS\text{-}FS}$ and $V_{TX\text{-}EIEOS\text{-}RS}$ at 8.0 GT/s

8.3.3.10 *Reduced Swing Signaling*

PCI Express Transmitters may optionally support a reduced swing signaling. It is left as an implementation option to define the maximum reduced swing voltage value for $V_{TX\text{-}DIFF\text{-}PP\text{-}LOW}$ anywhere up to the maximum full swing voltage. The minimum for $V_{TX\text{-}DIFF\text{-}PP\text{-}LOW}$ is captured indirectly by the constraint imposed by $V_{TX\text{-}EIEOS\text{-}RS}$, so there is no need to define a separate minimum limit for $V_{TX\text{-}DIFF\text{-}PP\text{-}LOW}$. Reduced swing limits the range of presets and the maximum boost. The boost for reduced swing must be in the region shown in Table 8-3 between the Max reduced swing limit and minimum reduced swing boost limit.

Form factors are permitted to disallow, optionally allow, or require Reduced Swing Signaling, depending on the channel requirements for the form factor. When Reduced Swing Signaling is allowed or required it is required that form factor specifications provide any additional details necessary to support interoperability.

8.3.3.11 *Effective Tx Package Loss at 8.0 GT/s and 16.0 GT/s*

Package loss (including silicon driver bandwidth) is representedby the $ps21_{TX}$ parameter. Since both package IL and driver bandwidth affect the signal as observed at the Tx pin, the $ps21_{TX}$ parameter has the advantage of representingboth of these effects, while permitting the measurement to be made at a point (TP1) that can easily be probed. It is necessary to include a package loss parameter in the Tx specification, since the voltage swing parameters ($V_{TX\text{-}DIFF\text{-}PP}$ and $V_{TX\text{-}DIFF\text{-}PP\text{-}LOW}$) are defined at an equivalent pulse frequency of 1/128 UI and purposely do not capture high frequency driver or package loss effects.

At 16.0 GT/s, separate $ps21_{TX}$ parameters are defined for packages containing Root Ports (Root Package) and for all other packages (Non-Root Package), based on the assumption that the former tend to be large and require socketing, while the latter are smaller and usually not socketed.

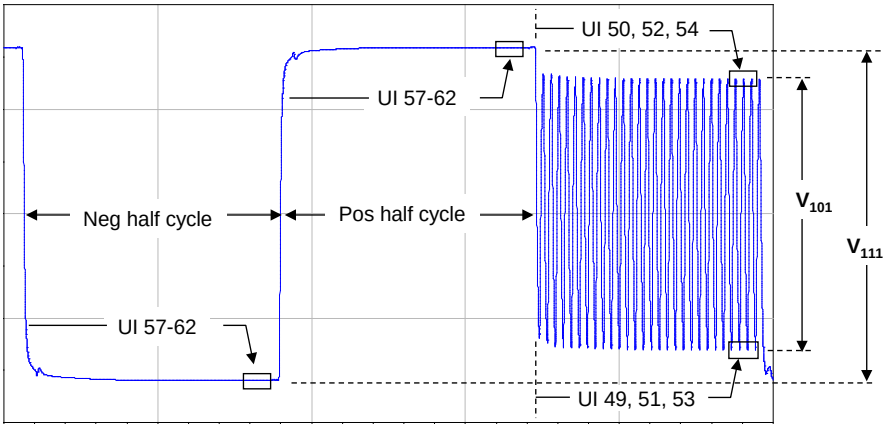
The $ps21_{TX}$ parameter is informative for 16.0 GT/s Root Package devices and for Non-Root Package devices that only support PCI Express standard form factors (i.e., CEM, M.2, etc.). The $ps21_{TX}$ parameter is normative for all 8.0 GT/s devices and for 16.0 GT/s Non-Root Package devices that support captive channels. (See Table 8-4 below).

Table 8-4. Cases that the Reference Packages and ps21TX Parameter are Normative

	8.0 GT/s		16.0 GT/s	
	Root Package Device	Non-Root Package Device	Root Package Device	Non-Root Package Device
Device supports captive channels	Normative	Normative	Informative	Normative
Device does not support captive channels	Normative	Normative	Informative	Informative

All implementations of PCI Express standard form factors must still meet form factor requirements. Devices for which the $ps21_{TX}$ parameter is informative, as defined above must provide a package model for use in channel compliance if they do not meet the informative $ps21_{TX}$ parameter.

Package loss is measured by comparing the 64-zeroes/64-ones voltage swing (V_{111}) against a 1010 pattern (V_{101}). Tx package loss measurement is made with c_{-1} and c_{+1} both set to zero. Measurements shall be made averaging over 500 repetitions of the compliance pattern.



$$ps21_{TX} = 20\log_{10}(V_{101}/V_{111})$$

Figure 8-10: Compliance Pattern and Resulting Package Loss Test Waveform

Measurement of V_{101} and V_{111} is made towards the end of each interval to minimize ISI and low frequency effects. V_{101} is defined as the peak-peak voltage between minima and maxima of the clock pattern. V_{111} is defined as the average voltage difference between the positive and negative levels of the two half cycles. The measurement should be averaged over 500 repetitions of the compliance pattern.

8.3.4 Transmitter Margining

Transmitters shall implement a margining procedure that allows the Tx launch voltage to be adjusted. Margining is enabled by programming a register set. Due to the larger range of Transmitter equalization, 8.0 GT/s and 16.0 GT/s Tx margining is subject to additional constraints: Tx margining at these speeds shall not require any coefficient or preset resolution finer than can be generated with 1/24 coefficient resolution defined for normal operation, and shall not require more Tx accuracy or capability than is required to support normal operation. It is acceptable that Vb fall below the limit set by $V_{TX\text{-}EIEOS\text{-}FS}$ or $V_{TX\text{-}EIEOS\text{-}RS}$, although proper end to-end operation is no longer guaranteed. Transmitter equalization accuracy requirements do not need to be met during margining. A Transmitter is not required to change the FS/LF values it sends in TS1 Ordered Sets during margining from the values used in normal operation.

There are 8 encoded values for transmit margin from 000b to 111b. Encoding 000b represents the normal operating range. For all supported data rates and Tx signalling mode (full swing or reduced swing), encoding 001b must produce a $V_{TX\text{-}DIFF\text{-}PP}$ compliant with the specification limits. At least three additional encodings with monotonically decreasing values for $V_{TX\text{-}DIFF\text{-}PP}$ must be supported for each data rate and Tx swing.mode. For full swing signalling there must be at least one encoding with index 100b or higher that produces a $V_{TX\text{-}DIFF\text{-}PP}$ between 200 and 400 mV. For reduced swing signalling there must be at least one encoding with value 100b or higher that produces a $V_{TX\text{-}DIFF\text{-}PP}$ between 100 and 200 mV.

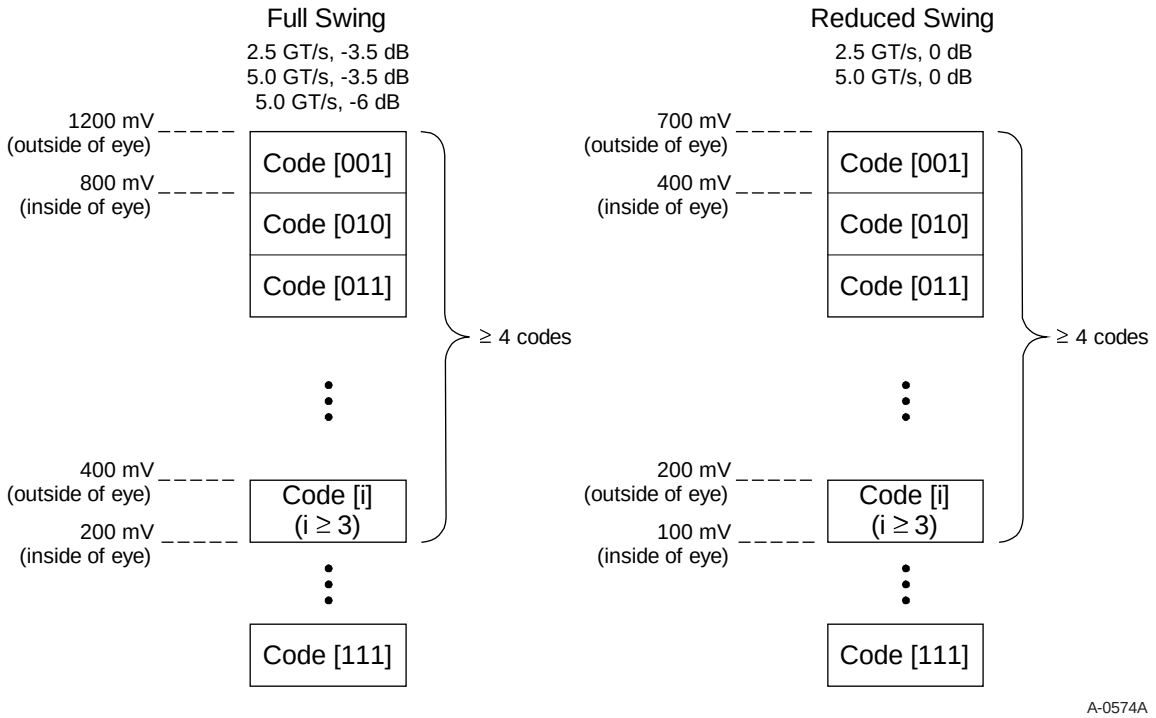


Figure 8-11: 2.5 and 5.0 GT/s Transmitter Margining Voltage Levels and Codes

8.3.5 Tx Jitter Parameters

Jitter limits are defined identically for all four data rates, although their respective values will vary with data rate. Jitter is measured at the zero crossing point at full speed using the compliance pattern. When measuring a particular Tx Lane it is necessary to ensure that all other PCI Express Lanes are transmitting compliance pattern in order to capture Tx die and package crosstalk effects. When measuring Tx jitter it is required for the DUT to drive as many of its outputs as would occur during normal operation in a system environment. When measuring jitter, the preset yielding the lowest jitter value should be selected.

8.3.5.1 *Post Processing Steps to Extract Jitter*

Measured Tx jitter is referenced to the Tx pin, and depending on what type of jitter is being measured and what reference clock architecture is being tested, is subsequently referenced to a recovered data clock, an embedded reference clock captured simultaneously with the data, or to a data edge. Data captured at TP1 requires post processing in order to remove the effects of the breakout channel and to regenerate a data clock (when an embedded reference clock is not captured simultaneously with the data).

8.3.5.2 *Applying CTLE or De-embedding*

Direct probing at a Transmitter's pins is not generally feasible, so data is instead measured at TP1 of the breakout channel. By means of the replica channel it is possible to determine the loss vs. frequency characteristics of the breakout channel and de-embed this channel, resulting in measurements that are effectively referenced to the DUT's pins. Note that since de-embedding amplifies HF noise there is a practical frequency cutoff limit to de-embedding. As de-embedding amplifies HF channel and measurement noise, an HF cutoff limit of 8GHz-12GHz and 20 GHz (3dB point) must be applied to de-embedding, depending on data rate as shown in Table 8-5.

Table 8-5: Recommended De-embedding Cutoff Frequency

Data Rate	HF Cutoff limit for de-embedding
8GT/s	8GHz-12 GHz
16GT/s	20GHz

Jitter is decomposed into data dependent and uncorrelated terms. This separation process effectively separates the jitter caused by package effects from that caused by signal integrity effects. As a result the uncorrelated jitter terms define jitter as it would appear at the die pad.

As an alternative to de-embedding at 16 GT/s the -12 dB CTLE in the reference equalizer can be applied to the data measured at TP1 for measuring all uncorrelated jitter parameters (not DDJ).

If both de-embedding and CTLE approaches are used and given different answers – only the lower values for the uncorrelated jitter parameters are used.

8.3.5.3 *Independent Refclk Measurement and Post Processing*

A Transmitter may operate in the independent Refclk (IR) mode, in which case the Transmitter may not provide a Refclk output. In this case a single-port jitter measurement is required. The post processing algorithm must employ the appropriate model CDR for the reference clock architecture being tested.

8.3.5.4 *Embedded and Non Embedded Refclk Measurement and Post Processing*

Previous versions of the PCIe specification assumed that the transmitting PCIe device was driven from an external source to its Refclk pin. Such an arrangement permits the Tx under test to be driven with a clean Refclk as shown in Figure 8-1.

The specification now explicitly supports the complete matrix of Refclk options, including where the Refclk is embedded, where the reference clock is external, where the reference clock is available at the DUT's pins, and where the reference clock is not available at the DUT's pins. Table 8-6 lists the post processing requirements for each of the four possible combinations. Embedded Refclk with Refclk available at the DUT's pin represents a special case where any jitter common to both the Refclk and the data must be removed via a two-port measurement.

If the DUT supports multiple Refclk modes, as described in Table 8-6 then the Tx needs to be tested in each of the Refclk modes it supports.

Table 8-6: Tx Measurement and Post Processing For Different Refclks

	Embedded Refclk	**Non-Embedded Refclk**
Refclk available at DUT pin and not testing SRIS mode	2-port measurement, CC1st order CDR, PLL1 and 10 ns transport delay2	1-port measurement, CC1st order CDR, clean external Refclk
Refclk not available at DUT pin or testing SRIS mode	1-port measurement, SRIS CDR	1-port measurement, CC1st order CDR, clean external Refclk

Notes:

1. PLL characteristics are defined in Refclk section for each data rate.
2. Refer to Section 8.6.6 for a discussion of the transport delay

8.3.5.5 *Behavioral CDR Characteristics*

A behavioral CDR filter is applied to reject low frequency jitter that would normally be tracked by the CDR in a Receiver. As such, the behavioral CDR represents a bounding function for actual CDR implementations. Rolloff characteristics of the behavioral CDR are dependent on whether the corresponding DUT supports an embedded vs. non-embedded Refclk, is operating in CC or IR mode, and whether the Refclk pin is available for probing (see Table 8-6). In all cases the behavioral CDR represents a highpass filter function where the corner frequency depends on the Tx data rate. Figure 8-12 shows the CC first-order CDR transfer functions for an f_{3dB} of 1.5 MHz, 5.0 MHz, and 10 MHz that corresponds to 2.5 GT/s, 5.0 GT/s, and 8.0 GT/s/16.0 GT/s, respectively.

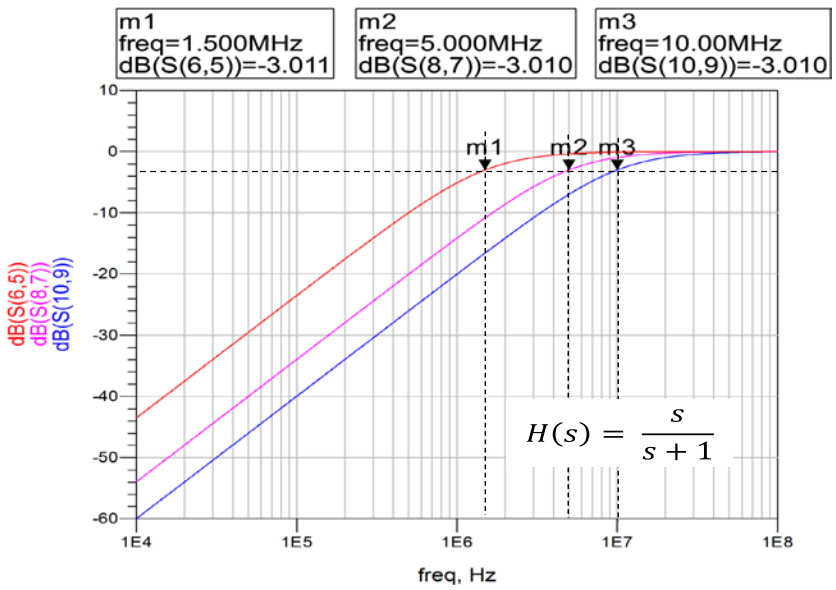


Figure 8-12: First Order CC Behavioral CDR Transfer Functions

Figure 8-13 illustrates second order CDR transfer functions corresponding to 2.5 GT/s and 5.0 GT/s. These functions are defined by a ζ of 0.707 and an f_{3dB} of 1.5 MHz and 5.0 MHz, respectively. Behavioral transfer functions for 8.0 GT/s and 16.0 GT/s approximate the piecewise linear sinusoidal jitter (Sj) masks shown in Section 8.4.2.2.1. SRIS capable Transmitters must be evaluated using these behavioral transfer functions.

Note: The common clock (CC) and independent reference clock (IR) architectures are not interoperable – although it is possible to design a single Receiver that meets both sets of electrical requirements.

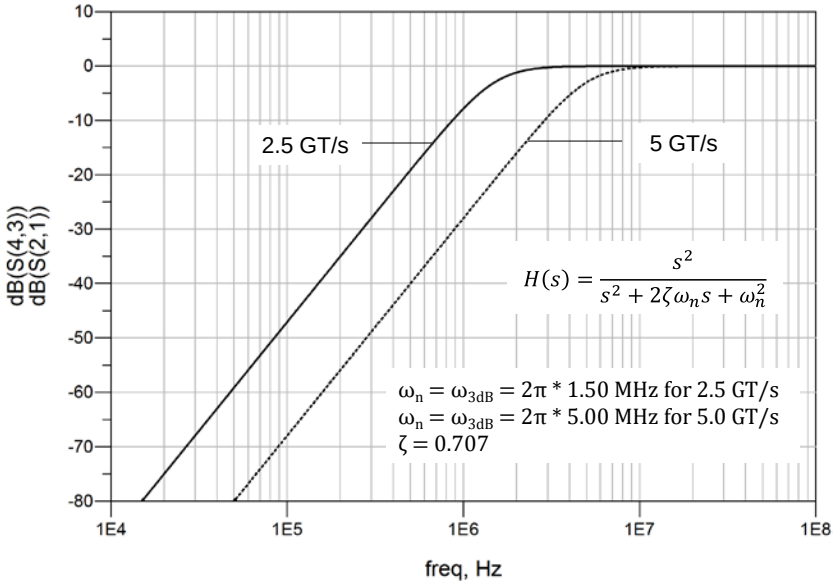


Figure 8-13: 2nd Order Behavioral SRIS CDR Transfer Functions for 2.5 GT/s and 5.0 GT/s

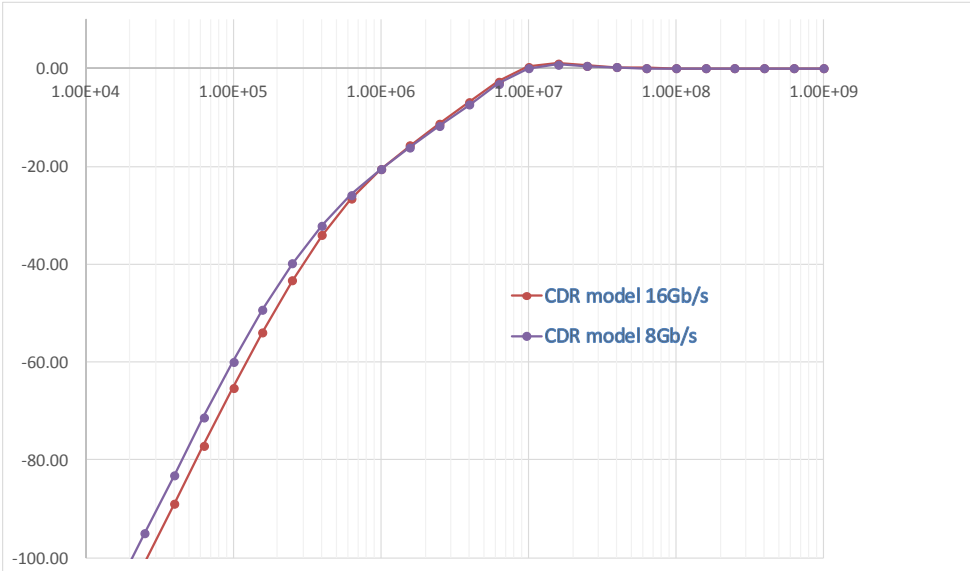


Figure 8-14: Behavioral SRIS CDR Functions for 8.0 and 16.0 GT/s

$$\boldsymbol{H(s) = \frac{s^2}{s^2 + sA + B} \times \frac{s^2 + 2\zeta_2\omega_0 s + \omega_0^2}{s^2 + 2\zeta_1\omega_0 s + \omega_0^2} \times \frac{s}{s + \omega_1}}$$

$$\zeta_1 = \frac{1}{\sqrt{2}};$$

$$\zeta_2 = 1;$$

$$\omega_0 = 10^7 \times 2\pi;$$

$$\omega_1 = 4 \times 10^5 \times 2\pi$$

Equation 8-3. SRIS Behavioral CDR at 8.0 and 16.0 GT/s

$$\boldsymbol{A = 10^7 \times 2\pi;}$$

$$\boldsymbol{B = 2.2 \times 10^{12} \times (2\pi)^2}$$

Equation 8-4. SRIS Behavioral CDR Parameters at 8.0 GT/s

$$\mathbf{A = 9.5 \times 10^{6} \times 2\pi};$$
$$\mathbf{B = 4.36 \times 10^{12} \times (2\pi)^{2}}$$

Equation 8-5. SRIS Behavioral CDR Parameters at 16.0 GT/s

8.3.5.6 *Data Dependent and Uncorrelated Jitter*

Measured at TP1 and de-embedded back to the pin, a Transmitter's jitter contains both data dependent and uncorrelated components. The data dependent components occur principally due to package loss and reflection. Uncorrelated jitter sources include PLL jitter, power supply noise, and crosstalk. The specification separates jitter into uncorrelated and data dependent bins, because such a separation matches well with the Tx and Rx equalization capabilities. Uncorrelated jitter is not mitigated by Tx or Rx equalization and represents timing margin that cannot be recovered via equalization. It is important that margin recoverable by means of equalization (data dependent) is not budgeted as non-recoverable jitter.

Once data dependent jitter has been removed from the Tx measurement it becomes possible to resolve the remaining jitter into Tj and deterministic jitter (Dual Dirac Model) (DJDD) components. High frequency jitter (which is subject to jitter amplification in the channel) is accounted for by separate $T_{TX\text{-}UPW\text{-}DJDD}$ and $T_{TX\text{-}UPW\text{-}TJ}$ parameters.

8.3.5.7 *Data Dependent Jitter*

While DDJ is not explicitly defined as a parameter in the specification, it is necessary to separate DDJ in order to eliminate package loss effects and reference the jitter parameters of interest to the Tx die pad. –Separation of jitter into data dependent and uncorrelated components may be achieved by averaging techniques; for example, by having the Tx repeatedly drive the compliance test pattern which is a repeating pattern.

Figure 8-15 illustrates the relation between Tx data, recovered clock, and the data's PDF. Data dependent jitter is defined as the time delta between the PDF's mean for each zero crossing point and the corresponding recovered clock edge. A sufficient number of repeated patterns must be accumulated to yield stable mean values and PDF profiles for each transition. These PDFs are then utilized to extract uncorrelated jitter parameters.

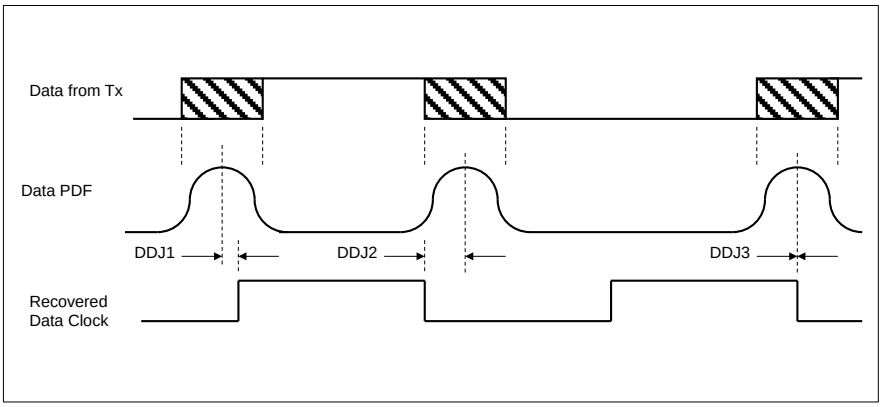


Figure 8-15: Relation Between Data Edge PDFs and Recovered Data Clock

8.3.5.8 *Uncorrelated Total Jitter and Deterministic Jitter (Dual Dirac Model) ($T_{TX\text{-}UTJ}$ and $T_{TX\text{-}UDJDD}$)*

Uncorrelated Total Jitter (UTJ) and uncorrelated deterministic jitter (Dual Dirac model) (UDJDD) are referenced to a recovered data clock generated by means of a CDR tracking function. Uncorrelated jitter may be derived after removing the DDJ component from each PDF and combining the PDFs for all edges in the pattern. By appropriately converting the PDF to a Q-scale it is possible to obtain the graphical relation shown in Figure 8-16, from which $T_{TX\text{-}UTJ}$ and $T_{TX\text{-}UDJDD}$ may be derived. In Figure 8-16 note that the two PDF curves are identical but that the fitted slopes, defined by $1/RJ_{LH}$ and $1/RJ_{RH}$, may differ.

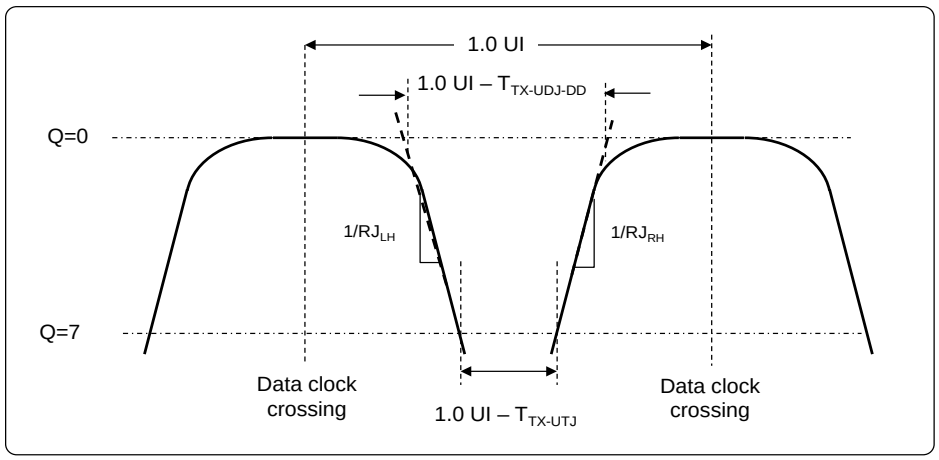


Figure 8-16: Derivation of $T_{TX\text{-}UTJ}$ and $T_{TX\text{-}UDJDD}$

8.3.5.9 *Random Jitter ($T_{TX\text{-}RJ}$) (informative)*

Random jitter is uncorrelated with respect to data dependent jitter. $T_{TX\text{-}RJ}$ may be obtained by subtracting $T_{TX\text{-}UDJ\text{-}DD}$ from $T_{TX\text{-}UTJ}$ and is included in the specification as an informative parameter only. It is typically used as a benchmark to characterize PLL performance.

8.3.5.10 *Uncorrelated Total and Deterministic PWJ ($T_{TX\text{-}UPW\text{-}TJ}$ and $T_{TX\text{-}UPW\text{-}DJDD}$)*

Pulse width jitter is defined as an edge to edge phenomenon on consecutive edges nominally 1.0 UI apart. Figure 8-17 illustrates how PWJ is defined, showing that it is typically present on both data edges of consecutive UI. To accurately quantify PWJ it is first necessary to remove the ISI contributions to PWJ. The shaded areas on either side of the unjittered edges represent the maximum amount of jitter about that edge. Note the jitter for one edge is assumed to be independent from the other.

An equivalent description of PWJ may be obtained by referencing to a fixed leading edge and having jitter contributions from both edges appear at the trailing edge. This approach yields a single PDF as shown below. Each 1 UI wide pulse in the pattern will have a different median for this PDF which is caused by ISI and F/2 jitter. The average of the medians for 1 UI wide pulses at odd and even UI numbers within the pattern are calculated, and the odd and even PDF's are normalized to the appropriate average of medians and summed to form an odd UI PDF and an even UI PDF. The final PDF is calculated from the sum of the summed odd and even UI PDFs. The key idea here is that the final PDF for uncorrelated PWJ should include F/2 or odd/even UI jitter

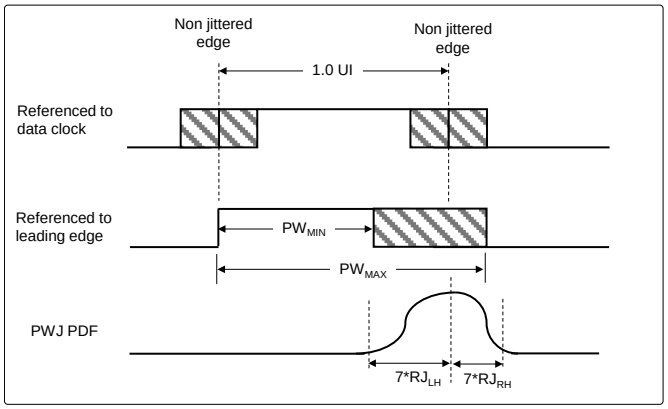


Figure 8-17: PWJ Relative to Consecutive Edges 1 UI Apart

The PDF of jitter around each non-jittered edge may be converted into the Q-scale (see Figure 8-18) from which $T_{TX\text{-}UPW\text{-}TJ}$ and $T_{TX\text{-}UPW\text{-}DJDD}$ may be derived in a manner analogous to $T_{TX\text{-}UTJ}$ and $T_{TX\text{-}UDJDD}$. Note that the PDF may not be symmetric, and the tail of interest is RJ_{LH}, since it represents pulse compression.

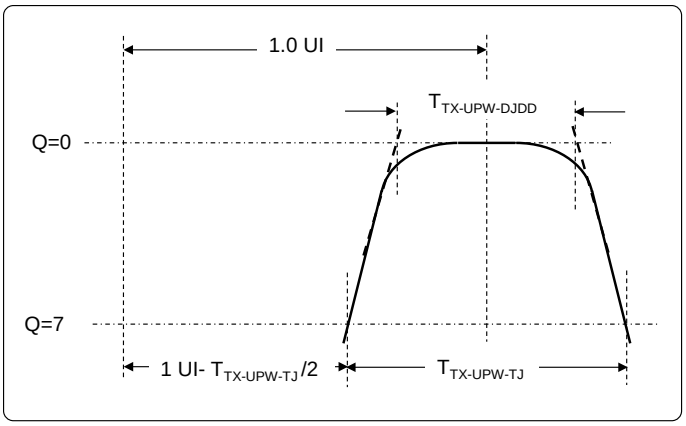


Figure 8-18: Definition of $T_{TX\text{-}UPW\text{-}DJDD}$ and $T_{TX\text{-}UPW\text{-}TJ}$ Data Rate Dependent Transmitter Parameters

8.3.6 Data Rate Dependent Parameters

Note: the jitter margins for 2.5 GT/s and 5.0 GT/s were previously defined at the device's pins. For 2.5 GT/s the jitter was defined via a single parameter that lumped DDJ, UDJDD, UTJ, PWJ-DDJ and PWJ-TJ into a single quantity. Consequently, it is necessary to first remove the DDj jitter component. Since there was no previous UDj-UTj separation $T_{TX\text{-}UTJ}$ and $T_{TX\text{-}UDJDD}$ are set equal to each other. Similarly, there was no UTj-PWj separation, so it is necessary to assume that the entirety of the uncorrelated jitter is PWJthat occurs oppositely on consecutive edges of a 1 UI wide pulse.

For 5.0 GT/s a similar removal of DDj must be performed to obtain UTj. However, the *PCI Express Base Specification, Revision 3.0* for 5.0 GT/s did specify Rj, so a distinct value for $T_{TX\text{-}UDJDD}$ can be obtained. Similarly, the *PCI Express Base Specification, Revision 3.0* for 5.0 GT/s defined a minimum pulse width, assumed to be 100% Dj, from which $T_{TX\text{-}UPW\text{-}TJ}$ and $T_{TX\text{-}UPWDJDD}$ may be derived.

Table 8-7: Data Rate Dependent Transmitter Parameters

Symbol	Parameter description	2.5 GT/s	5.0 GT/s	8.0 GT/s	16.0 GT/s	Units	Notes
UI	Unit Interval	399.88 (min) 400.12 (max)	199.94 (min) 200.06 (max)	124.9625 (min) 125.0375 (max)	62.48125 (min) 62.51875 (max)	ps	The specified UI is equivalent to a tolerance of ±300 ppm. Does not include SSC variations
$BW_{TX\text{-}PKG\text{-}PLL1}$	Tx PLL bandwidth corresponding to $PKG_{TX\text{-}PLL1}$	1.5 (min), 22.0 (max)	8.0 (min), 16.0 (max)	2.0 (min), 4.0 (max),	2.0 (min), 4.0 (max)	MHz	Second order PLL jitter transfer bounding function. Notes 1, 2.
$BW_{TX\text{-}PKG\text{-}PLL2}$	Tx PLL bandwidth corresponding to $PKG_{TX\text{-}PLL2}$	N/A	5.0 (min), 16.0 (max)	2.0 (min) 5.0 (max)	2.0 (min) 5.0 (max)	MHz	2.5 GT/s specifies only one combination of PLL BW and jitter. Notes 1, 2.
$PKG_{TX\text{-}PLL1}$	Tx PLL peaking corresponding to $BW_{TX\text{-}PKG\text{-}PLL1}$	3.0 (max)	3.0 (max)	2.0 (max)	2.0 (max)	dB	Second order PLL jitter transfer bounding function. Notes 1, 2.
$PKG_{\text{-}TX\text{-}PLL2}$	Tx PLL peaking corresponding to $BW_{TX\text{-}PKG\text{-}PLL2}$	N/A	1.0 (min)	1.0 (max)	1.0 (max)	dB	2.5 GT/s specifies only one combination of PLL BW and jitter. Notes 1, 2.
$V_{TX\text{-}DIFF\text{-}PP}$	Differential peak-peak Tx voltage swing for full swing operation	800-1200	800-1200	800-1300	800-1300	mVPP	As measured with compliance test load. Defined as 2*|VTXD+ - VbTXD-|. Note 3.
$V_{TX\text{-}DIFF\text{-}PP\text{-}LOW}$	Differential peak-peak Tx voltage swing for low swing operation	400 (min)- 1200 (max)	400-1200	400 (min)- 1300(max)	400 (min)- 1300 (max)	mVPP	As measured with compliance test load. Defined as 2*|VTXD+ - VbTXD-|. Note 3.
$V_{TX\text{-}EIEOS\text{-}FS}$	Minimum voltage swing during EIEOS for full swing signaling	N/A	N/A	250 (min)	250 (min)	mVPP	Note 4
$V_{TX\text{-}EIEOS\text{-}RS}$	Minimum voltage swing during EIEOS for reduces swing signaling	N/A	N/A	232 (min)	232 (min)	mVPP	Note 4
$ps21_{TX\text{-}ROOT\text{-}DEVICE}$	Pseudo package loss of a device containing root ports	N/A	N/A	3.0 (max)	5.0 (max)	dB	Note 5.
$ps21_{TX\text{-}NON\text{-}ROOT\text{-}DEVICE}$	Pseudo package loss for all devices not containing root ports	N/A	N/A	3.0 (max)	3.0 (max)	dB	Note 5.
$V_{TX\text{-}BOOST\text{-}FS}$	Maximum nominal Tx boost ratio for full swing	N/A	N/A	8.0	8.0 (min)	dB	Nominal boost beyond 8.0 dB is limited to guarantee that ps21TX limits are satisfied
$V_{TX\text{-}BOOST\text{-}RS}$	Maximum nominal Tx boost ratio for reduced swing	N/A	N/A	2.5	~2.5 (min)	dB	Assumes ±1.0 dB tolerance from diagonal elements in Table 8-3.

Symbol	Parameter description	2.5 GT/s	5.0 GT/s	8.0 GT/s	16.0 GT/s	Units	Notes
$EQ_{TX-COEFF-RES}$	Tx coefficient resolution	N/A	N/A	1/63 (min) 1/24 (max)	1/63 (min) 1/24 (max)	N/A	
$V_{TX-DE-RATIO-3.5dB}$	Tx de-emphasis ratio for 2.5 and 5.0 GT/s	2.5 (min) 4.5 (max)	2.5 (min) 4.5 (max)	N/A	N/A	dB	
$V_{TX-DE-RATIO-6dB}$	Tx de-emphasis ratio for 5.0 GT/s	N/A	4.5 (min) 7.5 (max)	N/A	N/A	dB	
T_{TX-UTJ}	Tx uncorrelated total jitter	100 (max)	50 (max)	31.25 (max)	12.5 (max)	ps PP at 10-12	See Sec 8.3.5.8 for details.
$T_{TX-UTJ-SRIS}$	Tx uncorrelated total jitter when testing for the IR clock mode with SSC	100 (max)	66.51 (max)	33.83 (max)	15.85 (max)	ps PP at 10-12	See Sec 8.3.5.8 for details.
$T_{TX-UDJDD}$	Tx uncorrelated Dj for non-embedded Refclk	100 (max)	30 (max)	12 (max)	6.25 (max)	ps PP	See Sec 8.3.5.8 for details.
$T_{TX-UPW-TJ}$	Total uncorrelated pulse width jitter	N/A	40 (max)	24 (max)	12.5 (max)	ps PP at 10-12	See Sec 8.3.5.9 for details
$T_{TX-UPWDJDD}$	Deterministic DjDD uncorrelated pulse width jitter	N/A	40 (max)	10 (max)	5 (max)	ps PP	See Sec 8.3.5.9 for details
T_{TX-RJ}	Tx Random jitter	N/A	1.4 – 3.6	1.4 – 2.2	0.45 – 0.89	ps RMS	Informative parameter only. Range of Rj possible with zero to maximum allowed $T_{TX-UDJDD}$.
$L_{TX-SKEW}$	Lane-to-Lane Output Skew	2.5 (max)	2.0 (max)	1.5 (max)	1.25 (max)	ns	Between any two Lanes within a single Transmitter.
$RL_{TX-DIFF}$	Tx package plus die differential return loss	See Figure 8-19	See Figure 8-19	See Figure 8-19	See Figure 8-19	dB	Note 6
RL_{TX-CM}	Tx package plus die common mode return loss	See Figure 8-20	See Figure 8-20	See Figure 8-20	See Figure 8-20	dB	Note 6

Notes:

1. A single combination of PLL BW and peaking is specified for 2.5 GT/s implementations. For other data rates, two combinations of PLL BW and peaking are specified to permit designers to make a tradeoff between the two parameters.
2. The Tx PLL Bandwidth must lie between the min and max ranges given in the above table. PLL peaking must lie below the value listed above. Note: the PLL B/W extends from zero up to the value(s) specified in the above table. The PLL BW is defined at the point where its transfer function crosses the -3dB point.
3. See Sections 8.3.3.4 and 0 for measurement details. For 8.0 GT/s and 16.0 GT/s no minimum voltage swing is specified because it is captured by $V_{TX-BOOST-FS}$ and $V_{TX-BOOST-RS}$ parameters.
4. $V_{TX-EIEOS-FS}$ and $V_{TX-EIEOS-RS}$ are measured at the device pin and include package loss. Voltage limits comprehend both full swing and reduced swing modes. A Transmitter must advertise a value for LF during TS1 at 8.0 and 16.0 GT/s that ensures that these parameters are met.
5. The numbers above take into account measurement error. For some Tx package/driver combinations $ps21_{TX}$ may be greater than 0 dB. The channel compliance methodology at 2.5 and 5.0 GT/s assumes the 8.0 GT/s package model.
6. The DUT must be powered up and DC isolated, and its data+/data- outputs must be in the low-Z state at a static value.
7. The reference plane for all parameters at 2.5 and 5.0 GT/s is the package pins.

8.3.7 Tx and Rx Return Loss

Return loss measurements for the Tx and Rx are essentially identical, so both are included in the Transmitter section. Return loss measurements are made at the end of the respective breakout channels and require that the breakout channel's contribution to RL be de-embedded, thereby associating the return loss with the Tx or Rx pin. Figure 8-19 defines the pass/fail mask for differential return loss. There is a small difference in the 2.5-8.0 GHz mask value range to account for the slightly lower C_{DIE} of the Receiver as compared to the Transmitter. This difference is also reflected in the Tx and Rx behavioral package models for 8.0 GT/s. Both differential and common mode are defined over a frequency range of 50 MHz to 8.0 GHz.

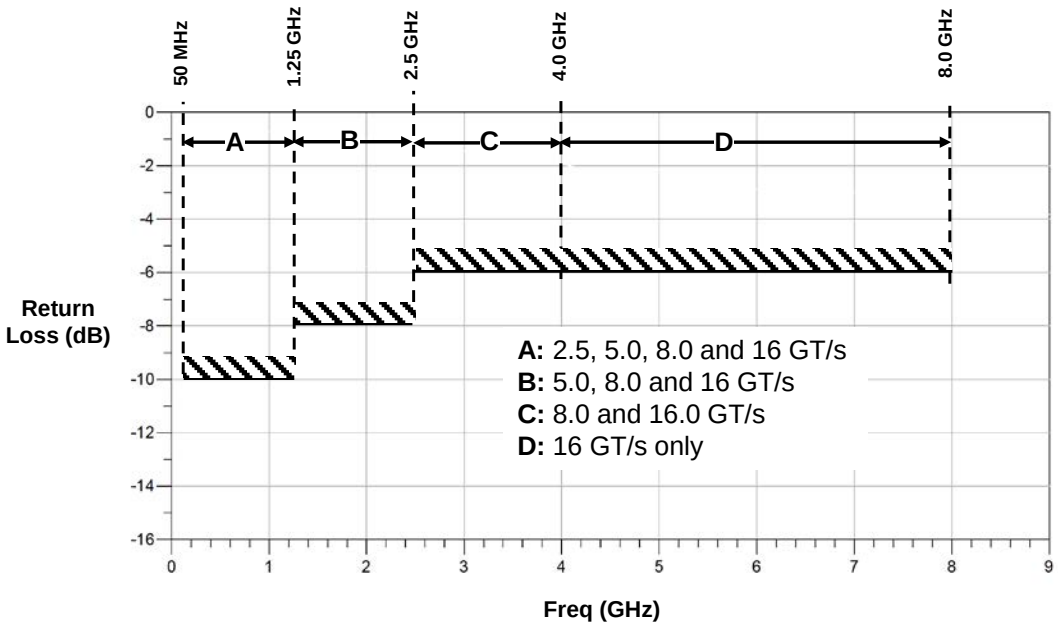


Figure 8-19: Tx, Rx Differential Return Loss Mask

The pass/fail mask for common mode return loss is shown in Figure 8-20. Return loss measurements require that both the Tx and Rx are powered up and that their respective termination circuits are enabled.

Microprobing the package may be required to measure RL accurately.

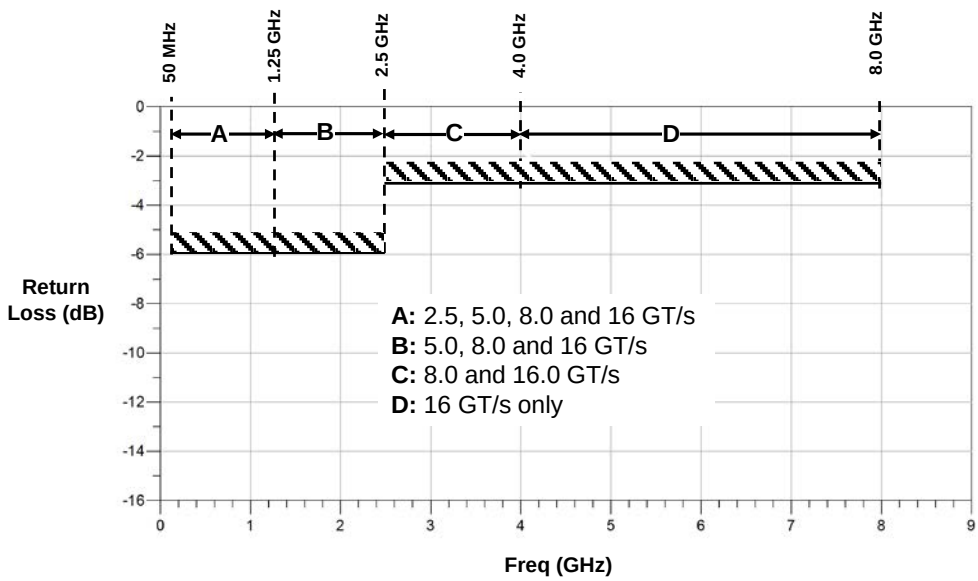


Figure 8-20: Tx, Rx Common Mode Return Loss Mask

8.3.8 Transmitter PLL Bandwidth and Peaking

8.3.8.1 *2.5 GT/s and 5.0 GT/s Tx PLL Bandwidth and Peaking*

PLL bandwidth and peaking are defined for both the Transmitter and Receiver in order to place an upper limit on the amount of Refclk jitter that is propagated to the transmitted data and to the CDR. Defining PLL BW and peaking limits also guarantees a minimum degree of Tx/Rx jitter tracking in those systems utilizing a common Refclk Rx architecture.

The 2.5 GT/s PLL characteristics have been moved from the 3.0 CEM spec to the Electrical Base Spec. A single PLL bandwidth range from 1.5 to 22 MHz is given, which is identical to that defined in the CEM spec. No range of peaking was given in the CEM spec for the 2.5 GT/s PLL. However, for the Electrical Base Spec a peaking range of 0.01 dB to 3 dB is now defined. It is necessary to place a non-zero lower limit on the peaking, both to define a corner case as well as to maintain a common mathematical expression for the PLL transfer function in terms of ω_n and ζ.

Two sets of bandwidth and peaking are defined for 5.0 GT/s: 8-16 MHz with 3 dB of peaking and 5.0-16.0 MHz with 1 dB of peaking. This gives the designer the option of trading off between a low peaking PLL design vs. a low bandwidth design. For 2.5 GT/s, a single PLL bandwidth and peaking range is specified at 1.5-22 MHz with 0.01 to 3.0 dB of peaking.

8.3.8.2 *8.0 GT/s and 16.0 GT/s Tx PLL Bandwidth and Peaking*

The Tx and Rx PLL bandwidth for 8.0 and 16.0 GT/s is 2-5 MHz with 1.0 dB of peaking or 2-4 MHz with 2.0 dB of peaking. The 8.0 GT/s PLL BW range is substantially lower than the PLL bandwidths specified for 5.0 GT/s or 2.5 GT/s to reduce the amount of Refclk jitter at the sample latch of the Receiver. A non-zero value of 0.01 dB is given for the lower limit of the peaking to define all the peaking corners.

8.3.8.3 *Series Capacitors*

PCI Express requires series capacitors to provide a DC block between Tx and Rx. The min/max capacitance spread has been decreased from that of the 2.5 and 5.0 GT/s standards, while the maximum value has been slightly increased. This change is necessary to minimize DC wander effects due to data scrambling implemented at 8.0 GT/s and 16.0 GT/s. Note that 2.5 GT/s and 5.0 GT/s signaling must also propagate through these larger value capacitors, but the small increase in capacitor size has no adverse impact on either 2.5 GT/s or 5.0 GT/signaling or low frequency in-band signaling such as Receiver detect.

8.3.9 Data Rate Independent Tx Parameters

Table 8-8: Data Rate Independent Tx Parameters

Symbol	Parameter Description	Value	Units	Notes
$V_{TX\text{-}AC\text{-}CM\text{-}PP}$	Tx AC peak-peak common mode voltage	150 (max)	mVPP	Over the 0.03-500 MHz range: no more than 100 mVPP at 5.0 GT/s, and no more than 50 mVPP at 8.0 or 16.0 GT/s. Note 1.
$V_{TX\text{-}DC\text{-}CM}$	Tx DC peak-peak common mode voltage	0 (min) 3.6 (max)	V	Total single-ended voltage a Tx can supply under any conditions with respect to ground. See also the $I_{TX\text{-}SHORT}$. See Note 2
$V_{TX\text{-}CM\text{-}DC\text{-}ACTIVE\text{-}IDLE\text{-}DELTA}$	Absolute delta of DC Common Mode Voltage during L0 and Electrical Idle	0 (min) 100 (max)	mV	$\lvert V_{TX\text{-}CM\text{-}DC}$ [during L0] – $V_{TX\text{-}CM\text{-}Idle\text{-}DC}$ [during Electrical Idle]$\rvert$ <= 100 mV $V_{TX\text{-}CM\text{-}DC} = DC_{(avg)}$ of $\lvert V_{TX\text{-}D+} + V_{TX\text{-}D-}\rvert/2$ $V_{TX\text{-}CM\text{-}Idle\text{-}DC} = DC_{(avg)}$ of $\lvert V_{TX\text{-}D+} + V_{TX\text{-}D-}\rvert/2$ [Electrical Idle]
$V_{TX\text{-}CM\text{-}DC\text{-}LINE\text{-}DELTA}$	Absolute Delta of DC Common Mode Voltage between D+ and D-	0 (min) 25 (max)	mV	$\lvert V_{TX\text{-}CM\text{-}DC\text{-}D+}$ [during L0] – $V_{TX\text{-}CM\text{-}DC\text{-}D-}$ [during L0.]$\rvert$ ≤ 25 mV $V_{TX\text{-}CM\text{-}DC\text{-}D+} = DC_{(avg)}$ of $\lvert V_{TX\text{-}D+}\rvert$ [during L0] $V_{TX\text{-}CM\text{-}DC\text{-}D-} = DC_{(avg)}$ of $\lvert V_{TX\text{-}D-}\rvert$ [during L0]
$V_{TX\text{-}IDLE\text{-}DIFF\text{-}AC\text{-}p}$	Electrical Idle Differential Peak Output Voltage	0 (min) 20 (max)	mV	$V_{TX\text{-}IDLE\text{-}DIFF\text{-}AC\text{-}p} = \lvert V_{TX\text{-}Idle\text{-}D+} - V_{TX\text{-}Idle\text{-}D-}\rvert$ ≤ 20 mV. Voltage must be band pass filtered to remove any DC component and HF noise. The bandpass is constructed from two first-order filters, the high pass and low pass 3 dB bandwidths are 10 kHz and 1.25 GHz, respectively.
$V_{TX\text{-}IDLE\text{-}DIFF\text{-}DC}$	DC Electrical Idle Differential Output Voltage	0 (min) 5 (max)	mV	$V_{TX\text{-}IDLE\text{-}DIFF\text{-}DC} = \lvert V_{TX\text{-}Idle\text{-}D+} - V_{TX\text{-}Idle\text{-}D-}\rvert$ ≤ 5 mV. Voltage must be low pass filtered to remove any AC component. The low pass filter is first-order with a 3 dB bandwidth of 10 kHz.
$V_{TX\text{-}RCV\text{-}DETECT}$	The amount of voltage change allowed during Receiver detection	600 (max)	mV	The total amount of voltage change in a positive direction that a Transmitter can apply to sense whether a low impedance Receiver is present. Note: Receivers display substantially different impedance for V_{IN} <0 vs V_{IN} > 0.

Symbol	Parameter Description	Value	Units	Notes
$T_{TX\text{-}IDLE\text{-}MIN}$	Minimum time spent in Electrical Idle	20 (min)	ns	The time a Tx must spend in Electrical Idle before transitioning to another state
$T_{TX\text{-}IDLE\text{-}SET\text{-}TO\text{-}IDLE}$	Maximum time to transition to a valid Electrical Idle after sending an EIOS	8 (max)	ns	After sending the required number of EIOSs, the Transmitter must meet all Electrical Idle specifications within this time. This is measured from the end of the last UI of the last EIOS to the Transmitter in Electrical Idle.
$T_{TX\text{-}IDLE\text{-}TO\text{-}DIFF\text{-}DATA}$	Maximum time to transition to valid diff signaling after leaving Electrical Idle	8 (max)	ns	Maximum time to transition to valid diff signaling after leaving Electrical Idle. This is considered a debounce time to the Tx.
$T_{CROSSLINK}$	Crosslink random timeout	1.0 (max)	ms	This random timeout helps resolve potential conflicts in the crosslink configuration.
C_{TX}	AC Coupling Capacitor	176 (min) 265 (max)	nF	All Transmitters shall be AC coupled. The AC coupling is required either within the media or within the transmitting component itself.
$Z_{TX\text{-}DIFF\text{-}DC}$	DC differential Tx impedance	120 (max)	Ω	Low impedance defined during signaling. The minimum value is bounded by $RL_{TX\text{-}DIFF}$.
$I_{TX\text{-}SHORT}$	Tx short circuit current	90 (max)	ma	Tx short circuit current. Note 2

Notes:

1. $V_{TX\text{-}AC\text{-}CM\text{-}PP}$ is measured at TP1 without de-embedding the breakout channel. This parameter captures device CM only and is not intended to capture system CM noise. For each data rate an LPF with a -3 dB point of data rate/2 is applied.
2. $I_{TX\text{-}SHORT}$ and $V_{TX\text{-}DC\text{-}CM}$ stipulate the maximum current/voltage levels that a Transmitter can generate and therefore define the worst case transients that a Receiver must tolerate.

8.4 Receiver Specifications

8.4.1 Receiver Stressed Eye Specification

All Receiver speeds: 2.5, 5.0, 8.0, and 16.0 GT/s are tested by means of a stressed eye applied over a calibration channel that approximates the near worst-case loss characteristics encountered in an actual channel. The recovered eye is defined at the input to the Receiver's latch. For 2.5 GT/s and 5.0 GT/s this point is equivalent to the Rx pins; for 8.0 GT/s and 16.0 GT/s it is equivalent to the signal at the Rx die pad after behavioral Rx equalization has been applied.

8.4.1.1 *Breakout and Replica Channels*

The closest practical measurement points to the Rx DUT are the coaxial connectors at the end of a breakout channel, while the Rx reference point of interest is the pin of the Rx. By constructing a replica channel that closely matches the electrical characteristics of the breakout channel it is possible to measure the signal as it would appear at the DUT's pin, if the DUT were an ideal termination. Impedance targets for the interconnect environment are 100Ω differential and 50Ω single-ended, and the impedance tolerance should be maintained within ±5% or better

In Figure 8-21 the stressed eye is observed at TP2 with the signal sources connected to the calibration channel. A calibration channel will be required for each data rate. Once the stressed eye has been calibrated, the signal source is applied to the DUT. Note that TP1-TP2 encompasses all the components between the signal source and the equivalent of the DUT pin, thereby capturing all non-ideal characteristics in the overall insertion loss due to cabling and replica/breakout channel, excluding Rx package. The AC and DC loss from generator to TP1 are assumed to be zero or must be otherwise de-embedded.

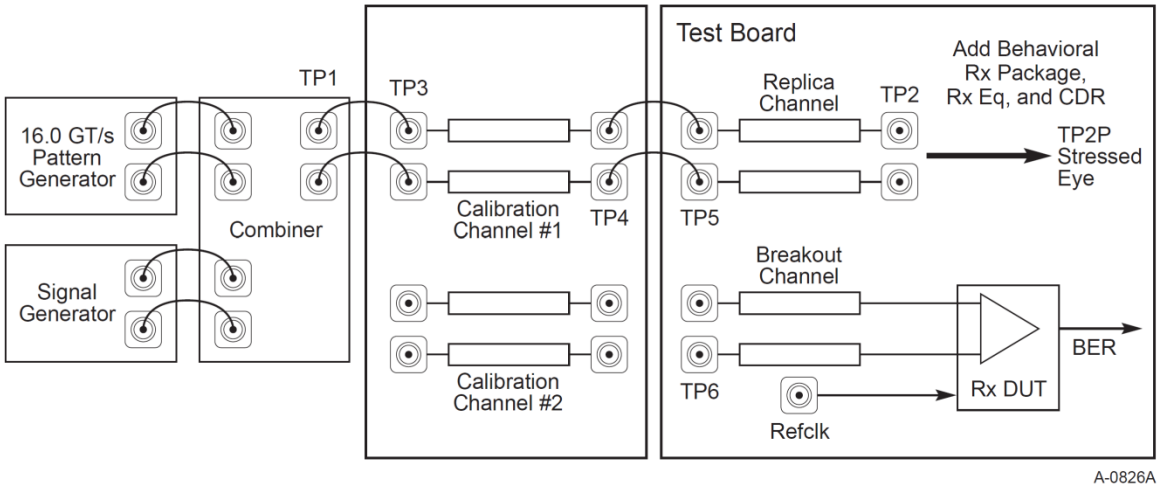


Figure 8-21: Rx Testboard Topology for 16.0 GT/s

8.4.1.2 *Calibration Channel Insertion Loss Characteristics*

Four calibration channels, each with a specified differential insertion loss at one of the four PCIe data rates, provide the means of generating prescribed amounts of ISI that approximates a worst case channel. For each data rate a single calibration channel loss mask is defined by means of two pairs of IL limits at a high and a low frequency. Note: It is acceptable to generate IL by means other than physical PCB traces, such as specialized filters. Note: Calibration channel IL needs to comprehend both actual channel loss and the Tx package loss. The Calibration Channel needs to include cabling losses within the IL mask.

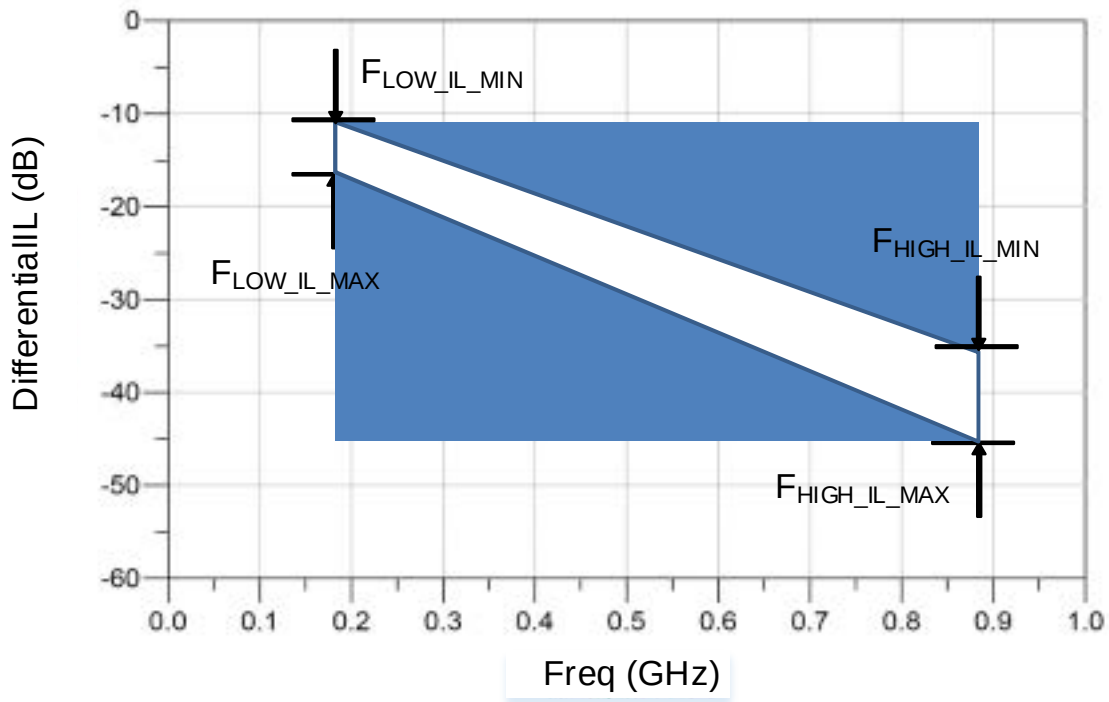


Figure 8-22: Calibration Channel IL Mask Excluding Rx Package

The following table defines the calibration channel IL masks by means of four loss points at two frequency limits, thereby creating a quadrilateral shaped solution area.

Table 8-9: Calibration Channel IL Limits

Data Rate	$F_{LOW-IL-MIN}$	$F_{LOW-IL-MAX}$	$F_{HIGH-IL-MIN}$	$F_{HIGH-IL-MAX}$
2.5 GT/s	4.5 dB @ 1 GHz	5.0 dB @ 1 GHz	4.7 dB @ 1.25 GHz	5.2 dB @ 1.25 GHz
5.0 GT/s	4.5 dB @ 1 GHz	5.0 dB @ 1 GHz	10.0 dB @2.5 GHz	11.0 dB @2.5 GHz
8.0 GT/s	5 dB @ 1 GHz	8 dB @ 1 GHz	~20 dB @ 4 GHz	~22 dB @ 4 GHz
16.0 GT/s Root Port Long	4.2 dB @ 1 GHz	5.2 dB @ 1 GHz	~22.5 dB @ 8 GHz	~23.5 dB @ 8 GHz
16.0 GT/s Non-Root Port Long	4.2 dB @ 1 GHz	5.2 dB @ 1 GHz	~24.5 dB @ 8 GHz	~25.5 dB @ 8 GHz

Note: Calibration channel plus Rx package is 28 dB nominally (informative) for 16.0 GT/s.

Note: Different reference packages are defined for devcies containing Root Ports and all other device types at 16.0 GT/s.

Note: It is recommeded that some validation be done with shorter channels at 16.0 GT/s.

The impedance targets for the Rx tolerancing interconnect environment are 100 Ω differential and 50 Ω single-ended for the 2.5, 5.0, and 8.0 GT/s channels and 85 Ω differential and 42.5 Ω single-ended for the 16.0 GT/s channels; the impedance tolerance should be maintained within ±5% or

better. The calibration channel for 16.0 GT/s must meet the following return loss mask when measured from either end of the calibration channel:.

- ≤ -12 dB for < 4 GHz
- ≤ - 8 dB for ≥ 4 GHz and < 12 GHz
- ≤ -6 dB for ≥ 12 GHz and ≤ 16 GHz

A calibration channel consists of a differential pair of PCB traces terminated at both ends by coaxial connectors. For 16.0 GT/s the calibration channel includes a 4.0 Card Electromechanical Specification compliant connector placed at least 4 dB away from the coaxial connectors where the signal generator is connected. The calibration channel's electrical characteristics are defined in terms of differential insertion loss masks as shown in Figure 8-22, where. S_{DD21} is measured between TP1 and TP2. Connections between TP1-TP3 and TP4-TP5 represent cabling and are included in the S_{DD21} measurement.

While the 8.0 GT/s and 16 GT/s-parameter masks do not extend below 1.0 GHz, all calibration channels must be well behaved below 1.0 GHz and must not have a DC resistance in excess of 7.5 ohms, as measured by the sum of the resistances of the D+ and D- traces. This limitation on DC resistance guarantees that the calibration channel low frequency characteristic is consistent with the extrapolations of the S_{DD21} masks to DC. The calibration loss targets for devices containing Root Ports and other devices are different because the reference package models have different losses.

For 16.0 GT/s the insertion loss range $F_{HIGH\text{-}IL\text{-}MIN}$ to $F_{HIGH\text{-}IL\text{-}MAX}$ is the nominal loss. The calibration channel must have a series of loss options covering a range from at least 2 dB below $F_{HIGH\text{-}IL\text{-}MIN}$ to 3 dB above $F_{HIGH\text{-}IL\text{-}MAX}$ (for example for the non-root case this means a loss range from -22.5 to -28.5 dB) with loss delta between consecutive options of 0.5 dB or less.

IMPLEMENTATION NOTE

16.0 GT/s Calibration Channel Reference Design

This section gives an example of a 16.0 GT/s calibration channel that was built and tested to meet the requirements in this specification. A high level block diagram of the calibration channel is shown in Figure 8-23. Note that this example fixture covers a wider loss range then required by the specification and can cover both root and non-root cases. The test fixture includes four PCBs:

16.0 GT/s Rx Calibration Base Boards

Sixteen differential pairs (85 Ohm Nominal Impedance) routed from SMA connectors to a CEM through-hole connector. There are three different base boards to achieve the following insertion loss ranges - The insertion loss of the differential pairs for the base board is varied as follows @ 8GHz in 0.5 dB steps.

Low-Loss Base Board:	4-11.5 dB
Mid-Loss Base Board:	12-19.5 dB
High-Loss Base Board:	20-27.5 dB

All traces are routed as microstrip on the bottom layer. The SMA connectors and CEM connectors are optimized with layout techniques at 8GHz.

16.0 GT/s Rx Calibration Riser Board

Sixteen differential pairs (85 Ohm Nominal Impedance) routed from SMA connectors to Gold Edge Fingers. The insertion loss of the differential pairs is fixed at 4 dB nominal @ 8GHz for all sixteen pairs. All traces are routed as microstrip on the bottom layer. The SMA connectors and CEM connectors are optimized with layout techniques at 8GHz.

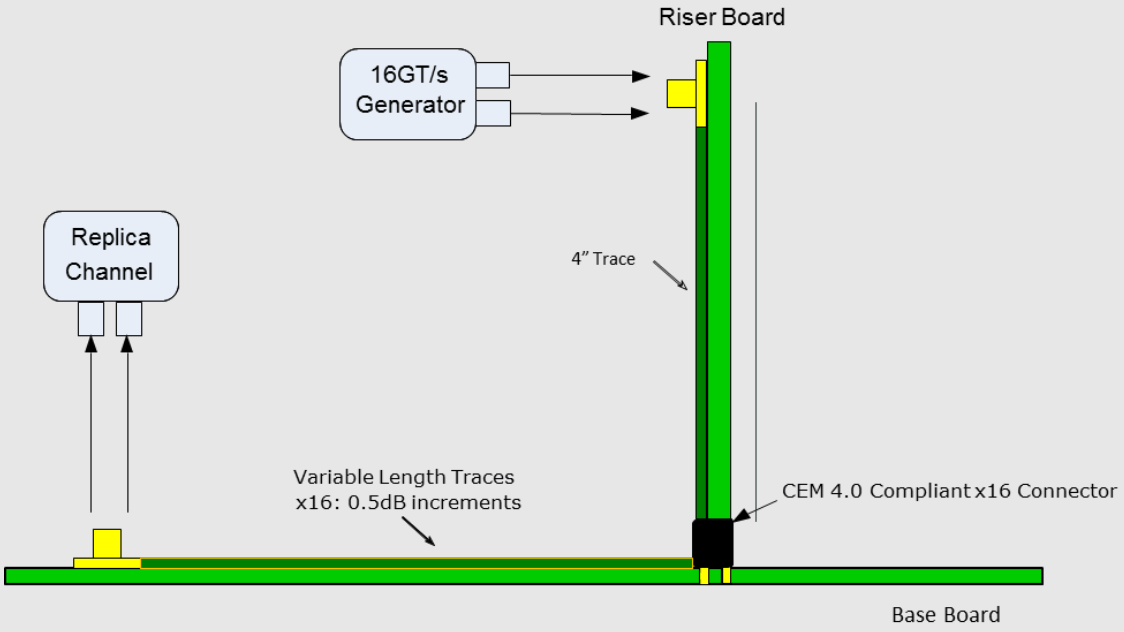


Figure 8-23: Example 16 GT/s Calibration Channel

IMPLEMENTATION NOTE (continued)

The stackup for both boards is shown in Figure 8-24 where 65% is the estimated copper fill percentage. Figure 8-24 includes stackups for both nominal 85 Ω and 100 Ω stackups – the 85 Ω stackup is used for the calibration channel example.

Figure 8-24: Stackup for Example 16 GT/s Calibration Channel

The pad stack for the CEM connector drill holes is shown in Figure 8-25 and the pad stack for the SMA drill holes is shown in Figure 8-26.

Padstack: C40P28P3M3-A59 Type: through Inner pads: Optional

Layer	Pad Type	Geometry	Width	Height	Offset X	Offset Y	Flash Name	Shape Name
TOP	ANTI	CIRCLE	59.00	59.00	0.00	0.00		
TOP	THERMAL	RECTANGLE	80.00	80.00	0.00	0.00	TH80X60X15_4X45	
TOP	REGULAR	CIRCLE	40.00	40.00	0.00	0.00		
GND-2	ANTI	CIRCLE	59.00	59.00	0.00	0.00		
GND-2	THERMAL	RECTANGLE	80.00	80.00	0.00	0.00	TH80X60X15_4X45	
GND-2	REGULAR	CIRCLE	40.00	40.00	0.00	0.00		
GND-3	ANTI	CIRCLE	59.00	59.00	0.00	0.00		
GND-3	THERMAL	RECTANGLE	80.00	80.00	0.00	0.00	TH80X60X15_4X45	
GND-3	REGULAR	CIRCLE	40.00	40.00	0.00	0.00		
BOTTOM	ANTI	CIRCLE	59.00	59.00	0.00	0.00		
BOTTOM	THERMAL	RECTANGLE	80.00	80.00	0.00	0.00	TH80X60X15_4X45	
BOTTOM	REGULAR	CIRCLE	40.00	40.00	0.00	0.00		
internal_pad_def	ANTI	CIRCLE	59.00	59.00	0.00	0.00		
internal_pad_def	THERMAL	RECTANGLE	80.00	80.00	0.00	0.00	TH80X60X15_4X45	
internal_pad_def	REGULAR	CIRCLE	40.00	40.00	0.00	0.00		
SOLDERMASK_TOP	REGULAR	CIRCLE	40.00	40.00	0.00	0.00		
SOLDERMASK_BOTTOM	REGULAR	CIRCLE	40.00	40.00	0.00	0.00		
PASTEMASK_TOP	REGULAR	NULL	0.00	0.00	0.00	0.00		
PASTEMASK_BOTTOM	REGULAR	NULL	0.00	0.00	0.00	0.00		
FILMMASKTOP	REGULAR	NULL	0.00	0.00	0.00	0.00		
FILMMASKBOTTOM	REGULAR	NULL	0.00	0.00	0.00	0.00		

Drill Data for C40P28P3M3-A59

Hole Type	Drill Dia	Plating	Figure	Characters	Width	Height	Offset X	Offset Y	Pos Tolerance	Neg Tolerance	Non-Standard
CIRCLE DRILL	28.00	PLATED	DIAMOND		50.00	50.00	0.00	0.00	3.00	3.00	

Figure 8-25: CEM Connector Drill Hole Pad Stack

IMPLEMENTATION NOTE (continued)

Padstack: C60_BOT75P20_SSM40P3M3 Type: through Inner pads: Optional

Layer	Pad Type	Geometry	Width	Height	Offset X	Offset Y	Flash Name	Shape Name
TOP	ANTI	CIRCLE	45.00	45.00	0.00	0.00		
TOP	THERMAL	RECTANGLE	5.00	5.00	0.00	0.00	5MIL_PAD	
TOP	REGULAR	CIRCLE	60.00	60.00	0.00	0.00		
GND-2	ANTI	CIRCLE	45.00	45.00	0.00	0.00		
GND-2	THERMAL	RECTANGLE	5.00	5.00	0.00	0.00	5MIL_PAD	
GND-2	REGULAR	CIRCLE	60.00	60.00	0.00	0.00		
GND-3	ANTI	CIRCLE	45.00	45.00	0.00	0.00		
GND-3	THERMAL	RECTANGLE	5.00	5.00	0.00	0.00	5MIL_PAD	
GND-3	REGULAR	CIRCLE	60.00	60.00	0.00	0.00		
BOTTOM	ANTI	CIRCLE	45.00	45.00	0.00	0.00		
BOTTOM	THERMAL	RECTANGLE	5.00	5.00	0.00	0.00	5MIL_PAD	
BOTTOM	REGULAR	CIRCLE	75.00	75.00	0.00	0.00		
internal_pad_def	ANTI	CIRCLE	45.00	45.00	0.00	0.00		
internal_pad_def	THERMAL	RECTANGLE	5.00	5.00	0.00	0.00	5MIL_PAD	
internal_pad_def	REGULAR	CIRCLE	60.00	60.00	0.00	0.00		
SOLDERMASK_TOP	REGULAR	CIRCLE	60.00	60.00	0.00	0.00		
SOLDERMASK_BOTTOM	REGULAR	CIRCLE	40.00	40.00	0.00	0.00		
PASTEMASK_TOP	REGULAR	NULL	0.00	0.00	0.00	0.00		
PASTEMASK_BOTTOM	REGULAR	NULL	0.00	0.00	0.00	0.00		
FILMMASKTOP	REGULAR	NULL	0.00	0.00	0.00	0.00		
FILMMASKBOTTOM	REGULAR	NULL	0.00	0.00	0.00	0.00		

Drill Data for C60_BOT75P20_SSM40P3M3

Hole Type	Drill Dia	Plating	Figure	Characters	Width	Height	Offset X	Offset Y	Pos Tolerance	Neg Tolerance	Non-Standard
CIRCLE DRILL	20.00	PLATED	CIRCLE	E	60.00	60.00	0.00	0.00	3.00	3.00	

Figure 8-26: Pad Stack for SMA Drill Holes

8.4.1.3 *Post Processing Procedures*

The Receiver test requires that the stressed eye characteristics be measured at TP2 (which is accessible) and then post-processed to yield a signal as it would appear at test point two post-processed (TP2P) (which is not accessible) for 8.0 and 16.0 GT/s. TP2P defines a reference point that comprehends the effects of the behavioral Rx package plus Rx equalization and represents the only location where a meaningful EH and EW limits can be defined.

8.4.1.4 *Behavioral Rx Package Models*

Behavioral Rx package models are included as part of the post processing to allow the calibrated eye to comprehend package insertion loss. A separate pair of package models is defined for 8.0 and 16.0 GT/s eye calibration. At 8.0 GT/s, separate package models are defined for TX and RX ports to reflect the smaller CPAD capacitance typical in most receiver implementations. At 16.0 GT/s, separate package models are defined for devices containing Root Ports and all other devices. This is necessary to allow a reasonable channel solution space and is based on the assumption that devices containing Root Complexes are usually large and socketed, while all other devices tend to be unsocketed and smaller. The 16.0 GT/s Root and Non-Root behavioral Rx package models have been constructed to represent respective package loss characteristics for high loss, but not worst case loss, packages.

The 8.0 GT/s stressed eye test for all devices and the 16.0 GT/s stressed eye test for Non-Root Package devices that support captive channels are required to use the appropriate behavioral package (see section 8.3.3.11). For all other device types, if the actual Rx package performance is worse than that of the behavioral package, then the actual package models are permitted to be used. If the actual package models are used, the calibration channel must be adjusted such that the total channel loss including the embedded actual package remains at 28 dB nominal. Note that form factor overall requirements still need to be met. The Rx package performance is assessed using the methodology defined in 8.5.1.2.

Details of the behavioral Rx packages are provided in Section 8.5.1.1 of the Channel Tolerancing section. S-parameter models for the behavioral Rx package models are available as design collateral. The reference impedance at the pad side of the packages model is assumed to be 2 x 50 Ω.

8.4.1.5 *Behavioral CDR Model*

Post processing shall include a behavioral CDR model with a data rate dependent transfer function. A first order CDR transfer function is utilized for Receivers operating with a CC Refclk architecture. For Receivers operating in IR Refclk mode an alternate CDR transfer function is required. For a given data rate the behavioral CDR used for Rx testing is the same as the corresponding CDR used for Tx testing. For details on behavioral CDR functions refer to Section 8.3.5.5.

8.4.1.6 *No Behavioral Rx Equalization for 2.5 and 5.0 GT/s*

The combination of worst case channel, behavioral Rx package, and Tx jitter at 2.5 and 5.0 GT/s will yield open eyes, when the appropriate Tx presets are set. Therefore there is no need to define a behavioral Rx equalization or to adjust the Tx equalization setting. Actual implementations of 2.5 and 5.0 GT/s receivers may, of course, include equalization.

8.4.1.7 *Behavioral Rx Equalization for 8.0 and 16.0 GT/s*

As measured at TP2, stressed eyes at 8.0 GT/s and 16 GT/s will usually be closed, making direct measurement of the stressed eye jitter parameters unfeasible. This problem is overcome by employing a behavioral Receiver equalizer that implements both a 1st order CTLE and a 1-tap (8.0 GT/s) or 2-tap DFE (16.0 GT/s).

Rx equalization algorithms of CTLE and DFE are only intended to be a means for obtaining an open eye in the presence of calibration channel ISI plus the other signal impairment terms. The behavioral Rx equalization algorithms are not intended to serve as a guideline for implementing actual Receiver equalization. For example, additional DFE taps can have significant benefit in actual implementations where the CTLE may differ from the behaviorial equalizer and/or CTLE selection may not always be optimal. Channel loss characteristics can vary significantly with temperature and humidity and a real Receiver must be able to continue to function at the target BER through such variations.

8.4.1.8 *Behavioral CTLE (8.0 and 16.0 GT/s Only)*

8.0 and 16.0 GT/s behavioral Rx equalization defines a 1st order CTLE with fixed LF and HF poles, and an adjustable DC gain (A_{DC}) specified according to the family of curves shown in Figure 8-28. For the 8.0 GT/s rates A_{DC} is adjustable over a minimum range of -6 to -12 dB in steps of 1.0 dB.

$$H(s) = \omega_{P2} \frac{s + \omega_{P1} * A_{DC}}{(s + \omega_{P1}) * (s + \omega_{P2})}$$

ω_{P1} = pole 1 = 2π* 2 GHz
ω_{P2} = pole 2 = 2π *8 GHz
A_{DC} = dc gain

Figure 8-27: Transfer Function for 8.0 GT/s Behavioral CTLE

The following diagram illustrates the gain vs. frequency behavior of the CTLE as A_{DC} is varied over its minimum to maximum range in 1.0 dB steps.

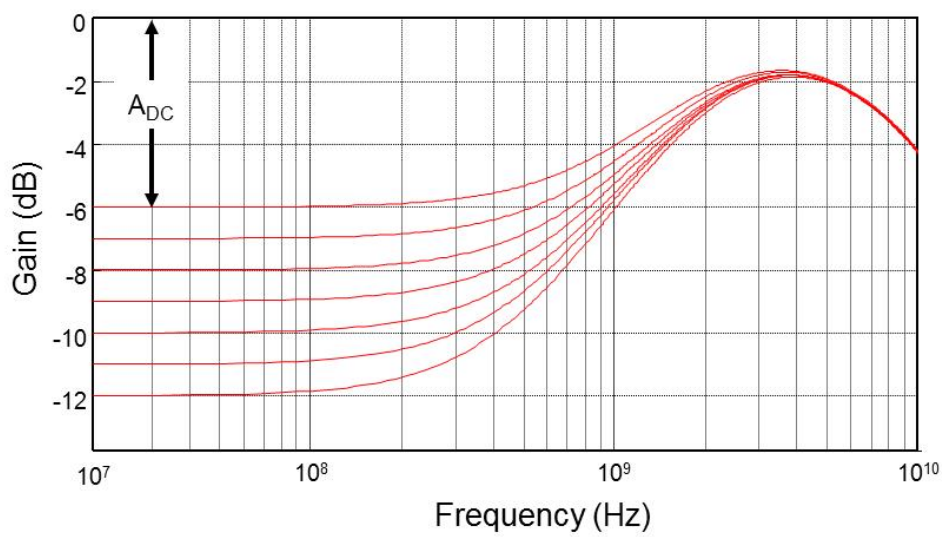


Figure 8-28: Loss Curves for 8.0 GT/s Behavioral CTLE

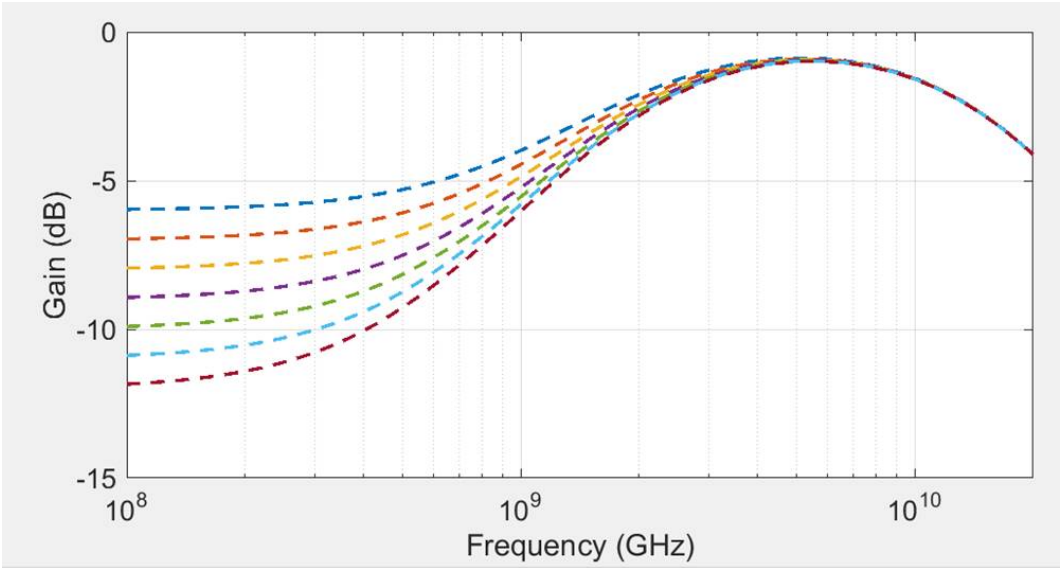
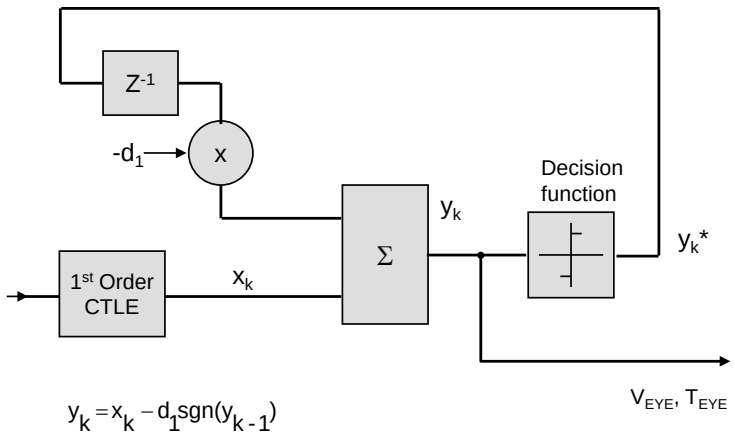


Figure 8-29: Loss Curves for 16.0 GT/s Behavioral CTLE

A Receiver operating at 16.0 GT/s utilizes a similar set of CTLE curves with different pole locations. The difference is that ω_{p1} =pole 1= 2π* 2 GHz and ω_{p2} = pole2 = 2π * 16 GHz. The range for A_{DC} remains the same as that for 8.0 GT/s.

8.4.1.9 *Behavioral DFE (8.0 and 16.0 GT/s Only)*

At 8.0 GT/s the combination of a 1st order CTLE and a one-tap DFE algorithm is required for calibrating the stressed eye when employing the max length calibration channel. The DFE may be represented by the following equation and flow diagram. For 8.0 GT/s and 16.0 GT/s the limits on d_1 are ±30 mV. For 16.0 GT/s the limits on d_2 are ±20 mV.



$y_k = x_k - d_1 \mathrm{sgn}(y_{k-1})$

y_k = DFE summer differential output voltage.

y_k^* = decision function output voltage. $|y_k^*| = 1$

x_k = DFE differential input voltage

d_1 = feedback coefficient

k = sample index in UI

Figure 8-30: Variables Definition and Diagram for 1-tap DFE

16.0 GT/s Receiver tolerancing utilizes a CTLE and a 2-tap behavioral DFE as illustrated below. Other than the inclusion of the second tap, it is identical to the 1-tap DFE shown above.

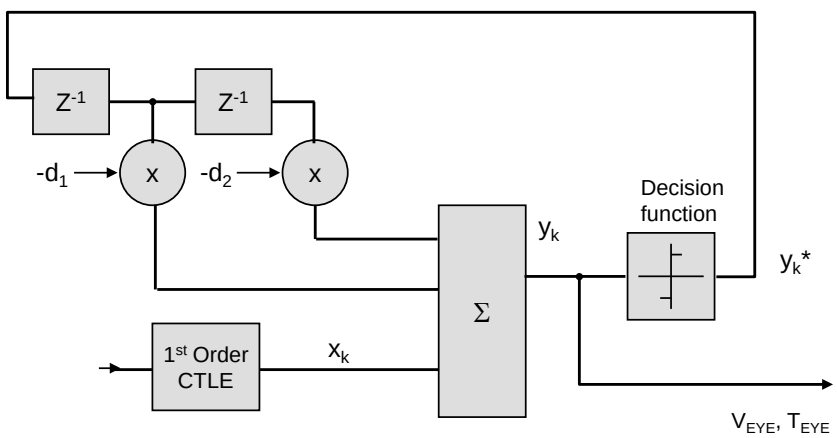


Figure 8-31: Diagram for 2-tap DFE

8.4.2 Stressed Eye Test

The *PCI Express Specification, revision 4.0* simplifies Rx testing by requiring only a single stressed voltage/stressed jitter test per data rate and eliminating the stressed voltage test.

When testing a Receiver it is required to have other PCI Express Lanes on the DUT sending or receiving data. Similarly, if the device supports other I/O, it should also be sending or receiving on these interfaces. The goal is to have the Rx test environment replicate the noise environment found in a real system as closely as possible.

8.4.2.1 *Procedure for Calibrating a Stressed EH/EW Eye*

The goal of calibrating a stressed voltage/jitter eye is to present the Receiver under test with simultaneously worst case margins whose distortion characteristics are similar to an eye produced by a real channel. Much of the distortion consists of the ISI produced by the calibration channel. Incremental changes of Rj and differential voltage are allowed to adjust the EW and EH, respectively at 8.0 GT/s. Incremental changes of Sj and differential voltage from nominal values may be used to adjust the EW and EH, respectively at 16.0 GT/s.

The reference point where EH/EW is defined corresponds to input to the Receiver latch at 8.0 and 16.0 GT/s. Since this point is not physically accessible it is necessary to construct its equivalent by means of a post-processing procedure. A two million unit interval data record of compliance pattern or a step that has been averaged 1024 times at TP2 is first post processed to mathematically include the additional signal distortion caused by the behavioral Receiver package. If a compliance pattern waveform is used then all stresses are turned on – if a step is used then all stresses are turned off.Then the resulting signal is recovered by means of Rx equalization, and a behavioral CDR function, resulting in an equivalent eye. The requirements for the waveform post processing tool used for the EH/EW calibration are described further in Section 8.4.2.1.1.

As the calibration procedure of the signal generator output contains steps where the generator is connected directly to measurement instrumentation, the transition time of the output waveform can be very fast. Therefore, it is important that the bandwidth of instrumentation used to calibrate the generator be matched appropriately to the edge rate of the generator output. This specification requires the use of a generator whose outputs have a rise time of 14ps-19ps (20% / 80%) which also requires a minimum oscilloscope bandwidth of 25GHz. This oscilloscope bandwidth is also the minimum required bandwidth for transmitter measurements.

For the eye calibration process, the Tx equalization is fixed to the preset that gives the optimal eye area with the post processing tool being used for calibration. Once the testing procedure is under way the Tx preset may be adjusted to yield the best eye margins with the DUT. During EH/EW calibration Sj is set to 100 MHz and 0.1 UI. For calibration at 16.0 GT/s the following process is used to calibrate the eye:

1. Calibrate the stress values to the nominal values in Table 8-10.
2. Select an initial test channel length that give a loss at TP2P at 8 GHz of 27 dB±0.5 dB.
3. Measure the eye diagram for each TX EQ preset using the nominal TX Eq for the preset +/- 0.1 dB and select the TX EQ preset that gives the largest eye area.

For all EH, EW and eye area measurements performed in receiver calibration the ADC in the reference receiver CTLE is varied over its minimum to maximum range in .25 dB steps. This is done to improve repeatability and accuracy in automated Rx calibration software and is only done for stressed eye calibration (not for channel compliance, etc.)

4. Increase the calibration channel loss to the next available length/loss and measure the new eye diagram at the selected preset. Continue to increase the length/loss until either the height or width have fallen below the targets in Table 8-10 then the previous calibration channel length/loss is selected. If neither the height or width have fallen below the targets and the TP1 to TP2P loss at 8 GHz has reached 30.0 dB then advance to the next step.

5. For the selected calibration channel length/loss, measure the eye diagram for each TX EQ presetand select the preset that gives the largest eye area. Note that this may be a different preset than step 3 due to the length/loss change.
6. Adjust Sj, DM, and Voltage Swing to make final adjustments to the eye by sweeping them through the following ranges:
   a. Sj – 5 to 10 ps PP.
   b. DM – 10 to 25 mV at TP2.
   c. Differential Voltage Swing – 720 to 800 mV PP at TP1.
7. If the final Sj value is less then 0.1 UI then the Rj level is reduced so the eye width meets the target eye width with 0.1 UI of 100 MHz Sj.
8. If there are multiple combinations of Sj, DM, and Voltage Swing that give valid solutions first pick the combination that is closest to the target eye width (18.75 ps). If there are multiple Sj, DM, and Voltage Swing combinations that are equally close to the target eye width then pick the one with Sj closest to nominal. The selected values must give a mean eye height and width (over at least 5 measurements – exact number of measurements needed for stable values will depend on lab set-up and tools) within the following ranges at BER E-12:
   a. Eye height – 15 mV +/- 1.5 mV
   b. Eye width – 18.75 ps +/- 0.5 ps

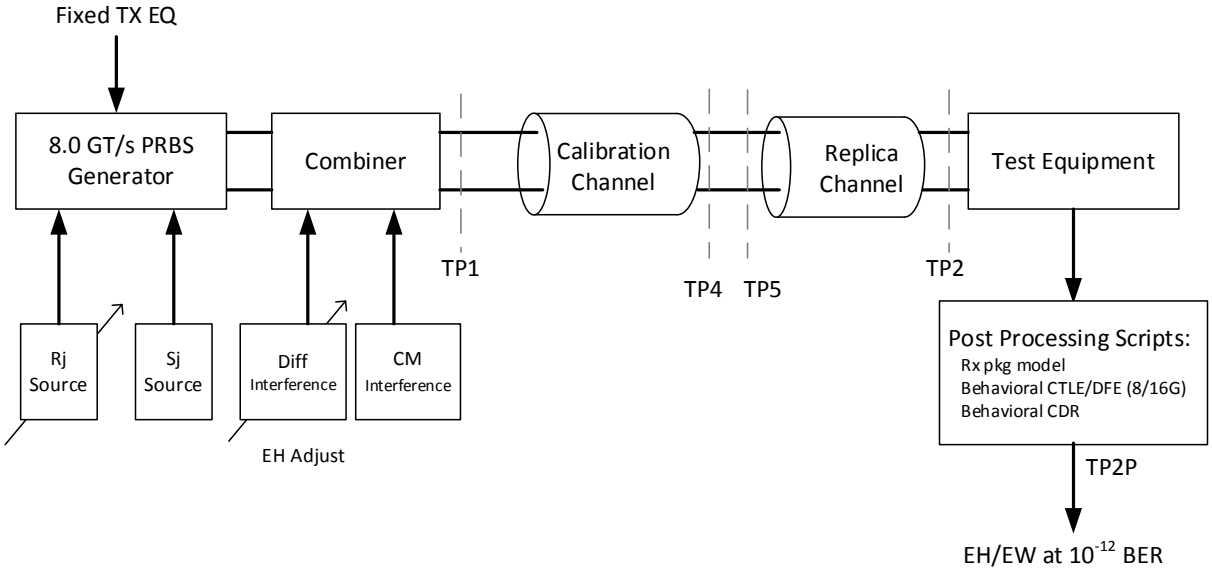


Figure 8-32: Layout for Calibrating the Stressed Jitter Eye at 8.0 GT/s

Based upon the data rate at which the Rx is being tested, Rj or Sj and differential interference sources are adjusted to fall within the VRX-ST and TRX-ST limits. The EH and EW ranges are designed to account for post processing or measurement errors. Figure 8-32 shows the process for calibrating the stressed jitter eye at 8.0 GT/s.

Figure 8-33 shows the process for calibrating the stressed jitter eye at 16.0 GT/s.

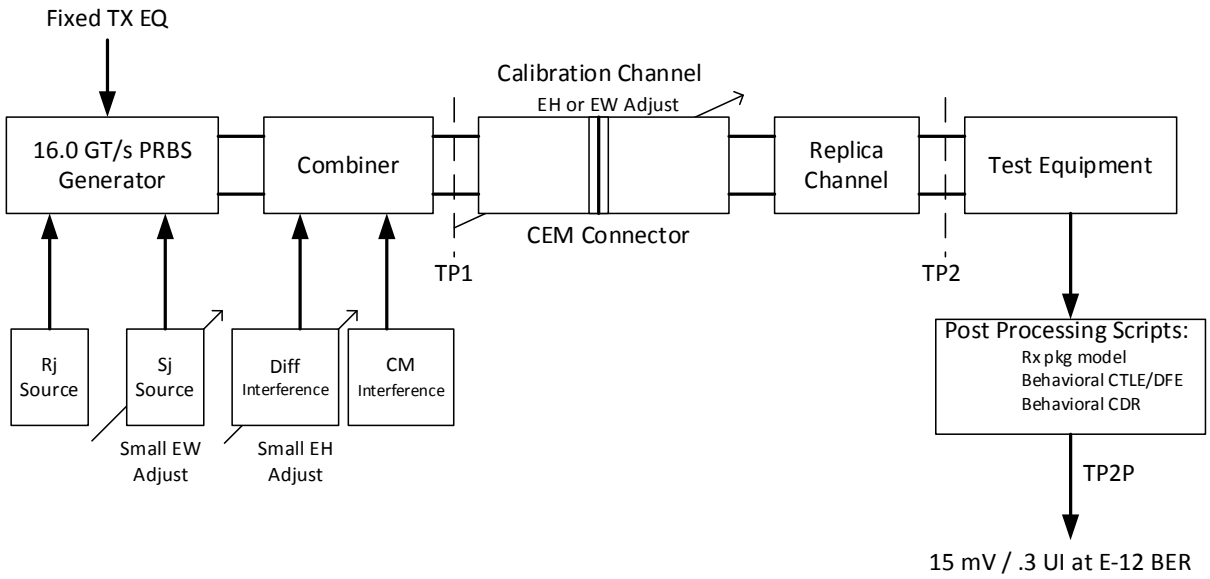


Figure 8-33: Layout for Calibrating the Stressed Jitter Eye at 16.0 GT/s

Table 8-10: Stressed Jitter Eye Parameters

Symbol	Parameter	2.5 GT/s	5.0 GT/s	8.0 GT/s	16.0 GT/s	Units	Details
$V_{RX-LAUNCH}$	Generator launch voltage	800 to 1200	800 to1200	800-to 1200	720 to 800	mV PP	Note 1
T_{RX-UI}	Unit Interval	400	200	125	62.5	ps	
T_{RX-ST}	Eye width	≤0.4	≤0.32	≤0.30	≤0.30	UI	Note 3, 4, 8,10
V_{RX-ST}	Eye height	≤175	≤100	≤25	≤15	mV PP	Note 2,4, 8, 9
$T_{RX-ST-SJ}$	Swept Sj	N/A	75 ps (max) See Note 11	See Section 8.4.2.2.1	See Section 8.4.2.2.1	ps	Note 5
$T_{RX-ST-RJ}$	Random Jitter	N/A	3.4	3.0 (max)	1.0	ps RMS	Note 6,7
$V_{RX-DIFF-INT}$	Differential noise	N/A	N/A	14	14	mV PP	Note 7, 12 Adjust to set EH. Frequency = 2.1 GHz .
$V_{RX-CM-INT}$	Common mode noise	150	150	150	150	mV PP	Note 8
$V_{SSC-RES}$	SSC Residual	N/A	75	N/A	500	ps	Note 11, 13

Notes:

1. $V_{RX-LAUNCH}$ may be adjusted to meet V_{RX-ST} as long as the outside eye voltage at TP2 does not exceed 1300 mVPP for calibration at 2.5, 5.0 and 8.0 GT/s. $V_{RX-LAUNCH}$ is adjusted from 720 to 800 mV for 16 GT/s calibration.
2. Voltages shown for 2.5 GT/s and 5.0 GT/s are at the Rx pins.
3. Eye widths shown for 2.5 GT/s and 5.0 GT/s are at the Rx pins.
4. V_{RX-ST} and T_{RX-ST} are referenced to TP2P for 8.0 GT/s and 16.0 GT/s and TP2 for 2.5 GT/s and 5.0 GT/s. For 8.0 GT/s and 16.0 GT/s behavioral equalization are applied to the data at TP2.
5. $T_{RX-ST-SJ}$ may be measured at either TP1 or TP2. Only 8.0 GT/s and 16.0 GT/s Receivers are tested with Sj mask.
6. $T_{RX-ST-RJ}$ may be adjusted to meet the target value for T_{RX-ST} at 8.0 GT/s. Rj is measured at TP1 to prevent data-channel interaction from adversely affecting the accuracy of the Rj calibration. Rj is applied over the following range: The low frequency limit may be between 1.5 and 10 MHz, and the upper limit is 1.0 GHz.
7. Both $T_{RX-ST-RJ}$ and $V_{RX-DIFF-INT}$ are limited to prevent the stressed eye from containing excessive amounts of jitter or noise distortion that are unrepresentative of a real channel. Too many of these distortion components produces a signal that cannot be equalized by an actual Receiver.
8. Defined as a single tone at 120 MHz. Measurement made at TP2 without post-processing. Common mode is turned off during T_{RX-ST} and V_{RX-ST} calibration and then turned on for the stressed eye jitter test.
9. For 2.5 GT/s and 5.0 GT/s Rx calibration variable channel loss is used to achieve the target eye height.
10. For 2.5 GT/s Rx calibration 100 MHz Sj is used to achieve the target eye width.
11. For 33 kHz SSC residual for common clock architecture testing only when testing at 5 GT/s.
12. Frequency for VRX-DIFF-INT is chosen to be slightly above the first pole of the reference CTLE.
13. Applied for CC testing only as a triangular phase modulation with a frequency between 30 kHz to 33 kHz when testing at 16 GT/s.

8.4.2.1.1 Post Processing Tool Requirements

A waveform post processing tool or a channel compliance methodology tool may be used for Rx stressed eye calibration at 16 GT/s. If a waveform post processing tool is used to calibrate the

EH/EW for the RX stressed eye testing, the tool must be consistent with the channel compliance methodology tool based on a consistency test defined as follows:

- The test channel is the long Rx calibration channel with the Root reference package applied in post-processing to give a total loss of 28.0 dB at 8 GHz. This means that the physical channel loss is 23.0 dB.
- All measurements are done at TP2.
- A step pattern with 256 ones and zeros is captured through the test channel by averaging 1024 times on a real time oscilloscope. The step is saved with an x-axis resolution of 1 ps or less to be used as the transmit waveform for the channel compliance methdology. The step pattern is captured for each preset using the nominal Tx EQ for each preset.
- The channel compliance methodology is run with no Tx EQ applied in simulation using the nominal stress values for Rx stressed eye calibration for each of the captured steps. The Tx EQ preset that produced the largest eye area is selected for exact eye height and width calibration.
- The channel compliance methodology is used with the selected Tx EQ preset to produce an EH/EW of 15 mV and 0.3 UI @ E-12 BER by adjusting the Sj and voltage swing at the Transmitter output.
- A pattern generator is calibrated to have the same jitter stress levels and Tx Swing as those used in the channel compliance simulations that produced an EH/EW of 15 mV and 0.3 UI @ E-12 BER at the pattern generator output.
- 2 million unit interval waveforms with compliance pattern are captured at each Tx EQ at the end of the channel. The Tx EQ is calibrated to the nominal values for each preset at the pattern generator output before doing the captures.
- For the preset that gives the largest eye area with the waveform post procesing tool the EH and EW @ E-12 BER must match 15 mV and 0.3 UI within +/- 15%.

8.4.2.2 *Procedure for Testing Rx DUT*

8.4.2.2.1 Sj Mask

Once a calibrated EH and EW have been obtained, the cables are moved to connect the Rx DUT to the far end of calibration channel. The Tx equalization may then be optimized with the assumption that the DUT Rx will also optimize its equalization. Sj is set to an initial value of 0.1 UI at 100 MHz and the Receiver CDR must achieve lock. At 8.0 GT/s and 16.0 GT/s the 100 MHz Sj initial tone is removed and then the appropriate swept Sj profile is tested. At 16.0 GT/s an additional Sj tone at 210 MHz is present for all testing. The amplitude of this additional tone is equal to the amplitude of the 100 MHz Sj required to achieve the target eye width minus 0.1 UI. If the calibration Sj level was less than 0.1 UI then no additional tone at 210 MHz is used. Different Sj profiles are used, depending on whether the Rx under test operates in the CC mode or the IR Refclk mode. See Table 8-10.

Receivers operating at 8.0 GT/s in the IR Refclk mode use the Sj mask profile shown in Figure 8-34, consisting of a fixed 33 kHz tone plus a swept tone from 400 kHz to 100 MHz that follows the curve shown in the figure. A Receiver must meet the 10^{-12} BER over the entire swept Sj range. It is not necessary to test a Receiver over the entire Sj frequency mask range, but a sufficient number

of frequency points should be tested to guarantee that the Rx does not fail BER at some resonance frequency. The magnitude of the 33 kHz spur is 25 ns pp, which corresponds to 200 UI pp at 8.0 GT/s and 400 UI pp at 16.0 GT/s. Swept Sj is required only for testing Receivers at 8.0 GT/s and 16.0 GT/s. Receiver operation at 2.5 GT/s and 5.0 GT/s is tested using a single 33 kHz Sj tone.

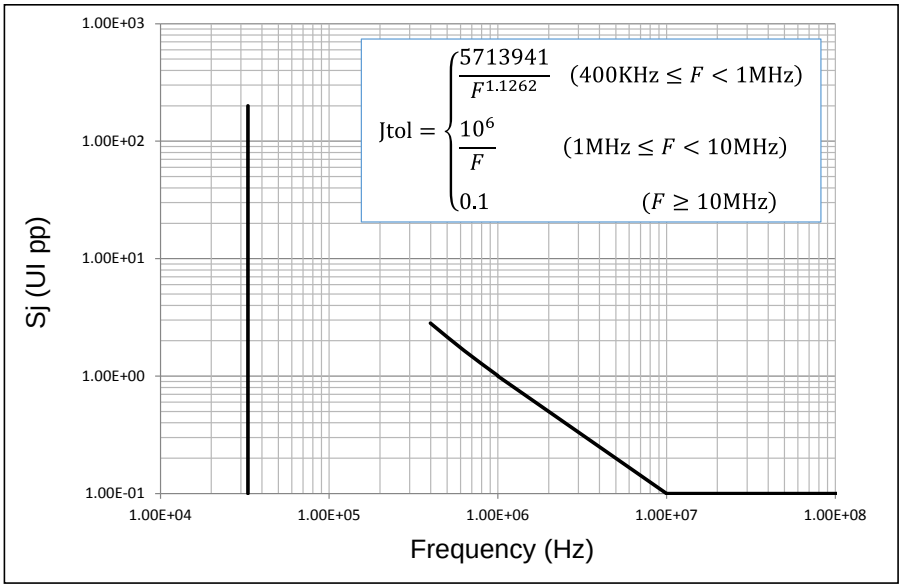


Figure 8-34: Sj Mask for Receivers Operating in IR mode at 8.0 GT/s

The Sj mask function for 16.0 GT/s differs slightly from that of 8.0 GT/s in the 400 kHz-1.0 MHz range as well as in the magnitude of the 33 kHz single tone. The latter applies 400 UI, vs. 200 UI at 8.0 GT/s, to account for the halving of the UI interval at 16.0 GT/s.

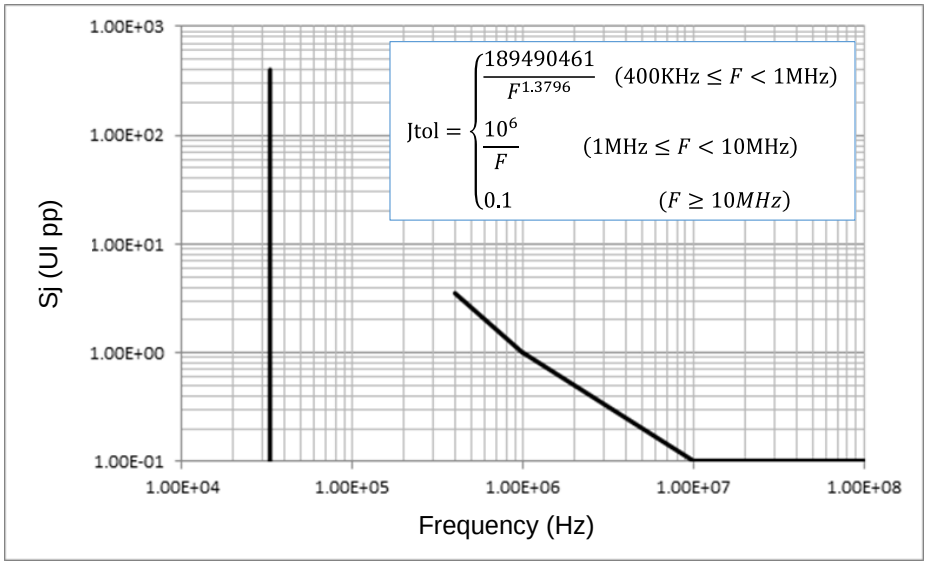


Figure 8-35: Sj Mask for Receivers Operating in IR mode at 16.0 GT/s

Receivers operating the CC Refclk mode shall utilize the Sj profile shown in Figure 8-36. The testing procedure is essentially identical as for IR clocking modes, except that the clock topology differs. See Section 8.4.2.3 for details.

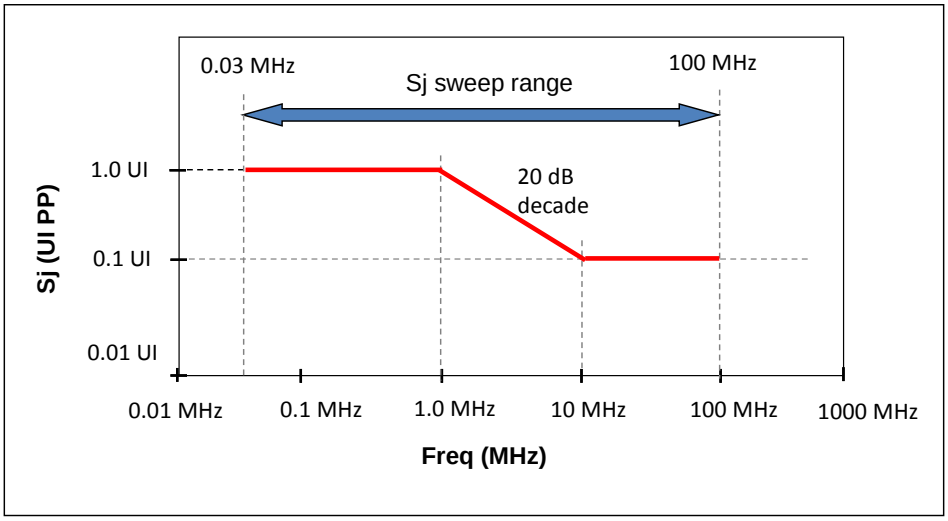


Figure 8-36: Sj Masks for Receivers Operating in CC Mode at 8.0 GT/s and 16.0 GT/s

8.4.2.3 *Receiver Refclk Modes*

A Rx is permitted to support one or both of two clock modes: CC, and IR although only one clock mode may be operational at a given time. Receivers can support more than one Refclk mode by selecting a mode at power-up or by means of strapping pins, etc.

8.4.2.3.1 Common Refclk Mode

Figure 8-37 shows the Refclk connection for a receiver in the Common Clock Refclk mode. A single Refclk source drives both the Generator and the DUT. It is typical that SSC would be applied. This test utilizes the Sj mask specified in section 8.4.2.2.1.

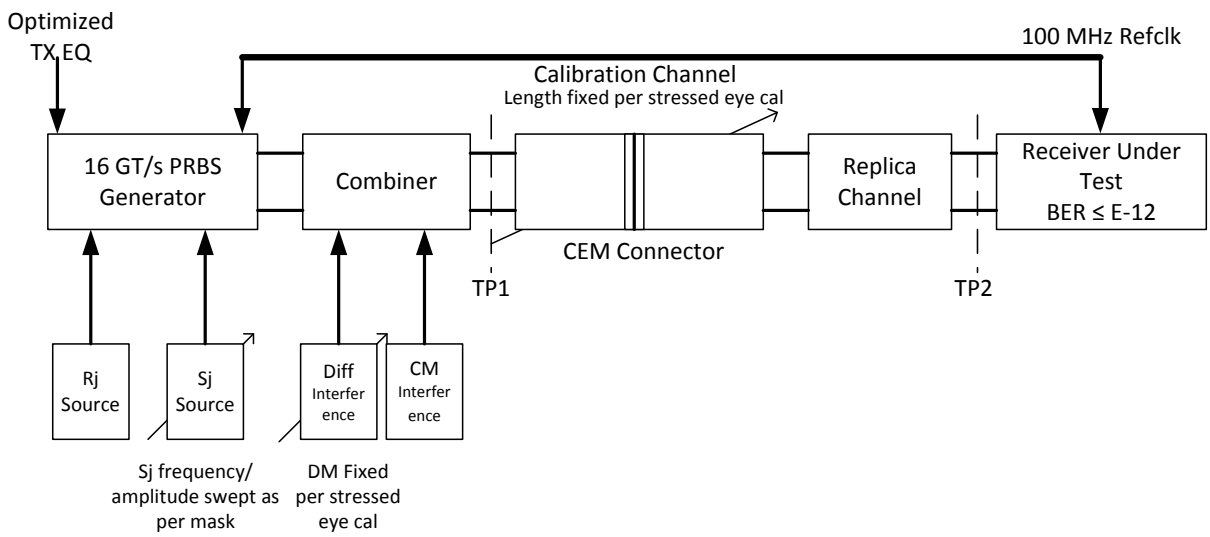


Figure 8-37: Layout for Jitter Testing Common Refclk Rx at 16.0 GT/s

8.4.2.3.2 Independent Refclk Mode

Figure 8-38 illustrates the configuration for testing a Receiver in the IR Refclk mode. Two Refclk sources, with independent SSC, are required. The test utilizes the Sj mask specified in section 8.4.2.2.1.

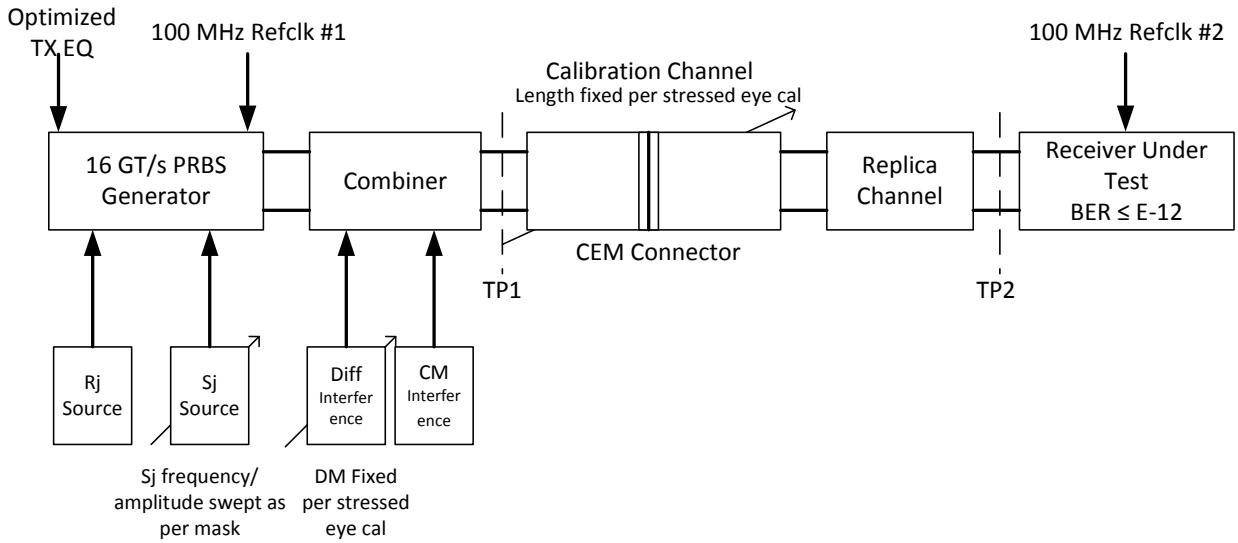


Figure 8-38: Layout for Jitter Testing for Independent Refclk Rx at 16.0 GT/s

8.4.3 Common Receiver Parameters

Table 8-11 lists the common Receiver parameters that are not directly associated with stressed eye tolerancing. Values are separately defined for the four data rates.

Table 8-11: Common Receiver Parameters

Symbol	Parameter	2.5 GT/s Value	5.0 GT/s value	8.0 GT/s Value	16.0 GT/s value	Units	Notes
UI	Unit Interval	399.88 (min) 400.12 (max)	199.94 (min) 200.06 (max)	**124.9625 (min) 125.0375 (max)**	62.48125 (min) 62.51875 (max)	ps	UI is tolerance is equivalent to ±300 ppm and does not include SSC effects
$BW_{RX-PKG-PLL1}$	Rx PLL bandwidth corresponding to $PKG_{RX-PLL1}$	22 (max), 1.5 (min)	16 (max), 8 (min)	**4.0 (max), 2.0 (min)**	4.0 (max), 2.0 (min)	MHz	Second order PLL transfer bounding function
$BW_{RX-PKG-PLL2}$	Rx PLL bandwidth corresponding to $PKG_{RX-PLL2}$	Not Specified	16 (max), 5.0 (min)	**5.0 (**max**), 2.0 (min)**	5.0 (max), 2.0 (min)	MHz	Second order PLL transfer bounding function
$PKG_{RX-PLL1}$	Maximum Rx PLL peaking corresponding to $BW_{RX-PKG-PLL1}$	3.0 (max)	3.0	**2.0**	2.0	dB	Second order PLL transfer bounding function
$PKG_{RX-PLL2}$	Maximum Rx PLL peaking corresponding to $BW_{RX-PKG-PLL2}$	Not specified	1.0	**1.0**	1.0	dB	Second order PLL transfer bounding function
$RL_{RX-DIFF}$	Differential receiver return loss	See Figure 8-19	See Figure 8-19	See Figure 8-19	See Figure 8-19	dB	
RL_{RX-CM}	Common mode receiver return loss	See Figure 8-20	See Figure 8-20	See Figure 8-20	See Figure 8-20	dB	
$RX_{GND-FLOAT}$	Rx termination float time	500 (max)	500 (max)	500 **(max)**	500 (max)	ns	Limits added for 2.5 GT/s and 5.0 GT/s that match those for 8.0 GT/s
$V_{RX-CM-AC-P}$	Rx AC common Mode Voltage	150 (max)	150 (max)	**75 (max) for EH< 100 mVPP 125 (max) for EH≥ 100 mVPP**	75 (max) for EH< 100 mVPP 125 (max) for EH≥ 100 mVPP	mVP	Measured at Rx pins into a pair of 50Ω terminations to ground
Z_{RX-DC}	Receiver DC single ended impedance	40 (min) 60 (max)	40 (min) 60 (max)	Not specified	Not specified	Ω	DC impedance limits are needed to guarantee Receiver detect. For 8.0 and 16.0 GT/s is bounded by RLRX-CM. See Note 5.
$Z_{RX-HIGH-IMP-DC-POS}$	DC input CM input impedance for V≥0 during Reset or power-down	≥10K (0-200 mV) ≥20K (> 200 mV)	≥10K (0-200 mV ≥20K (> 200 mV)	**≥10K (0-200 mV ≥20K (> 200 mV)**	≥10K **(0-200 mV ≥20K (> 200 mV)**	Ω	Voltage measured wrt. ground. Parameters may not scale with process technology.
$Z_{RX-HIGH-IMP-DC-NEG}$	DC input CM input impedance for V<0 during Reset or power-down	1.0K (min)	1.0K (min)	**1.0K (min)**	1.0K (min)	Ω	Parameters may not scale with process technology.

Symbol	Parameter	2.5 GT/s Value	5.0 GT/s value	8.0 GT/s Value	16.0 GT/s value	Units	Notes
$V_{RX-IDLE-DET-DIFF-PP}$	Electrical Idle Detect threshold	65 (min) 175 (max)	65 (min) 175 (max)	**65 (**min**)** **175 (max)**	65 (min) 175 (max)	mV	$V_{RX-IDLE-DET-DIFFp-p} = 2*\|V_{RX-D+} - V_{RX-D-}\|$. Measured at the package pins of the Receiver.
$T_{RX-IDLE-DET-DIFF-ENTERTIME}$	Unexpected Electrical Idle Enter Detect Threshold Integration Time	10 (max)	10 (max)	**10 (max)**	10 (max)	ms	An unexpected Electrical Idle ($V_{RX-DIFF-PP} < V_{RX-IDLE-DET-DIFFp-p}$) must be recognized no longer than $T_{RX-IDLE-DET-DIFF-ENTERTIME}$ to signal an unexpected idle condition.
$L_{RX-SKEW}$	Lane to Lane skew	20 (max)	8 (max)	**6 (max)**	5 (max)	ns	Across all Lanes on a Port. $L_{RX-SKEW}$ comprehends Lane-Lane variations due to channel and repeater delay differences.

Notes:

1. Receiver eye margins are defined into a 2 x 50 Ω reference load. A Receiver is characterized by driving it with a signal whose characteristics are defined by the parameters specified in.

2. The four inherent timing error parameters are defined for the convenience of Rx designers, and they are measured during Receiver tolerancing.

3. Two combinations of PLL BW and peaking are specified at 5.0 GT/s to permit designers to make tradeoffs between the two parameters. If the PLL's min BW is ≥8 MHz, then up to 3.0 dB of peaking is permitted. If the PLL's min BW is relaxed to ≥5.0 MHz, then a tighter peaking value of 1.0 dB must be met. Note: a PLL BW extends from zero up to the value(s) defined as the min or max in the above table. For 2.5 GT/s a single PLL bandwidth and peaking value of 1.5-22 MHz and 3.0 dB are defined.

4. Measurements must be made for both common mode and differential return loss. In both cases the DUT must be powered up and DC isolated, and its D+/D- inputs must be in the low-Z state.

5. The Rx DC single ended impedance must be present when the Receiver terminations are first enabled to ensure that the Receiver Detect occurs properly. Compensation of this impedance is permitted to start immediately and the Rx Common Mode Impedance (constrained by RL_{RX-CM} to 50 Ω ±20%) must be within the specified range by the time Detect is entered.

6. Common mode peak voltage is defined by the expression: max{|(Vd+ - Vd-) - V_{-CMDC}|}.

7. $Z_{RX-HIGH-IMP-DC-NEG}$ and $Z_{RX-HIGH-IMP-DC-POS}$ are defined respectively for negative and positive voltages at the input of the Receiver. Transmitter designers need to comprehend the large difference between >0 and <0 Rx impedances when designing Receiver detect circuits.

8. Defines the time for the Receiver's input pads to settle to new common-mode on 2.5/5.0 GT/s transition to 8.0 GT/s.

8.4.3.1 *5.0 GT/s Exit From Idle Detect (EFI)*

It is difficult to scale the capabilities of the EFI detect circuits with data rate, and for this reason the 5.0, 8.0 and 16.0 GT/s specification defines different data patterns in the FTS and the TS1 and TS2

Ordered Sets than are defined for 2.5 GT/s operation. In particular, repeated K28.7 patterns are defined to guarantee sufficient voltage and time requirements, as illustrated in the figure below. Concatenated EIE Symbols yield alternating one/zero run lengths of five UI each.

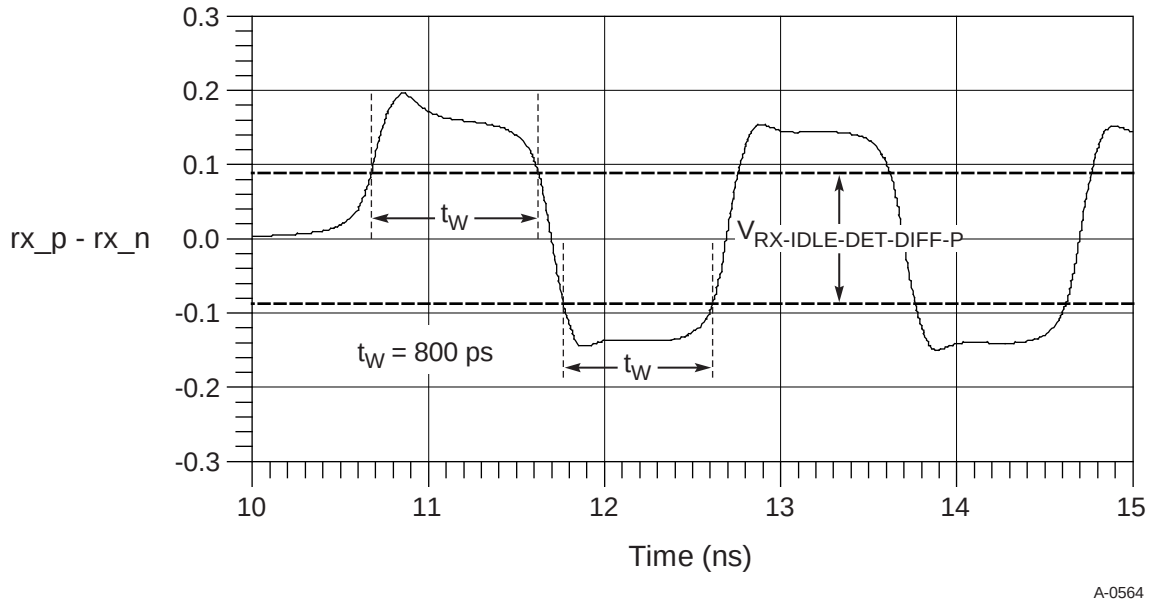


Figure 8-39: Exit from Idle Voltage and Time Margins

8.4.3.2 *Receiver Return Loss*

The measurement methodology and frequency binning for differential and common mode Rx RL is identical to that for the Tx. The only difference exists between the 2.5-4.0 GHz differential RL limits. For details refer to Figure 8-19 and Figure 8-20.

8.4.4 Lane Margining at the Receiver – Electrical Requirements

PCI Express components including retimers that support the 16.0 GT/s rate are required to support Lane margining at the Receiver. Lane Margining enables system software to get the margin information of a given Lane while the Link is in L0 state. The margin information includes both voltage and time, in either direction from the current Receiver position. The margin feature is not permitted to require any additional external hardware to function. Support of Lane margining for voltage is optional and support of independent timing margin to the left or to the right is optional. For simplicity, the margin commands and requirements described in the protocol chapter(s) of this specification are described in terms of moving the data sample location - but the actual margining method is implementation specific. For example - the timing margin could be implemented on the actual data sampler or an independent/error sampler. Further the timing margin can be achieved by injecting an appropriate amount of stress/jitter to the data sample location, or by actually moving the data/error sample location.

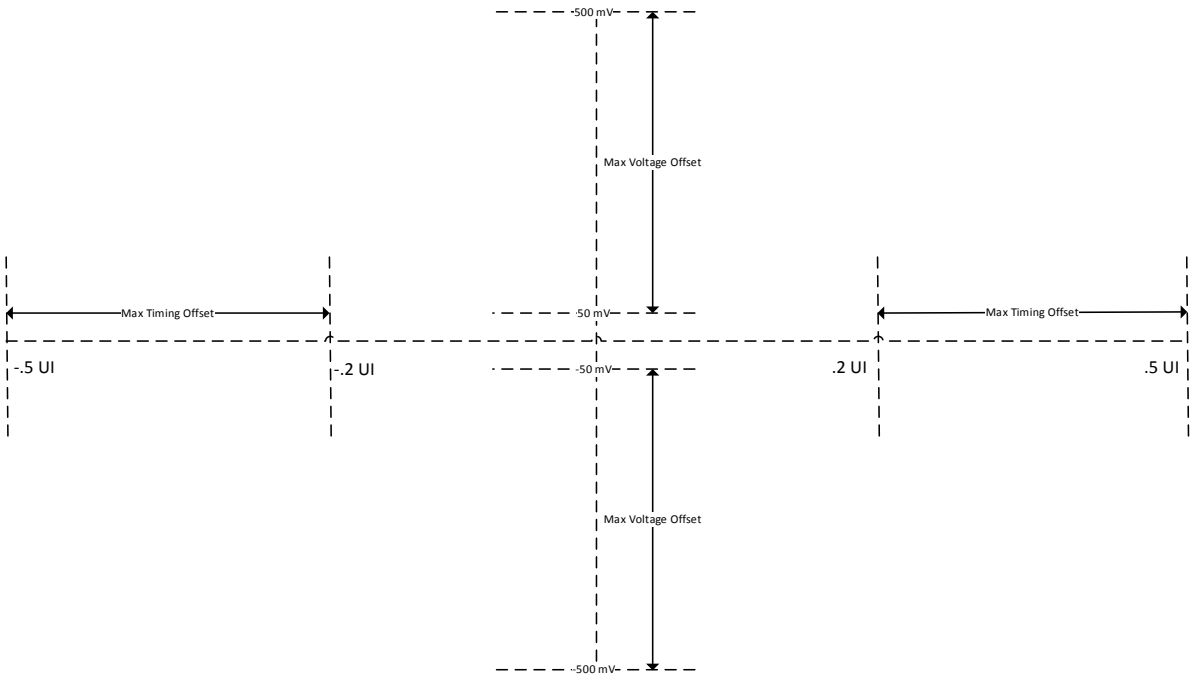


Figure 8-40: Allowed Ranges for Maximum Timing and Voltage Margins

Table 8-12: Lane Margining Timing

Parameter Name	Min	Max	Description
$M_{NumTimingSteps}$	6	63	Number of time steps from default (to either left or right), range must be at least +/-0.2 UI Timing offset must increase monotonically The number of steps in both positive (toward the end of the unit interval) and negative (toward the beginning of the unit interval) must be identical
$M_{MaxTimingOffset}$	20	50	Offset from default at maximum step value as percentage of a nominal UI at 16.0 GT/s A 0 value may be reported if the vendor chooses not to report the offset
$M_{NumVoltageSteps}$	32	127	Number of voltage steps from default (either up or down), minimum range +/-50 mV as measured by 16.0 GT/s reference equalizer Voltage offset must increase monotonically The number of steps in both positive and negative direction from the default sample location must be identical This value is undefined if $M_{VoltageSupported}$ is 0b
$M_{MaxVoltageOffset}$	5	50	Offset from default at maximum step value as percentage of one volt A 0 value may be reported if the vendor chooses not to report the offset when $M_{VoltageSupported}$ is 1b This value is undefined if $M_{VoltageSupported}$ is 0b
$M_{SamplingRateVoltage}$	0	63	The ratio of bits tested to bits received during voltage margining. A value of 0 is a ratio of 1:64 (1 bit of every 64 bits received), and a value of 63 is a ratio of 64:64 (all bits received).
$M_{SamplingRateTiming}$	0	63	The ratio of bits tested to bits received during timing margining. A value of 0 is a ratio of 1:64 (1 bit of every 64 bits received), and a value of 63 is a ratio of 64:64 (all bits received).
$M_{VoltageSupported}$	0	1	1b indicates that voltage margining is supported
$M_{IndLeftRightTiming}$	0	1	1b indicates independent left/right timing margin supported
$M_{IndUpDownVoltage}$	0	1	1b independent up and down voltage margining supported
$M_{IndErrorSampler}$	0	1	1b Margining will not produce errors (change in the error rate) in data stream (ie. – error sampler is independent) 0b Margining may produce errors in the data stream
$M_{MaxLanes}$	0	31	Maximum number of Lanes minus 1 that can be margined at the same time. It is recommended that this value be greater than or equal to the number of Lanes in the Link minus 1. Encoding Behavior is undefined if software attempts to margin more than $M_{MaxLanes}$+1 at the same time. Note: This value is permitted to exceed the number of Lanes in the Link minus 1.
$M_{SampleReportingMethod}$	0	1	Indicates whether sampling rates ($M_{SamplingRateVoltage}$ and $M_{SamplingRateTiming}$) are supported (1) or a sample count is supported (0). One of the two methods is supported by each device.

Parameter Name	Min	Max	Description
$M_{ErrorCount}$	0	63	If $M_{IndErrorSampler}$ is 1b this is a count of the actual bit errors since margining started. If $M_{IndErrorSampler}$ is 0b this is the actual count of the logical errors since margining started. See the Physical Layer Logical Block chapter for the definition of what errors are counted. The count saturates at 63.
$M_{SampleCount}$	0	127	Value = $3*\log_2$ (number of bits margined). Where number of bits margined is a count of the actual number of bits tested during marging. The count stops when margining stops. The count saturates at 127 (after approximately 5.54×10^{12} bits). The count resets to zero when a new margin command is received.

8.4.5 Low Frequency and Miscellaneous Signaling Requirements

The parameters defined in this section are relevant to 2.5, 5.0, 8.0 and 16 GT/s designs.

8.4.5.1 *ESD Standards*

All PCI Express signal and power supply pins must be tested for ESD protection levels to the Human Body Model (HBM) and the Charged Device Model (CDM) standards in accordance with ESDA/JEDEC JS-001-2010 (for HBM) and in accordance with JEDEC JESD22-C101 (for CDM). Pins must meet or exceed the minimum levels recommended in JEDEC JEP155/JEP157 (HBM/CDM) or JEDEC approved superseding documents.

8.4.5.2 *Channel AC Coupling Capacitors*

Each Lane of a PCI Express Link must be AC coupled. The minimum and maximum values for the capacitance are given in Table 8-8. Capacitors must be placed on only one side of an interface that permits adapters to be plugged and unplugged. Form factor specifications must define the required location of the capacitor. In a topology where everything is located on a single substrate, the capacitors may be located anywhere along the channel. External capacitors are assumed because the values required are too large to feasibly construct on-chip.

8.4.5.3 *Short Circuit Requirements*

All Transmitters and Receivers must support surprise hot insertion/removal without damage to the component. The Transmitter and Receiver must be capable of withstanding sustained short circuit to ground of D+ and D-.

8.4.5.4 *Transmitter and Receiver Termination*

- The Transmitter is required to meet $RL_{TX-DIFF}$ and RL_{TX-CM} (see Figure 8-19 and Figure 8-20) any time functional differential signals are being transmitted.
- The Transmitter is required only to meet $I_{TX-SHORT}$ (see Table 8-8 any time functional differential signals are not being transmitted.
- Note: The differential impedance during this same time is not defined.
- The Receiver is required to meet $RL_{RX-DIFF}$ and RL_{RX-CM} (see Table 8-21 and Table 8-22) during all LTSSM states excluding only times during when the device is powered down, Fundamental Reset is asserted, or when explicitly specified.
- The Receiver is required to meet $Z_{RX-HIGH-IMP-DC-NEG}$ and $Z_{RX-HIGH-IMP-DC-POS}$ (see Table 8-8) any time adequate power is not provided to the Receiver, Fundamental Reset is asserted, or when explicitly specified.

8.4.5.5 *Electrical Idle*

Electrical Idle is a steady state condition where the Transmitter D+ and D- voltages are held constant at the same value. Electrical Idle is primarily used in power saving and inactive states (e.g., Disabled).

Before a Transmitter enters Electrical Idle, it must always send the required number of EIOSs except for the LTSSM substates explicitly exempted from this requirement. After sending the last Symbol of the last of the required number of EIOSs, the Transmitter must be in a valid Electrical Idle state within the time as specified by $T_{TX-IDLE-SET-TO-IDLE}$ in Table 8-8.

The successful reception of an EIOS occurs based on the rules defined in the Physical Layer Logical Block chapter. It should be noted that in substates (e.g., Loopback.Active for a loopback slave) where multiple consecutive EIOSs are expected, the Receiver must receive the appropriate number of EIOS sequences comprising of COM, IDL, IDL, IDL.

The low impedance common mode and differential Receiver terminations values (see Table 8-8 and Table 8-11) must be met in Electrical Idle. The Transmitter can be in either a low or high impedance mode during Electrical Idle.

Any time a Transmitter enters Electrical Idle it must remain in Electrical Idle for a minimum of $T_{TX-IDLE-MIN}$. The Receiver should expect the last EIOS followed by a minimum amount of time in Electrical Idle ($T_{TX-IDLE-MIN}$) to arm its Electrical Idle Exit detector.

When the Transmitter transitions from Electrical Idle to a valid differential signal level it must meet the output return loss specifications described in Figure 8-19 and Figure 8-20.

Electrical Idle Exit shall not occur if a signal smaller than $V_{RX-IDLE-DET-DIFFp-p}$ minimum is detected at a Receiver. Electrical Idle Exit shall occur if a signal larger than $V_{RX-IDLE-DET-DIFFp-p}$ maximum is detected at a Receiver. Electrical Idle may be detected on the received signal regardless of its frequency components, or it may be detected only when the received signal is switching at a frequency of 125 MHz or higher.

8.4.5.6 *DC Common Mode Voltage*

The Receiver DC common mode voltage is nominally 0 V when operating at 2.5 GT/s or 5.0 GT/s.

Transmitter DC common mode voltage is held at the same value during all states unless otherwise specified. The range of allowed Transmitter DC common mode values is specified in Table 8-11 ($V_{TX\text{-}DC\text{-}CM}$).

8.4.5.7 *Receiver Detection*

The Receiver detection circuit is implemented as part of a Transmitter and must correctly detect whether a load impedance equivalent to a DC impedance implied by the $Z_{RX\text{-}DC}$ parameter (40 Ω-60 Ω) is present. Note: Support for Rx detect, which only occurs at 2.5 GT/s, is the reason why 2.5 GT/s Receivers impedance at DC is specified.

The recommended behavior of the Receiver detection sequence is described below:

Step 1. A Transmitter must start at a stable voltage prior to the detect common mode shift.

Step 2. A Transmitter changes the common mode voltage on D+ and D- consistent with meeting the $V_{TX\text{-}RCV\text{-}DETECT}$ parameter and consistent with detection of Receiver high impedance which is bounded by parameters $Z_{RX\text{-}HIGH\text{-}IMP\text{-}DC\text{-}POS}$, $Z_{RX\text{-}HIGH\text{-}IMP\text{-}DC\text{-}NEG}$ in Table 8-11. Receiver is detected based on the rate that the lines change to the new voltage.

a. The Receiver is not present if the voltage at the Transmitter charges at a rate dictated only by the Transmitter impedance and the capacitance of the interconnect and series capacitor.

b. The Receiver is present if the voltage at the Transmitter charges at a rate dictated by the Transmitter impedance, the series capacitor, the interconnect capacitance, and the Receiver termination.

Any time Electrical Idle is exited the detect sequence does not have to execute or may be aborted on that Lane.

If an implementation chooses to transition from Detect to Polling based on Electrical Idle being broken prior to performing a Receiver detect sequence, an unreliable Link could be formed; due to the possibility that there may not be a low impedance termination resistor on both Rx differential conductors, which make up the differential pair.

If the Receiver detection circuit performs the detect sequence on each conductor of the differential pair (both D+ and D-) and detects a load impedance greater than $Z_{RX\text{-}DC}$ on either conductor, the Receiver detection circuit shall interpret this as no termination load present and respond as if neither load were present.

It is required that the detect sequence be performed on both conductors of a differential pair.

8.5 Channel Tolerancing

8.5.1 Channel Compliance Testing

This section of the specification is relevant only for those cases where a platform design comprehends the relevant channel between Transmitter device pins and Receiver device pins. These types of platform designs are called "captive channels". Designs that are not captive channels shall refer to the appropriate form factor (CEM is one example) specification, since in this case the form factor specification takes precedence over the *PCI Express Base Specification* and splits the channel between two different types of components.

The key components and processes of channel tolerancing are illustrated in Figure 8-41 and Figure 8-42. The major difference lies in the complexity of the Behavioral Tx and Rx equalization, which depends on the data rate. 2.5 and 5.0 GT/s utilize fixed Tx presets and assume no Rx equalization, whereas 8.0 and 16.0 GT/s assume multi-valued, adjustable Tx presets and a combination of Rx DFE and CTLE.

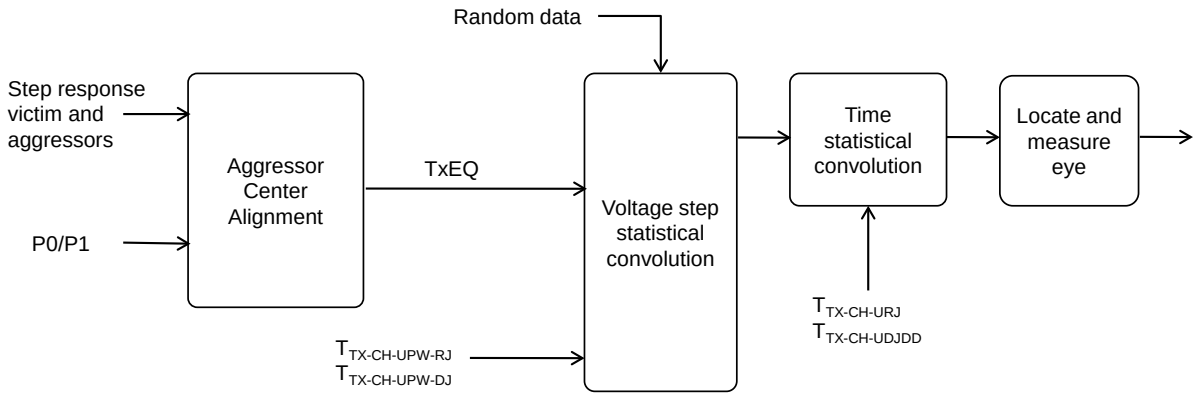


Figure 8-41 : Flow Diagram for Channel Tolerancing at 2.5 and 5.0 GT/s

The basic channel compliance approach is to first acquire the channel's characteristics, usually by means of s-parameters or equivalent model. Behavioral Tx and Rx package models are then appended to the channel model to yield a die pad to die pad topology. The model shall include both victim path and a sufficient number of aggressor paths to accurately capture channel crosstalk effects. Using the Tx voltage and jitter limits defined in the Transmitter specification section it is possible to transform these parameters to what would appear at the die pad of a Tx.

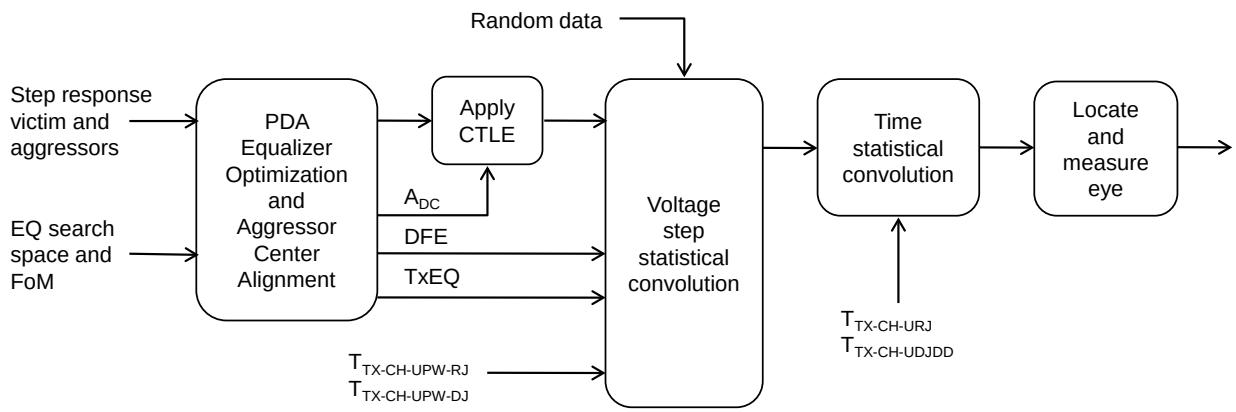


Figure 8-42: Flow Diagram for Channel Tolerancing at 8.0 and 16.0 GT/s

The resulting model is analyzed via simulation, yielding voltage and jitter at a point equivalent to the input latch of the Receiver. The signal observed at the Receiver's latch is referenced to a recovered data clock from which an eye diagram may be constructed.

For 8.0 GT/s and 16.0 GT/s testing the simulation process must properly account for Tx and Rx equalization optimization as must be supported by a minimum capability Tx/Rx pair. This means the simulation process must be able to select the optimum values for the Tx presets or coefficients and Rx equalization settings based upon a 1st order CTLE and a 1-tap DFE for 8.0 GT/s and a 1st order CTLE and a 2-tap DFE for 16.0 GT/s.

8.5.1.1 *Behavioral Transmitter and Receiver Package Models*

Behavioral package models are defined in this specification to represent the combined die and package loss that is expected to interoperate with the targeted range of channels. Note that at 16.0 GT/s, the behavioral packages represent a high, but not worst case, loss for many devices. (see sections 8.3.3.11 and 8.4.1.4)

A separate pair of package models is defined for 8.0 and 16.0 GT/s. At 8.0 GT/s, separate package models are defined for TX and RX ports to reflect the smaller CPAD capacitance typical in most reciever implementations. At 16.0 GT/s, separate package models are defined for devices containing Root Ports and all other devices to reflect the large and socketed nature of most devices containing Root Complexes. Channel testing for both data rates typically requires testing in both directions.

The package models are included with the specification as design collateral. Each model comprehends C_{PIN} and C_{PAD} parasitic capacitances plus a differential t-line element as illustrated in Figure 8-43.

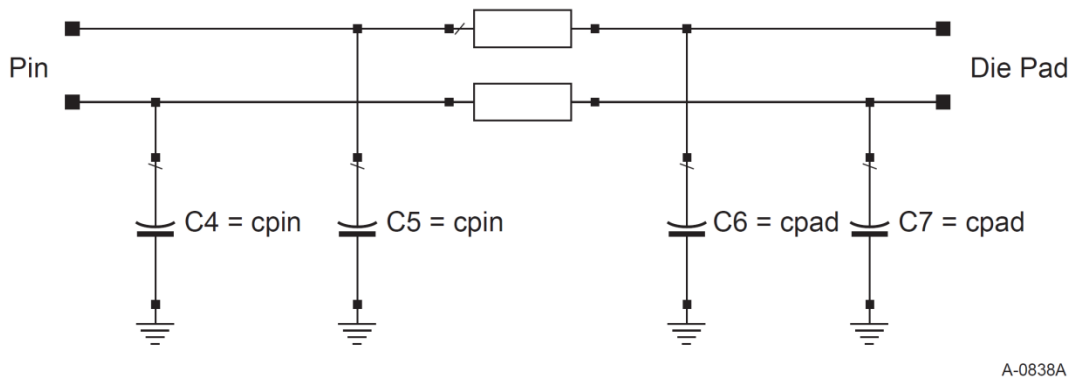


Figure 8-43: Tx/Rx Behavioral Package Models

The C_{PIN} and C_{PAD} values used in the package model generation are provided for informative purposes only.

Table 8-13: Package Model Capacitance Values

	8.0 GT/s Tx	8.0 GT/s Rx	16.0 GT/s
C_{PIN}	0.25 pF	0.25 pF	0.25 pF
C_{PAD}	1.0 pF	0.8 pF	0.5 pF

For ease of incorporation into the post processing flow the behavioral package models are specified as 4-port s-parameter files. The files are specified with port designations, frequency range and granularity as listed below. The reference impedance for the s-parameters is 50 Ω. File format is Touchstone.

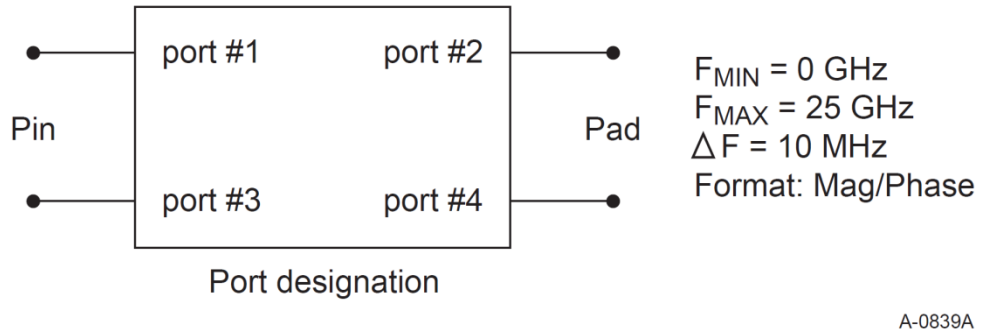


Figure 8-44: Behavioral Tx and Rx S-Port Designation

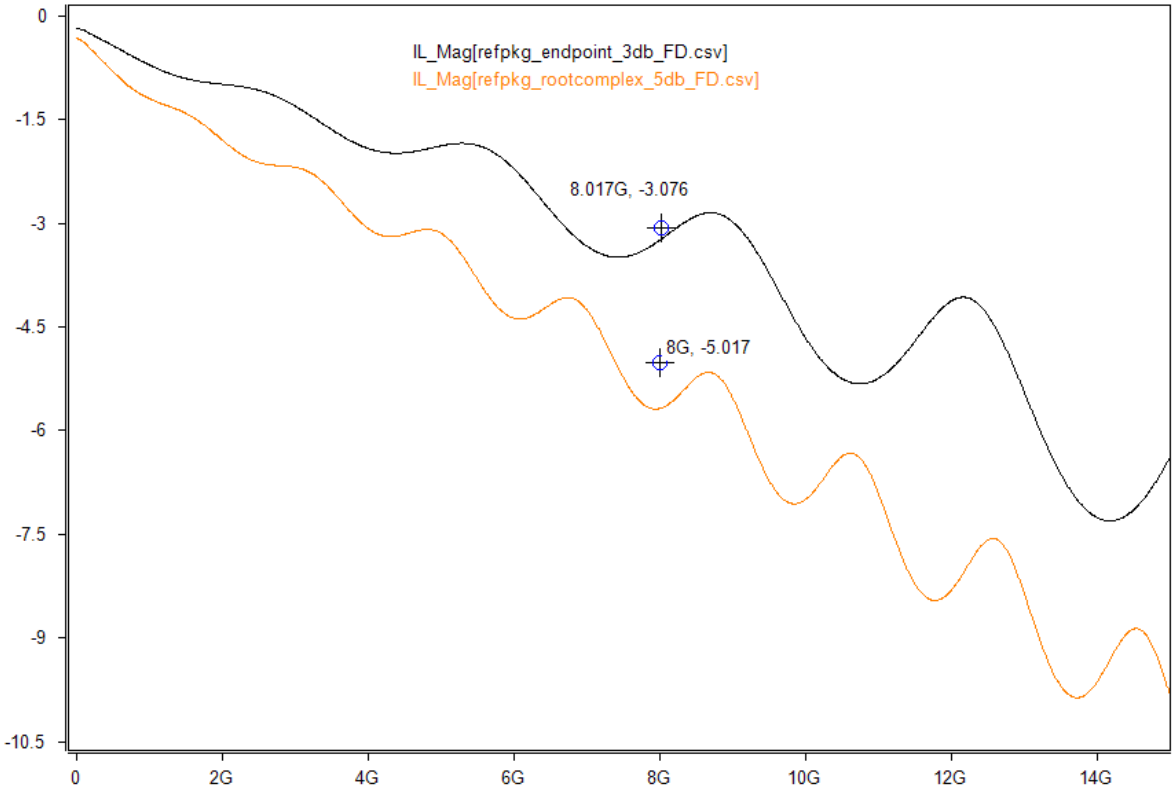


Figure 8-45: SDD21 Plots for Root and Non-Root Packages

8.5.1.2 *Measuring Package Performance (16.0 GT/s only)*

Some implementations (see section 8.3.3.11) are allowed to have packages that exceed reference packages in insertion loss and/or cross-talk. Actual package performance must be assessed by performing channel compliance with reference channels provided with the specification on the PCI-SIG website. A set of channel compliance simulations are run on the reference channels with one of the reference packages being replaced by the package that is being evaluated. If the eye height or eye width is smaller for any of the channels with the package that is being evaluated then the package is considered to have worse performance than the reference package. An implementation with a package that has worse performance than the reference package must use the implementation package model in the channel compliance methodology and may optionally use the implementation package in the Receiver stressed eye calibration. Note that form factor and channel compliance (for a captive channel) overall requirements still need to be met regardless of package characteristics.

8.5.1.3 *Simulation Tool Requirements*

Channel tolerancing is implemented by means of simulation, where the pass/fail criteria are defined in terms of a time domain eye diagram. The simulation tool must accept a prescribed set of inputs, including the channel under consideration and then simulate based upon a set of post processing requirements. The *PCI Express Base Specification* does not stipulate the use of any specific tool for simulating channels. However any simulation tool must meet the following requirements.

8.5.1.3.1 Simulation Tool Chain Inputs

❑Channel characteristics defined as s-parameters or equivalent model. The model must include the victim differential Lane plus as many aggressors as required to accurately capture crosstalk. In most cases this will be between 2 and 4 additional differential Lanes.

- ❑ Behavioral Root and Non-Root package models. The models will be included as part of the specification in the form of s-parameter files (see section 8.5.1.1).

❑Transmitter Jitter and voltage: The voltage and jitter parameters input to the simulator may be directly obtained from a combination of the Transmitter and Refclk jitter. Since these parameters are fixed the simulation tool may choose to hard code their values.

❑Transmitter and Receiver Termination Impedance: The simulator shall use a 2 x 50 Ω termination for both the Transmitter and Receiver. This value matches the assumptions which are implicit in generating and measuring the stressed eye for Rx tolerancing.

8.5.1.3.2 Processing Steps

❑ Time domain representation of the end-to-end connectivity: Included are the behavioral Tx and Rx packages and the channel under test.

❑ Tx voltage and jitter: Voltage and jitter parameters defined for the Transmitter, but have been recalculated to properly comprehend high and low frequency jitter components, and also include Refclk jitter contributions.

- ❑ Behavioral Transmitter Equalization: The simulator shall replicate the Transmitter equailzation capabilities defined in the Transmitter section.

❑ Behavioral Rx CTLE: The simulation tool shall implement a behavioral CTLE that replicates the CTLE function employed for Rx tolerancing.

- ❑ Behavioral DFE: The simulation tool shall implement a 1-tap (8.0 GT/s) or 2-tap (16.0 GT/s) DFE, where the dynamic range for the feedback coefficient is defined in Section 8.4.1.9.

❑ Optimizing Tx equalization and Rx DFE/CTLE settings: The simulation tool shall implement an optimization algorithm that selects the combination of Tx equalization and Rx CTLE and DFE settings that yields a maximum eye width at the far end of the channel. For details refer to Section 8.4.1.8.

❑ Statistical Treatment of jitter: In order to avoid overestimating the effect of channel-data and channel-jitter interactions, the tool shall use a statistical analysis of these parameters to generate voltage/jitter eye margins.

8.5.1.3.3 Simulation Tool Outputs

Output eye parameters: The simulator shall generate a statistically defined output that displays the eye width and eye height. EH will be measured as the peak eye height, while EW shall be measured at the zero crossing line. Additionally the simulator shall have the capability to adjust the data sample point by ±0.1 UI from the mean center of the UI.

8.5.1.3.4 Open Source Simulation Tool

An open source simulation tool shall be provided with the specification as design collateral. The tool will provide a turnkey capability, where the user provides the channel characteristics at the Receiver's die pad as step responses, and the tool calculates a statistical eye showing pass/fail.

8.5.1.4 *Behavioral Transmitter Parameters*

8.5.1.4.1 Deriving Voltage and Jitter Parameters

This section is for informative purposes. The voltage and jitter parameters may be derived from the Transmitter voltage and jitter parameters, but are referenced to the die pad. This is necessary to allow the channel simulation to include a behavioral Tx package and drive the package from the die pads. Additionally, the Tj terms must be decomposed into separate Rj and DjDD terms.

- $V_{TX\text{-}CH\text{-}FS\text{-}NO\text{-}EQ}$ and $V_{TX\text{-}CH\text{-}RS\text{-}NO\text{-}EQ}$: These two parameters define the minimum peak-peak voltage corresponding to Vd in Figure 8-5.
- The jitter parameters are derived based on the following set of equations. Algebraic manipulation is used to extract the Rj implicitly defined by the combination of Tj and DjDD terms. The following numbers are based on 8.0 GT/s Tx jitter parameters. The same approach is used to extract jitter parameters for 2.5, 5.0, and 16.0 GT/s.

$$jit_hfrj_nui = (T_{TX\text{-}UTJ} - T_{TX\text{-}UDJ\text{-}DD})/14 = 1.375\ ps$$
$$T_{TX\text{-}CH\text{-}UPW\text{-}RJ} = (T_{TX\text{-}UPWJ\text{-}TJ} - T_{TXUPWJ\text{-}DJDD})/14 = 1\ ps$$
$$T_{TX\text{-}CH\text{-}UPW\text{-}DJ} = T_{TXUPWJ\text{-}DJDD} = 10\ ps$$
$$T_{TX\text{-}CH\text{-}URJ} = sqrt(jit_hfrj_nui^{**}2 - (T_{TX\text{-}CH\text{-}UPW\text{-}RJ} {*}0.707)^{**}2 + T_{REFCLK\text{-}RMS}{}^{**}2) = 1.55\ ps$$
$$T_{TX\text{-}CH\text{-}UDJDD} = T_{TX\text{-}UDJ\text{-}DD} - (T_{TXUPWJ\text{-}DJDD})/2 = 7\ ps$$

A-0840

Figure 8-46: Derivation of 8.0 GT/s Jitter Parameters for Table 8-14

Values for 2.5 GT/s, 5.0 GT/s, and 16.0 GT/s jitter parameters in Table 8-14 will be added when the respective Tx jitter parameters become available.

A channel must be tested at all data rates, with the corresponding Tx jitter parameters, that it is intended to support during normal operation. For example, a channel intended to support a max data rate of 8.0 GT/s must be tested at 2.5, 5.0, and 8.0 GT/s.

Note: Values for the jitter parameters for 2.5 GT/s and 5.0 GT/s in Table 8-14 will be added once corresponding jitter parameter numbers are derived for the Tx section in Table 8-7:.

Table 8-14: Jitter/Voltage Parameters for Channel Tolerancing

Symbol	Parameter	Value	Units	Notes
$V_{TX\text{-}CH\text{-}FS\text{-}NO\text{-}EQ}$	Full swing Tx voltage	804	mVPP	Full swing, No Tx Eq.
$V_{TX\text{-}CH\text{-}RS\text{-}NO\text{-}EQ}$	Reduced swing Tx voltage	402	mVPP	Reduced swing, No Tx Eq.
2.5 GT/s Jitter Parameters and Voltage Parameters				
$T_{TX\text{-}CH\text{-}URJ\text{-}2.5G}$	Tx uncorrelated Rj	3.45	ps RMS	See Note 1
$T_{TX\text{-}CH\text{-}UDJDD\text{-}2.5G}$	Tx uncorrelated DjDD	20	ps PP	
$T_{TX\text{-}CH\text{-}UPW\text{-}RJ\text{-}2.5G}$	Uncorrelated PW Rj	1.42	ps RMS	See Note 2
$T_{TX\text{-}CH\text{-}UPW\text{-}DJ\text{-}2.5G}$	PW DDj	80	ps PP	
$T_{TX\text{-}DIEPAD\text{-}EDGERATE\text{-}2.5G}$	Signal edge rate at behavioral Tx die pad	140	ps	Measured 10% to 90% using a gaussian lowpass filter to shape the edge. See Note 3.
5.0 GT/s Jitter Parameters and Voltage Parameters				
$T_{TX\text{-}CH\text{-}URJ\text{-}5G}$	Tx uncorrelated Rj	3.45	ps RMS	See Note 1
$T_{TX\text{-}CH\text{-}UDJDD\text{-}5G}$	Tx uncorrelated DjDD	20-	ps PP	
$T_{TX\text{-}CH\text{-}UPW\text{-}RJ\text{-}5G}$	Uncorrelated PW Rj	1.42	ps RMS	
$T_{TX\text{-}CH\text{-}UPW\text{-}DJ\text{-}5G}$	PW DDj	40	ps PP	See Note 2.
$T_{TX\text{-}DIEPAD\text{-}EDGERATE\text{-}5G}$	Signal edge rate at behavioral Tx die pad	70	ps	Measured 10% to 90% using a gaussian lowpass filter to shape the edge. See Note 3.
8.0 GT/s Jitter Parameters and Voltage Parameters				
$T_{TX\text{-}CH\text{-}URJ\text{-}8G}$	Tx uncorrelated Rj	1.55	ps RMS	No DDj of HF jitter. See Note 1.
$T_{TX\text{-}CH\text{-}UDJDD\text{-}8G}$	Tx uncorrelated DjDD	7.0	ps PP	No DDj of HF jitter
$T_{TX\text{-}CH\text{-}UPW\text{-}RJ\text{-}8G}$	Uncorrelated PW Rj	1.0	ps RMS	
$T_{TX\text{-}CH\text{-}UPW\text{-}DJ\text{-}8G}$	PW DDj	10	ps PP	See Note 2.
$T_{TX\text{-}DIEPAD\text{-}EDGERATE\text{-}8G}$	Signal edge rate at behavioral Tx die pad	43.75	ps	Measured 10% to 90% using a gaussian lowpass filter to shape the edge. See Note 3.
16.0 GT/s Jitter Parameters and Voltage Parameters				
$T_{TX\text{-}CH\text{-}URJ\text{-}16G}$	Tx uncorrelated Rj	0.74	ps RMS	See Note 1.
$T_{TX\text{-}CH\text{-}UDJDD\text{-}16G}$	Tx uncorrelated DjDD	3.75	ps PP	
$T_{TX\text{-}CH\text{-}UPW\text{-}RJ\text{-}16G}$	Uncorrelated PW Rj	0.54	ps RMS	
$T_{TX\text{-}CH\text{-}UPW\text{-}DJ\text{-}16G}$	PW DDj	5.0	ps PP	See Note 2.
$T_{TX\text{-}DIEPAD\text{-}EDGERATE\text{-}16G}$	Signal edge rate at behavioral Tx die pad	21.875	ps	Measured 10% to 90% using a gaussian lowpass filter to shape the edge. See Note 3.

Notes:

1. Includes low frequency (non F/2) Rj components from the Transmitter and Rj from the Refclk.
2. Applied on a per edge basis as a dual Dirac model.
3. Does not include parasitic die pad capacitance. See Figure 8-43 for details of behavioral package.

8.5.1.5 *Optimizing Tx/Rx Equalization (8.0 GT/s and 16.0 GT/s only)*

The algorithm for optimizing Tx/Rx equalization shall operate as follows. For every combination of CTLE and Transmitter equalization coefficients calculate the zero forced DFE and use Peak Distortion Analysis to find maximum eye area as figure of merit within the constraints set by the ±0.1 UI aperture and ±30 mV d_1 and ±20 mV d_2 (16.0 GT/s only) DFE coefficient limits.

8.5.1.6 *Pass/Fail Eye Characteristics*

The output of the simulation tool shall be in the form of pass/fail characteristics as defined by an eye mask as shown in Figure 8-47. EH and EW must meet respectively the voltage and jitter parameters defined in Table 8-15. Eye margins are defined at the die pad of the Receiver after the appropriate Tx and Rx equalization algorithms have been applied.

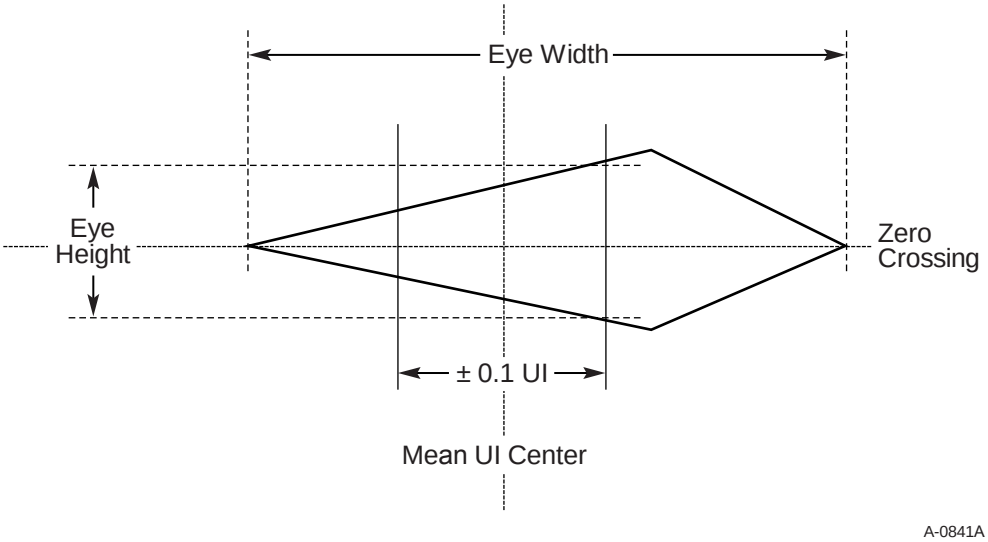


Figure 8-47: EH, EW Mask

Note that the pass/fail EH and EW limits shown in Figure 8-47 are identical to the limits defined for Rx testing in Table 8-10 for 8 and 16.0 GT/s. For 2.5 and 5.0 GT/s the limits for Rx testing are referenced to the package pins and the limits for channel tolerancing in this table are referenced to the Receiver pad after applying the same reference package used for 8.0 GT/s channel tolerancing.

Table 8-15: Channel Tolerancing Eye Mask Values

Symbol	Parameter	Value	Units	Comments
2.5 GT/s Eye Margins				
$V_{RX\text{-}CH\text{-}EH\text{-}2.5G}$	Eye height	<130 (min)	mVPP	Eye height at BER=10^{-12}. Note 1
$T_{RX\text{-}CH\text{-}EW\text{-}2.5G}$	Eye width at zero crossing	<0.35 (min)	UI	Eye width at BER=10^{-12}
$T_{RX\text{-}DS\text{-}OFFSET\text{-}2.5G}$	Peak EH offset from UI center	±0.1	UI	See Figure 8-47 for details.
5.0 GT/s Eye Margins				
$V_{RX\text{-}CH\text{-}EH\text{-}5G}$	Eye height	<85 (min)	mVPP	Eye height at BER=10^{-12}. Note 1
$T_{RX\text{-}CH\text{-}EW\text{-}5G}$	Eye width at zero crossing	<0. 30 (min)	UI	Eye width at BER=10^{-12}
$T_{RX\text{-}DS\text{-}OFFSET\text{-}5G}$	Peak EH offset from UI center	±0.1	UI	See Figure 8-47 for details.
8.0 GT/s Eye Margins				
$V_{RX\text{-}CH\text{-}EH\text{-}8G}$	Eye height	25 (min)	mVPP	Eye height at BER=10^{-12}. Note 1.
$T_{RX\text{-}CH\text{-}EW\text{-}8G}$	Eye width at zero crossing	0.3 (min)	UI	Eye width at BER=10^{-12}
$T_{RX\text{-}DS\text{-}OFFSET\text{-}8G}$	Peak EH offset from UI center	±0.1	UI	See Figure 8-47 for details.
$V_{RX\text{-}DFE\text{-}D1\text{-}8G}$	Range for DFE d_1 coefficient	±30	mV	
16.0 GT/s Eye Margins				
$V_{RX\text{-}CH\text{-}EH\text{-}16G}$	Eye height	15 (min)	mVPP	Eye height at BER=10^{-12}. Note 1.
$T_{RX\text{-}CH\text{-}EW\text{-}16G}$	Eye width at zero crossing	0.3 (min)	UI	Eye width at BER=10^{-12}
$T_{RX\text{-}DS\text{-}OFFSET\text{-}16G}$	Peak EH offset from UI center	±0.1	UI	See Figure 8-47 for details.
$V_{RX\text{-}DFE\text{-}D1\text{-}16G}$	Range for DFE d_1 coefficient	±30	mV	
$V_{RX\text{-}DFE\text{-}D2\text{-}16G}$	Range for DFE d_2 coefficient	±20	mV	

Notes:

1. $V_{RX\text{-}CH\text{-}EH}$ is defined as max EH within an aperture of ±0.1 UI from mean UI center.

8.5.1.7 *Characterizing Channel Common Mode Noise*

A channel must meet the common mode requirements as they are defined in the receiver specification. In general, it is not possible to accurately simulate all the channel's common mode noise contributions due to the large number of mechanisms that can generate CM noise, including the Transmitter. Typically channel common mode noise is a budgeted parameter, and the limits defined below assume a budgeting process. The channel's CM limit is defined as the amount of CM noise that a channel can add and still meet the Rx CM limits assuming the worst case Tx CM. This limit is 75 mVPP for EH < 100 mV and 125 mVPP for EH ≥ 100 mVPP.

Note that the Tx and channel CM noise parameters cannot simply be added to obtain the Rx CM limit. This is due to the fact that a channel will attenuate some of high frequency Tx CM noise while propagating Tx LF CM noise through with little loss. The channel may also contribute both high and low frequency CM components of its own.

8.5.1.8 *Verifying $V_{CH\text{-}IDLE\text{-}DET\text{-}DIFFp\text{-}p}$*

$V_{CH\text{-}IDLE\text{-}DET\text{-}DIFFp\text{-}p}$ is defined to guarantee that, when a Transmitter issues an EIEOS sequence, the Receiver is guaranteed to detect it. Potentially larger Transmitter equalization boost ratios at 8.0 and 16.0 GT/s necessitate that this parameter be verified; this procedure was not necessary for 2.5 or 5.0 GT/s, where the max Transmitter equalization boost is smaller. Defining the launch and detect voltages at the Tx/Rx die pad permits $V_{CH\text{-}IDLE\text{-}DET\text{-}DIFF\text{-}PP}$ to be verified with the same channel and Tx/Rx package models used to determine eye margins. It is also acceptable to simulate from Tx pin to Rx pin (excluding the Tx and Rx behavioral package models), in which case the EIEOS and idle detect parameters defined in the Tx and Rx sections are applicable.

Long channels, where $V_{TX\text{-}EIEOS\text{-}FS}$ is applicable, are characterized by driving the channel under test with the EIEOS pattern and -11.0 dB de-emphasis and zero dB preshoot. For short channels, where $V_{TX\text{-}EIEOS\text{-}RS}$ is applicable, -4.5 dB of de-emphasis and zero dB of preshoot are applied.

Table 8-16: EIEOS Signaling Parameters

Parameter	Description	Value	Units	Comments
$V_{CH\text{-}IDLE\text{-}EXIT\text{-}pp}$	Idle detect voltage seen at the Rx die pad	172	mVPP	Assuming Rx R_{TERM} of 2 x 50 Ω
$V_{CH\text{-}EIEOS\text{-}FS\text{-}Vb}$	PP voltage during Vb interval at behavioral Tx die pad for full swing signaling	255	mVPP	Assuming Tx R_S of 2 x 50 Ω
$V_{CH\text{-}EIEOS\text{-}RS\text{-}Vb}$	PP voltage during Vb interval at behavioral Tx die pad for reduced swing signaling	237	mVPP	Assuming Tx R_S of 2 x 50 Ω

8.6 Refclk Specifications

This version of the specification consolidates and streamlines the Refclk requirements. The 2.5 GT/s Refclk parameters are moved from the CEM spec to this spec so that all Refclk parameters four data rates: 2.5, 5.0, 8.0 and 16.0 GT/s are now contained in this subsection.

8.6.1 Refclk Test Setup

The test setup for the Refclk assumes that only the Refclk generator itself is present. Provision is made in the test setup to account for signal degradation that occurs between the pins of the Refclk generator and the Transmitter or Receiver in an actual system. The above described setup emulates the worst case signal degradation that is likely to occur at the pins of a PCI Express device. Note that the Refclk signal is tested into a load that represents the series (open) termination appearing at the Refclk input pins of a PCIe device.

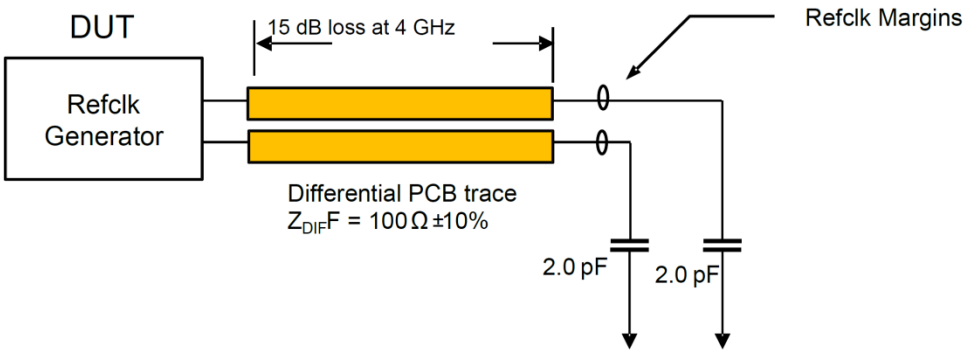


Figure 8-48: Refclk Test Setup

8.6.2 REFCLK AC Specifications

All specifications in Table 8-17 are to be measured using a test configuration as described in Note 11 with a circuit as shown in Figure 8-48.

Table 8-17: REFCLK DC Specifications and AC Timing Requirements

Symbol	Parameter	100 MHz Input		Unit	Note
		Min	Max		
Rising Edge Rate	Rising Edge Rate	0.6	4.0	V/ns	2, 3
Falling Edge Rate	Falling Edge Rate	0.6	4.0	V/ns	2, 3
V_{IH}	Differential Input High Voltage	+150		mV	2
V_{IL}	Differential Input Low Voltage		-150	mV	2
V_{CROSS}	Absolute crossing point voltage	+250	+550	mV	1, 4, 5
$V_{CROSS\ DELTA}$	Variation of V_{CROSS} over all rising clock edges		+140	mV	1, 4, 9
V_{RB}	Ring-back Voltage Margin	-100	+100	mV	2, 12
T_{STABLE}	Time before V_{RB} is allowed	500		ps	2, 12
$T_{PERIOD\ AVG}$	Average Clock Period Accuracy	-300	+2800	ppm	2, 10, 13
$T_{PERIOD\ ABS}$	Absolute Period (including Jitter and Spread Spectrum modulation)	9.847	10.203	ns	2, 6
$T_{CCJITTER}$	Cycle to Cycle jitter		150	ps	2
V_{MAX}	Absolute Max input voltage		+1.15	V	1, 7
V_{MIN}	Absolute Min input voltage		- 0.3	V	1, 8
Duty Cycle	Duty Cycle	40	60	%	2
Rise-Fall Matching	Rising edge rate (REFCLK+) to falling edge rate (REFCLK-) matching		20	%	1, 14
$Z_{C\text{-}DC}$	Clock source DC impedance	40	60	Ω	1, 11

Notes:

1. Measurement taken from single ended waveform.
2. Measurement taken from differential waveform.

3. Measured from -150 mV to +150 mV on the differential waveform (derived from REFCLK+ minus REFCLK-). The signal must be monotonic through the measurement region for rise and fall time. The 300 mV measurement window is centered on the differential zero crossing. See Figure 8-53.
4. Measured at crossing point where the instantaneous voltage value of the rising edge of REFCLK+ equals the falling edge of REFCLK-. See Figure 8-49.
5. Refers to the total variation from the lowest crossing point to the highest, regardless of which edge is crossing. Refers to all crossing points for this measurement. See Figure 8-49.
6. Defines as the absolute minimum or maximum instantaneous period. This includes cycle to cycle jitter, relative PPM tolerance, and spread spectrum modulation. See Figure 8-52.
7. Defined as the maximum instantaneous voltage including overshoot. See Figure 8-49.
8. Defined as the minimum instantaneous voltage including undershoot. See Figure 8-49.
9. Defined as the total variation of all crossing voltages of Rising REFCLK+ and Falling REFCLK-. This is the maximum allowed variance in V_{CROSS} for any particular system. See Figure 8-50.
10. Refer to Section 4.3.7.1.1 of the *PCI Express Base Specification, Revision 3.0* for information regarding PPM considerations.
11. System board compliance measurements must use the test load card described in Figure 2-9. REFCLK+ and REFCLK- are to be measured at the load capacitors C_L. Single ended probes must be used for measurements requiring single ended measurements. Either single ended probes with math or differential probe can be used for differential measurements. Test load C_L = 2 pF.
12. T_{STABLE} is the time the differential clock must maintain a minimum ±150 mV differential voltage after rising/falling edges before it is allowed to droop back into the V_{RB} ±100 mV differential range. See Figure 8-54.
13. PPM refers to parts per million and is a DC absolute period accuracy specification. 1 PPM is 1/1,000,000th of 100.000000 MHz exactly or 100 Hz. For 300 PPM, then we have an error budget of 100 Hz/PPM * 300 PPM = 30 kHz. The period is to be measured with a frequency counter with measurement window set to 100 ms or greater. The ±300 PPM applies to systems that do not employ Spread Spectrum Clocking, or that use common clock source. For systems employing Spread Spectrum Clocking, there is an additional 2,500 PPM nominal shift in maximum period resulting from the 0.5% down spread resulting in a maximum average period specification of +2,800 PPM.
14. Matching applies to rising edge rate for REFCLK+ and falling edge rate for REFCLK-. It is measured using a ±75 mV window centered on the median cross point where REFCLK+ rising meets REFCLK- falling. The median cross point is used to calculate the voltage thresholds the oscilloscope is to use for the edge rate calculations. The Rise Edge Rate of REFCLK+ should be compared to the Fall Edge Rate of REFCLK-; the maximum allowed difference should not exceed 20% of the slowest edge rate. See Figure 8-51.

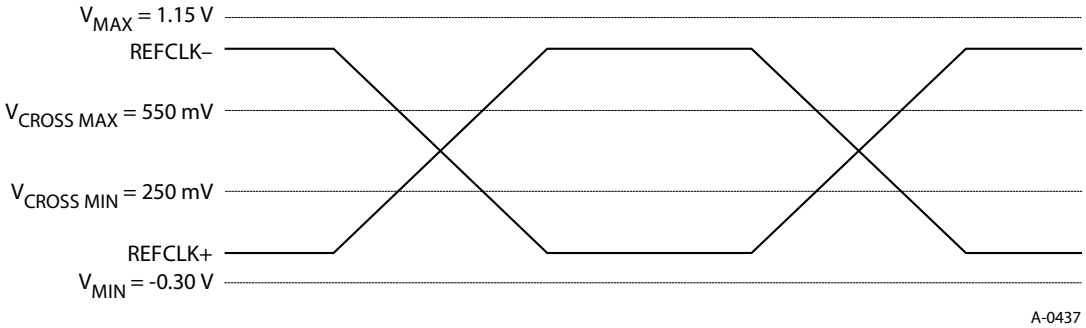


Figure 8-49: Single-Ended Measurement Points for Absolute Cross Point and Swing

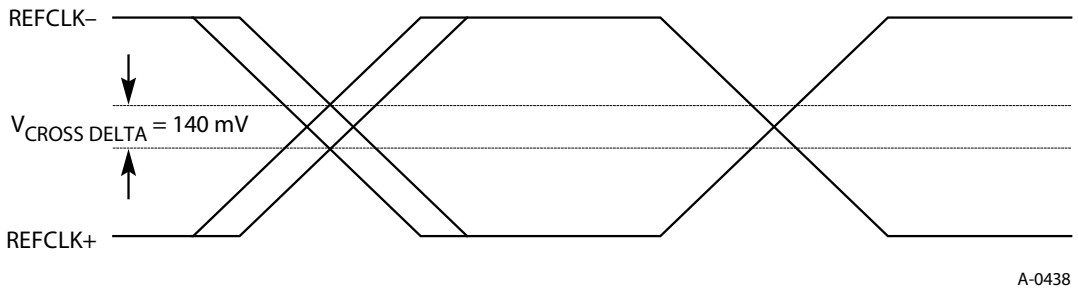


Figure 8-50: Single-Ended Measurement Points for Delta Cross Point

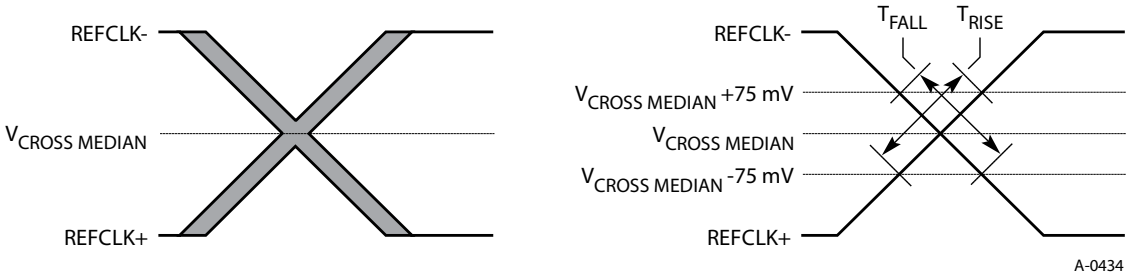


Figure 8-51: Single-Ended Measurement Points for Rise and Fall Time Matching

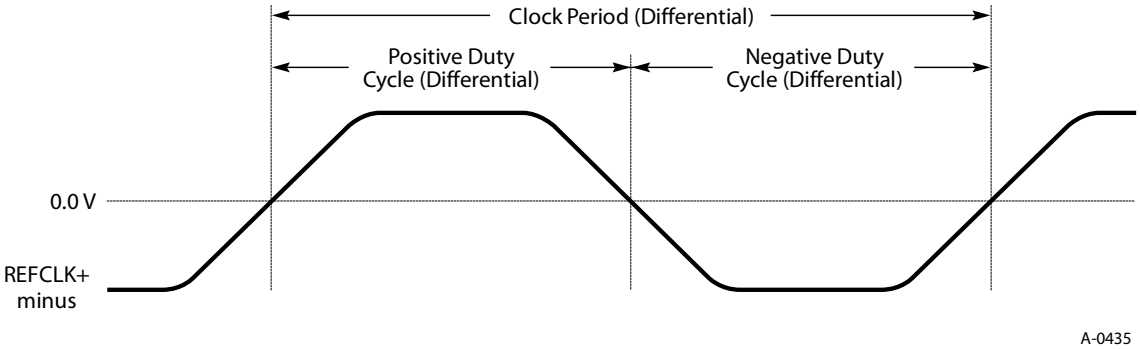


Figure 8-52: Differential Measurement Points for Duty Cycle and Period

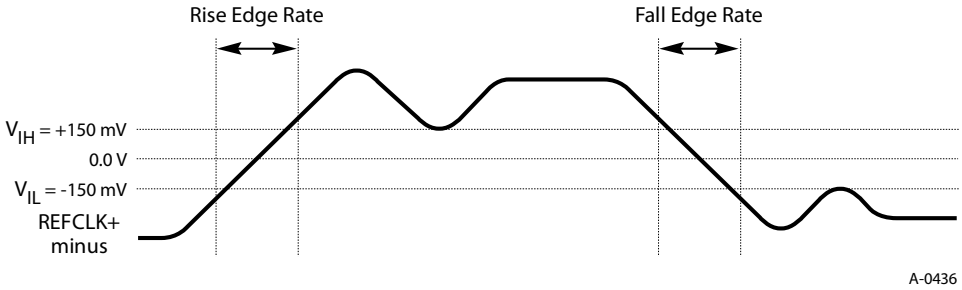


Figure 8-53: Differential Measurement Points for Rise and Fall Time

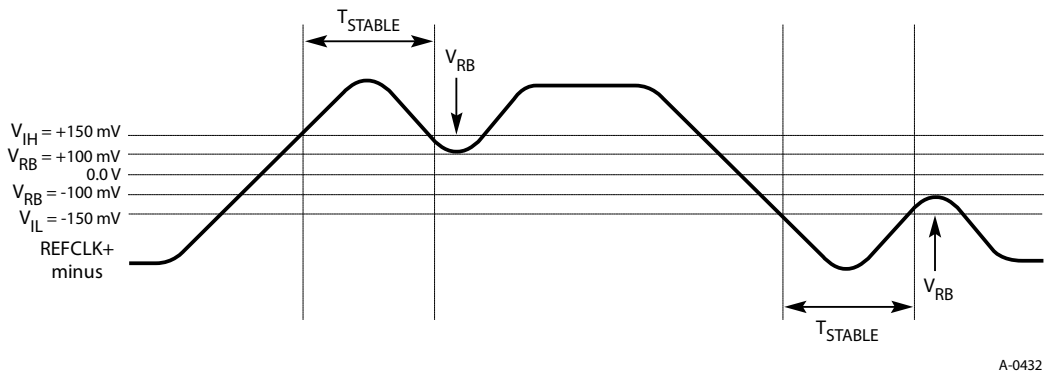


Figure 8-54: Differential Measurement Points for Ringback

8.6.3 Data Rate Independent Refclk Parameters

A number of Refclk parameters are data rate independent and are listed in the table below. $T_{TRANSPORT_DELAY}$ is defined in Section 8.6.6 and illustrated in Figure 8-57. It is relevant only for the Common Refclk architecture. For the SRIS mode the source of the SSC modulation is implementation dependent.

Table 8-18: Data Rate Independent Refclk Parameters

Symbol	Description	Limits	Units	Notes
F_{REFCLK}	Refclk Frequency	99.97 (min), 100.03 (max)	MHz	
F_{SSC}	SSC frequency range	30 (min), 33 (max)	kHz	3
$T_{SSC\text{-}FREQ\text{-}DEVIATION}$	SSC deviation	-0.5 (min), 0.0 (max)	%	3
$T_{TRANSPORT\text{-}DELAY}$	Tx-Rx transport delay	12 (max)	ns	1, 4
$T_{SSC\text{-}MAX\text{-}FREQ\text{-}SLEW}$	Max SSC df/dt	1250	ppm/us	2, 3

Notes:

1. Parameter is relevant only for common Refclk architecture.
2. Measurement is made over 0.5µsec time interval with an 1st order LPF with an f_c of 60x the modulation frequency.
3. When testing the a device configured for the IR reference clock architecture the SSC related parameters must be tested with the Tx output data instead of the reference clock.
4. There are form factors (for example topologies including long cables) that may exceed the transport delay limit. Extra jitter from the large transport delay must be accounted by these form factor specifications.

8.6.3.1 *Low Frequency Refclk Jitter Limits*

Refclks supporting SSC must meet an additional jitter limit over a range of low frequencies. Low frequency Refclk jitter limits are defined as a continuous, piece-wise linear graph from 30 kHz to 500 kHz as shown below. Unfiltered Refclk phase jitter must fall below this graph over the frequency range of interest.

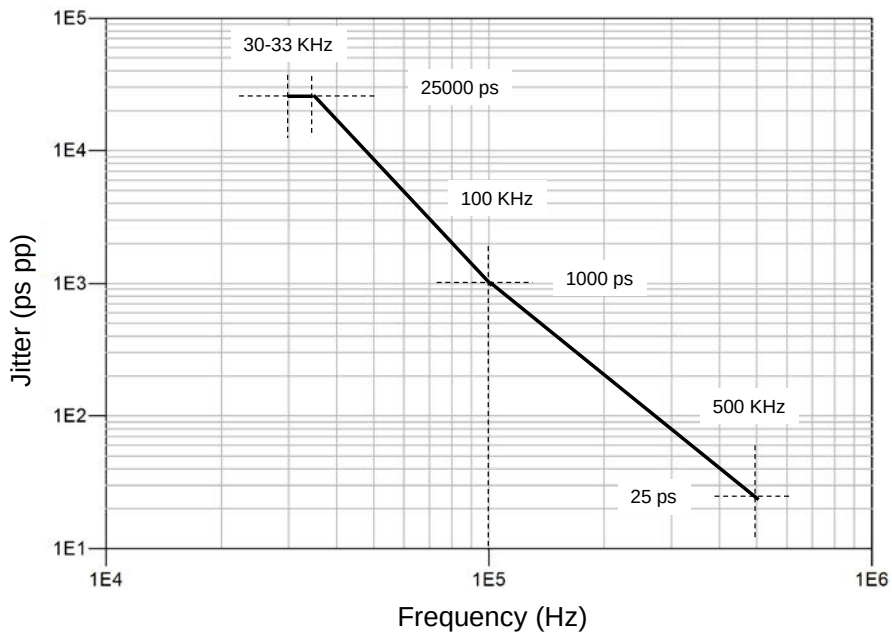


Figure 8-55: Limits for phase jitter from the Reference

8.6.4 Refclk Architectures Supported

Two Refclk architectures are supported: Common Refclk (CC) and Independent Refclk (IR). The CC clock architecture is described in terms of its topology and its corresponding jitter transfer function based on PLL and CDR characteristics. Finally, the corresponding Refclk jitter limits are given for each data rate. The jitter transfer function and corresponding jitter limits are not defined for the IR clock architecture. It is up to the implementer to trade off reference clock jitter and PLL characteristics to ensure that Transmitter requirements are met in the IR clocking mode.

8.6.5 Filtering Functions Applied to Raw Data

Two types of filtering are applied to the raw Refclk data. The first, edge filtering, minimizes the measurement-induced jitter due to the finite sampling rate of the test equipment. The second, replicates the jitter filtering that is inherent in the combination of Tx/Rx PLLs, the Rx CDR and (where applicable) the transport delay. The combination of the preceding filter functions yields the effective Refclk jitter as it appears at the sample latch of the Receiver.

Note that the PLL and CDR filter functions represent minimally capable approximations to actual Receiver implementations and are not intended to define actual PLL or CDR implementations.

8.6.5.1 *PLL Filter Transfer Function Example*

All PLLs are behaviorally modeled with a second order transfer function, H(s), as defined in Figure 8-57. The parameters defining the transfer function include the damping factor ζ and the natural frequency ω_n.

The relation between the 2nd order PLL (lowpass) natural frequency, ω_n and the 3 dB point ω_{3dB}, is given by the following expression:

$$\frac{\omega_{3dB}}{\omega_n} = \sqrt{\sqrt{(2\zeta^2+1)^2+1} + 2\zeta^2 + 1}$$

Equation 8-6

The following plot of a 2nd order PLL illustrates the transfer function with an f_{3dB} of 5.0 MHz and 1.0 dB of peaking. This corresponds to ζ = 1.15, and ω_n = 11.55 Mrad/sec.

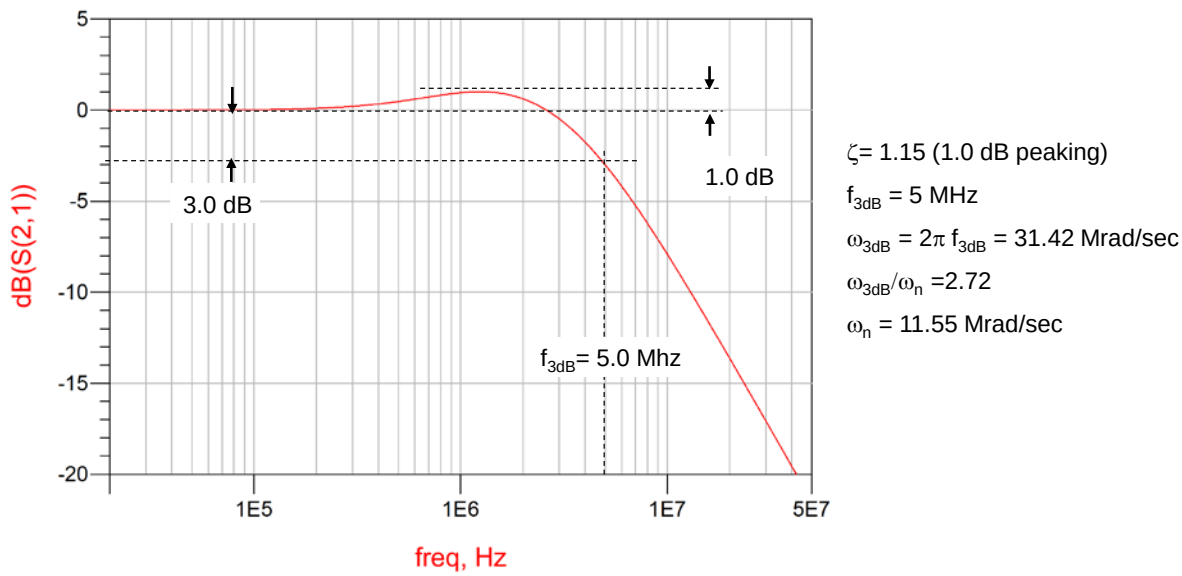


Figure 8-56: 5 MHz PLL Transfer Function Example

8.6.5.2 *CDR Transfer Function Examples*

Depending on the Refclk architecture, either a first or second order transfer function shall be used as a behavioral CDR bounding limit. CC assumes a 1st order transfer function, while IR assumes a 2nd order CDR. In both cases the CDR transfer function may be represented as a highpass filter defined by ω_{3dB} for the 1st order case and by ω_{3db}, ζ and ω_n for 2nd order. Knowing two of the three previous parameters for 2nd order permits the third to be calculated, and converting between radians and Hz is simply: $\omega_{3dB} = 2\pi f_{3dB}$.

For behavioral CDR functions refer to Section 8.3.5.5

8.6.6 Common Refclk Rx Architecture (CC)

This architecture utilizes a single Refclk source that is distributed to both the Tx and Rx. Most of the SSC jitter sourced by the Refclk is propagated equally through Tx and Rx PLLs, and so intrinsically tracks LF jitter. This is particularly true for SSC which tends to be low frequency. Figure 8-57 illustrates the common Refclk Rx architecture, showing key jitter, delay, and PLL and CDR transfer function sources. The amount of jitter appearing at the CDR is then defined by the difference function between the Tx and Rx PLLs multiplied by the CDR highpass characteristic.

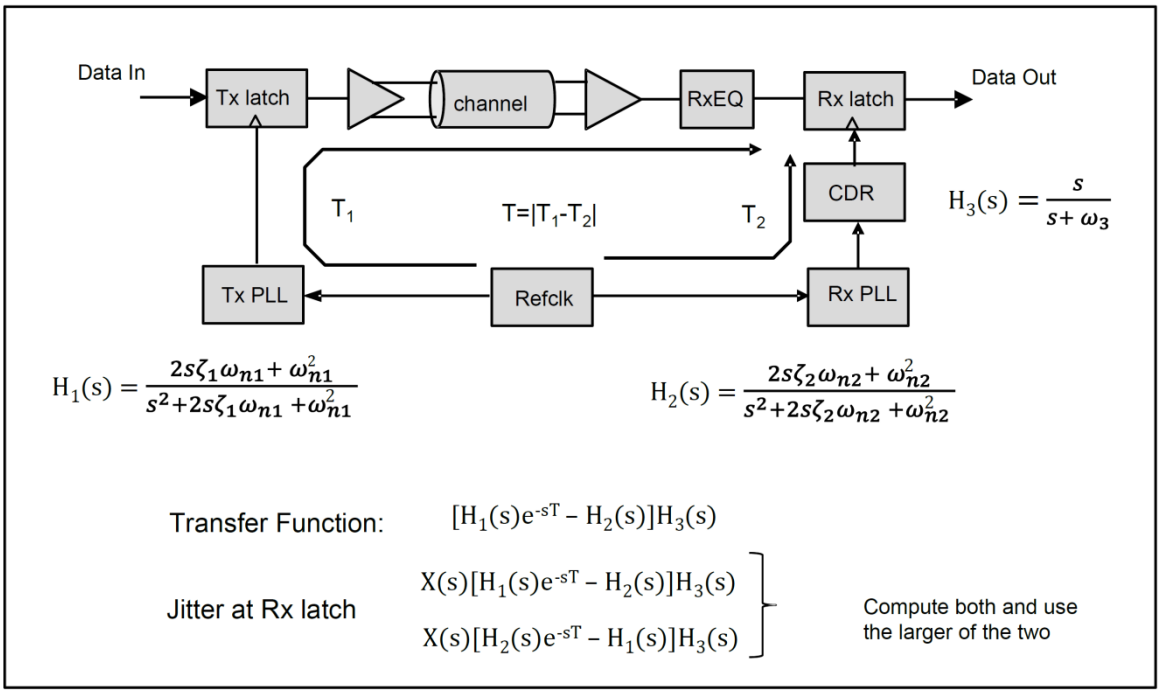


Figure 8-57: Common Refclk Rx Architecture

Based on the above clock architecture, it is possible to define a difference function that corresponds to the worst case mismatch between Tx and Rx PLLs. Second order PLL transfer functions are assumed, (even though most PLL transfer functions are 3rd order or higher), since a 2nd order function tends to yield a slightly conservative difference function vis-a-vis most actual PLL implementations. For CC the CDR is assumed to be first order. Actual CDRs of higher order will tend to filter more low frequency jitter, so assuming a 1st order transfer function will yield slightly conservative jitter results in these cases.

In the common Refclk Rx architecture it is also necessary to comprehend a maximum Transmitter to Receiver transport delay difference. This delay delta is illustrated in Figure 8-57 and represents the delay difference between the Transmitter data and recovered Receiver clock as seen at the inputs to the receive's data latch.

8.6.6.1 *Determining the Number of PLL BW and peaking Combinations*

A Tx or Rx PLL is defined by the combination of min/max bandwidth and peaking, making for a total of four possible limits. In the CC architecture both the Tx and Rx PLLs contribute to the jitter transfer function. At 2.5 GT/s only one set of BW/peaking limits is defined. If the combinations for the Tx and Rx PLLs limits are defined by the sets (A_{TX}, B_{TX}, C_{TX}, D_{TX}), (A_{RX}, B_{RX}, C_{RX}, D_{RX}) then it's easily demonstrated that there are a total of ten ways of selecting one element from each set. The delay term e^{-ST} can be applied to either $H_1(s)$ or $H_2(s)$, adding another six possibilities. Only six, as opposed to ten, terms are added when the delay term is considered because terms like A_{TX}, A_{RX}, are identical to A_{RX}, A_{TX}.

At 5.0 GT/s and higher data rates, two possible sets of limits of PLL bandwidth and peaking are defined, which may be defined by the sets (A_{TX}, B_{TX}, C_{TX}, D_{TX}) and (E_{RX}, F_{RX}, G_{RX}, H_{RX}). In this

case the number of unique 2-element combinations from the above 4-element sets is 36, which increases to 64 when the delay term is considered.

8.6.6.2 *CDR and PLL BW and Peaking Limits for Common Refclk*

The common Refclk architecture filter function is dependent on the difference function defined by the BW and peaking of Tx and the Rx PLLs, plus the CDR high-pass characteristic. It is necessary to consider all corner case combinations of Tx and Rx PLL peaking and bandwidth plus the CDR characteristics at their minimum and maximum peaking values.

This procedure must be applied to all four data rates.

Table 8-19 lists the PLL BW and CDR BW/peaking values that need to be applied as filter functions for 2.5 GT/s data rates. It is necessary to assign a min and a max value for peaking for the 2.5 GT/s PLL. The values of 0.01 dB and 3.0 dB represent best estimates of realistic PLL implementations.

The minimum peaking for 2.5 and 5.0 GT/s has been reduced to 0.01 dB to bring it into line with the 8.0 and 16.0 GT/s cases. Note that the Rx CDR is 1st order, so its natural frequency, ω_n can be directly obtained from its BW, unlike the 2nd order CDRs, where ω_n is a function of both BW and peaking.

Table 8-19: Common Refclk PLL and CDR Characteristics for 2.5 GT/s

PLL #1, PLL #2	0.01 dB peaking	3.0 dB peaking
BW_{PLL}(min) = 1.5 MHz	ω_{n1}= .336 Mrad/s ζ_1 = 14	ω_{n1}= 5.09 Mrad/s ζ_1 = 0.54
BW_{PLL}(max) = 22 MHz	ω_{n1}= 4.93 Mrad/s ζ_1 = 14	ω_{n1}= 74.68 Mrad/s ζ_1 = 0.54

BW_{CDR}(min) = 1.5 MHz, 1st order **CDR**

16 combinations

2.5 GT/s

PLL and CDR jitter and peaking characteristics for 5.0, 8.0, and 16.0 GT/s yield a larger number of possible combinations because two sets of PLL BW and peaking limits are given. This choice to support two sets of BW and peaking was made to give designers as much latitude as possible when designing PLL circuits.

Table 8-20: Common Refclk PLL and CDR Characteristics for 5.0 GT/s

PLL #1	0.01 dB peaking	1.0 dB peaking
BW_{PLL}(min) = 5.0 MHz	ω_{n1}= 1.12 Mrad/s ζ_1 = 14	ω_{n1}= 11.01 Mrad/s ζ_1 = 1.16
BW_{PLL}(max) = 16 MHz	ω_{n1}= 3.58 Mrad/s ζ_1 = 14	ω_{n1}= 35.26 Mrad/s ζ_1 = 1.16

PLL #2	0.01 dB peaking	3.0 dB peaking
BW_{PLL}(min) = 8.0 MHz	ω_{n2}= 1.79 Mrad/s ζ_2 = 14	ω_{n2}= 26.86 Mrad/s ζ_2 = 0.54
BW_{PLL}(max) = 16 MHz	ω_{n2}= 3.58 Mrad/s ζ_2 = 14	ω_{n2}= 53.73 Mrad/s ζ_2 = 0.54

BW_{CDR}(min) = 5 MHz, 1st order

64 combinations

5 GT/s

Table 8-21: Common Refclk PLL and CDR Characteristics for 8.0 and 16.0 GT/s

PLL #1	0.01 dB peaking	2.0 dB peaking
BW_{PLL}(min) = 2.0 MHz	ω_{n1}= 0.448 Mrad/s ζ_1 = 14	ω_{n1}= 6.02 Mrad/s ζ_1 = 0.73
BW_{PLL}(max) = 4.0 MHz	ω_{n1}= 0.896 Mrad/s ζ_1 = 14	ω_{n1}= 12.04 Mrad/s ζ_1 = 0.73

PLL #2	0.01 dB peaking	1.0 dB peaking
BW_{PLL}(min) = 2.0 MHz	ω_{n2}= 0.448 Mrad/s ζ_2 = 14	ω_{n2}= 4.62 Mrad/s ζ_2 = 1.15
BW_{PLL}(max) = 5.0 MHz	ω_{n2}= 1.12Mrad/s ζ_2 = 14	ω_{n2}= 11.53 Mrad/s ζ_2 = 1.15

BW_{CDR}(min) = 10 MHz, 1st order	64 combinations	8.0, 16.0 GT/s

8.6.7 Jitter Limits for Refclk Architectures

Table 8-22 lists the jitter limits for the CC Refclk architecture at each of the four data rates

Jitter at 2.5 GT/s is measured as a peak to peak jitter value, because a substantial proportion of the jitter is SSC harmonics which appears at the receiver as Dj. The combination of the 2.5 GT/s PLL and CDR bandwidths passes a significant amount of SSC residual, where it appears Dj. The 86 ps number is the same as that specified in the 3.0 CEM spec.

For 5.0, 8.0, and 16.0 GT/s jitter is specified as an RMS (Rj) value. These signaling speeds utilize a lower PLL BW and a higher CDR BW, and the effect is to suppress SSC harmonics such that almost all the jitter appears as Rj.

Table 8-22: Jitter Limits for CC Architecture

Data Rate	CC jitter Limit	Notes
2.5 GT/s	86 ps pp	1, 2
5.0 GT/s	3.1 ps RMS	1, 2
8.0 GT/s	1.0 ps RMS	1, 2
16.0 GT/s	≤ 0.5 ps RMS	1, 2, 3, 4

Notes:

1. The Refclk jitter is measured after applying the filter function in Figure 8-57
2. Jitter measurements shall be made with a capture of at least 100,000 clock cycles captured by a real time oscilloscope with a sample rate of 20 GS/s or great. Broadband oscilloscope noise must be minimized in the measurement. The measured PP jitter is used (no extrapolation).
3. For the 16.0 GT/s CC measurement SSC spurs from the fundamental and harmonics are removed up to a cutoff frequency of 2 MHz taking care to minimize removal of any non-SSC content.
4. Note that 0.7 ps RMS is to be used in channel simulations to account for additional noise in a real system.

8.6.8 Form Factor Requirements for RefClock Architectures

Each form factor specification must include the following table providing a clear summary of the clocking architecture requirements for devices that support the form factor specification. For each clocking architecture the table indicates whether that architecture is required, optional, or not allowed for this form factor. Note that this refers to the operation of the device – not the underlying silicon capabilities.

Table 8-23: Form Factor Clocking Architecture Requirements

Clock Architecture	System Board (Motherboard)	Add-in Card (Module)	Retimer
Common			
SRNS			
SRIS			

If the Common Reference Clock architecture is required or optional for the form factor, then there must be an additional table providing details for the common clock. Each entry in the table is marked required, optional, not allowed, or NA. "Clock Source" indicates the source of the common reference clock, if applicable. "SSC" indicates whether the clock source is spread and "CLKREQ#" indicates whether the CLKREQ# signal must be implemented.

Table 8-24: Common Clock Architecture Details

Common Clock Details	System Board (Motherboard)	Add-in Card (Module)	Retimer
Clock Source			
SSC			
CLKREQ#			

As an example the populated tables are shown for a hypothetical form factor that requires all components use the common clock architecture and does not allow the use of any other clocking architecture. The common clock source is required to be provided by the motherboard component and may optionally have SSC. CLKREQ# is not defined as a connector signal for this example form factor.

Table 8-25: Form Factor Clocking Architecture Requirements Example

Clock Architecture	System Board (Motherboard)	Add-in Card (Module)	Retimer
Common	Required	Required	Required
SRNS	Not Allowed	Not Allowed	Not Allowed
SRIS	Not Allowed	Not Allowed	Not Allowed

Table 8-26: Common Clock Architecture Details Example

Common Clock Details	System Board (Motherboard)	Add-in Card (Module)	Retimer
Clock Source	Required	Not Allowed	Not Allowed
SSC	Optional	N/A	N/A
CLKREQ#	N/A	N/A	N/A

If a form factor has clocking requirements that can not be provided in this simple one or two table form then careful consideration must be given to ensure that the form factor requirements are supported by this specification.

9

9 Single Root I/O Virtualization and Sharing

9.1 Architectural Overview

Within the industry, significant effort has been expended to increase the effective hardware resource utilization (i.e., application execution) through the use of virtualization technology. Single Root I/O Virtualization and Sharing (SR-IOV) consists of extensions to the PCI Express (PCIe) specification suite to enable multiple System Images (SI) to share PCI hardware resources.

To illustrate how this technology can be used to increase effective resource utilization, consider the generic platform configuration illustrated in Figure 9-1.

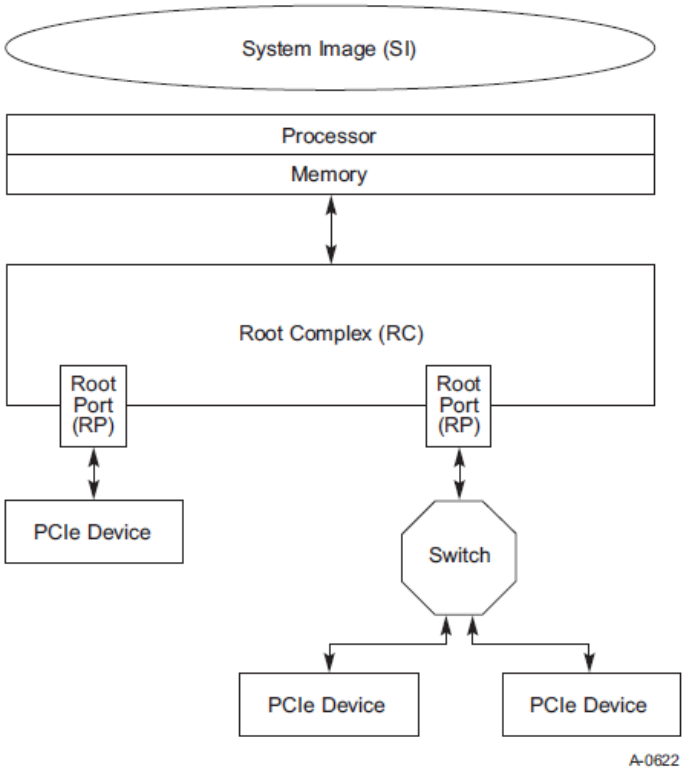


Figure 9-1: Generic Platform Configuration

The generic platform configuration is composed of the following components:

- Processor – general purpose, embedded, or specialized processing element
- Memory – general purpose or embedded
- PCIe Root Complex (RC) – one or more RC can be supported per platform

- PCIe Root Port (RP) – one or more RP can be supported per RC. Each RP represents a separate hierarchy. Each hierarchy is referred to a single root hierarchy to delineate it from the multiple hierarchy technology defined within the *Multi Root I/O Virtualization Specification*.
- PCIe Switch – provides I/O fan-out and connectivity
  - PCIe Device – multiple I/O device types, (e.g., network, storage, etc.)
  - System Image – software such an operating system that is used to execute applications or trusted services, (e.g., a shared or non-shared I/O device driver.)

In order to increase the effective hardware resource utilization without requiring hardware modifications, multiple SI can be executed. Software termed a Virtualization Intermediary (VI) is interposed between the hardware and the SI as illustrated in Figure 9-2.

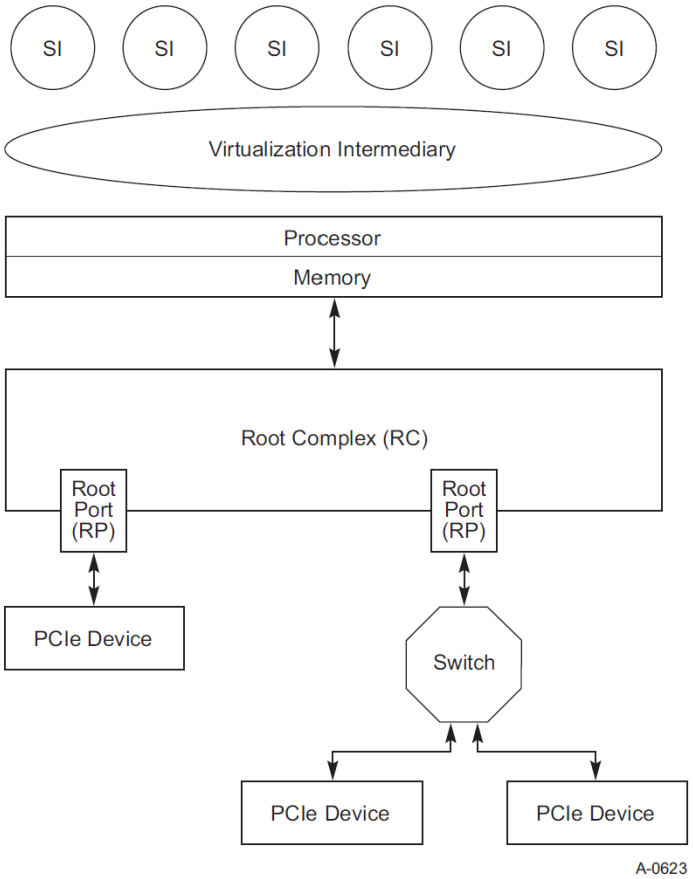


Figure 9-2: Generic Platform Configuration with a VI and Multiple SI

The VI takes sole ownership of the underlying hardware. Using a variety of methods outside of the scope of this specification, the VI abstracts the hardware to present each SI with its own virtual system. The actual hardware resources available to each SI can vary based on workload or customer-specific policies. While this approach works well for many environments, I/O intensive workloads can suffer significant performance degradation. Each I/O operation – inbound or outbound – must be intercepted and processed by the VI adding significant platform resource overhead.

To reduce platform resource overhead, PCI-SIG® developed the SR-IOV technology contained within this specification. The benefits of SR-IOV technology are:

- The ability to eliminate VI involvement in main data movement actions – DMA, Memory space access, interrupt processing, etc. Elimination of VI interception and processing of each I/O operation can provide significant application and platform performance improvements.
- Standardized method to control SR-IOV resource configuration and management through Single Root PCI Manager (SR-PCIM).
    - Due to a variety of implementation options – system firmware, VI, operating system, I/O drivers, etc. – SR-PCIM implementation is outside the scope of this specification.
- The ability to reduce the hardware requirements and associated cost with provisioning potentially a significant number of I/O Functions within a device.
- The ability to integrate SR-IOV with other I/O virtualization technologies such as Address Translation Services (ATS), Address Translation and Protection Table (ATPT) technologies, and interrupt remapping technologies to create robust, complete I/O virtualization solutions.

Figure 9-3 illustrates an example SR-IOV capable platform.

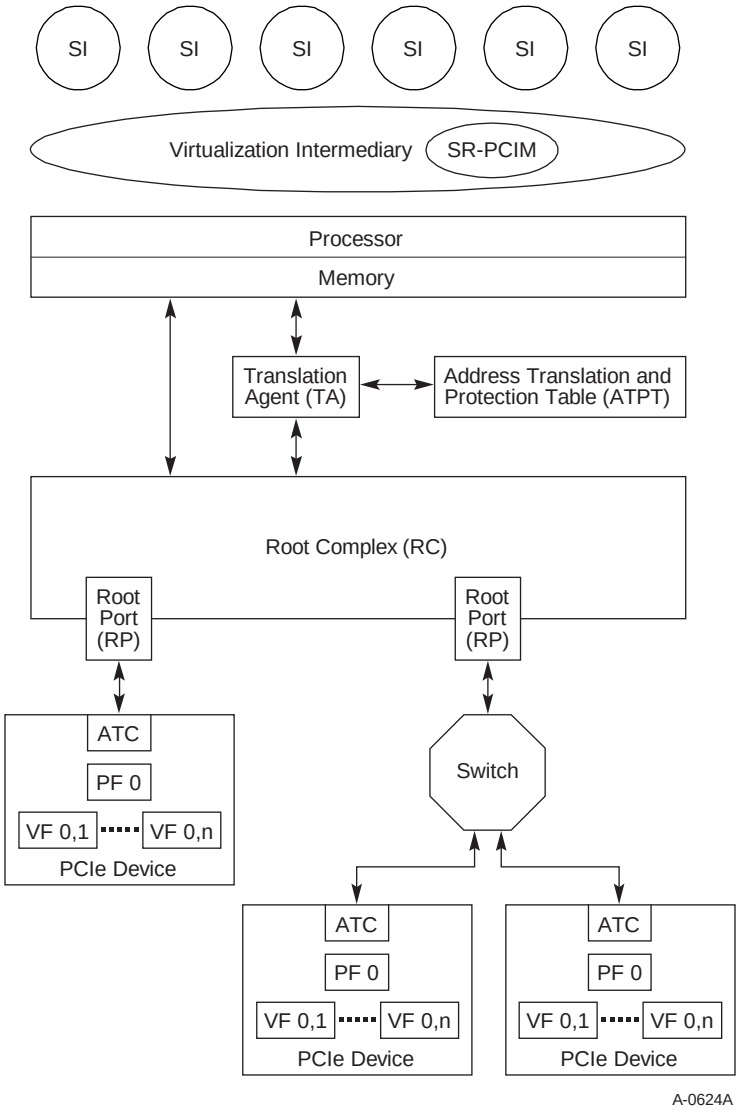


Figure 9-3: Generic Platform Configuration with SR-IOV and IOV Enablers

The SR-IOV generic platform configuration is composed of the following additional functional elements:

- SR-PCIM – Software responsible for the configuration of the SR-IOV capability, management of Physical Functions and Virtual Functions, and processing of associated error events and overall device controls such as power management and hot-plug services.
- Optional Translation Agent (TA) – A TA is hardware or a combination of hardware and software responsible for translating an address within a PCIe transaction into the associated platform physical address. A TA may contain an Address Translation Cache (ATC) to accelerate translation table access. A TA may also support Address Translation Services (ATS) which enables a PCIe Function to obtain address translations *a priori* to DMA access to the associated memory. See Chapter 10 for more details on ATS benefits and operation.

- Optional Address Translation and Protection Table (ATPT) – An ATPT contains the set of address translations accessed by a TA to process PCIe requests – DMA Read, DMA Write, or interrupt requests. See Address Translation Services (Chapter 10) for additional details.
    - In PCIe, interrupts are treated as memory write operations. Through the combination of a Requester Identifier and the address contained within a PCIe transaction, an interrupt can be routed to any target (e.g., a processor core) transparent to the associated I/O Function.
    - DMA Read and Write requests are translated through a combination of the Routing ID and the address contained within a PCIe transaction.
- Optional Address Translation Cache (ATC) – An ATC can exist in two locations within a platform – within the TA which can be integrated within or sit above an RC – or within a PCIe Device. Within an RC, the ATC enables accelerated translation look ups to occur. Within a Device, the ATC is populated through ATS technology. PCIe transactions that indicate they contain translated addresses may bypass the platform's ATC in order to improve performance without compromising the benefits associated with ATPT technology. See Address Translation Services (Chapter 10) for additional details.
- Optional Access Control Services (ACS) – ACS defines a set of control points within a PCI Express topology to determine whether a TLP should be routed normally, blocked, or redirected. In a system that supports SR-IOV, ACS may be used to prevent device Functions assigned to the VI or different SIs from communicating with one another or a peer device. Redirection may permit a Translation Agent to translate Upstream memory TLP addresses before a peer-to-peer forwarding decision is made. Selective blocking may be provided by the optional ACS P2P Egress Control. ACS is subject to interaction with ATS. See Section 9.3.7.6 for additional details.
- Physical Function (PF) – A PF is a PCIe Function that supports the SR-IOV capability and is accessible to an SR-PCIM, a VI, or an SI.
- Virtual Function (VF) – A VF is a "light-weight" PCIe Function that is directly accessible by an SI.
    - Minimally, resources associated with the main data movement of the Function are available to the SI. Configuration resources should be restricted to a trusted software component such as a VI or SR-PCIM.
    - A VF can be serially shared by different SI, (i.e., a VF can be assigned to one SI and then reset and assigned to another SI.)
    - A VF can be optionally migrated from one PF to another PF. The migration process itself is outside the scope of this specification but is facilitated through configuration controls defined within this specification.
- All VFs associated with a PF must be the same device type as the PF, (e.g., the same network device type or the same storage device type.)

To compare and contrast a PCIe Device with a PCIe SR-IOV capable device, examine the following set of figures. Figure 9-4 illustrates an example PCIe-compliant Device.

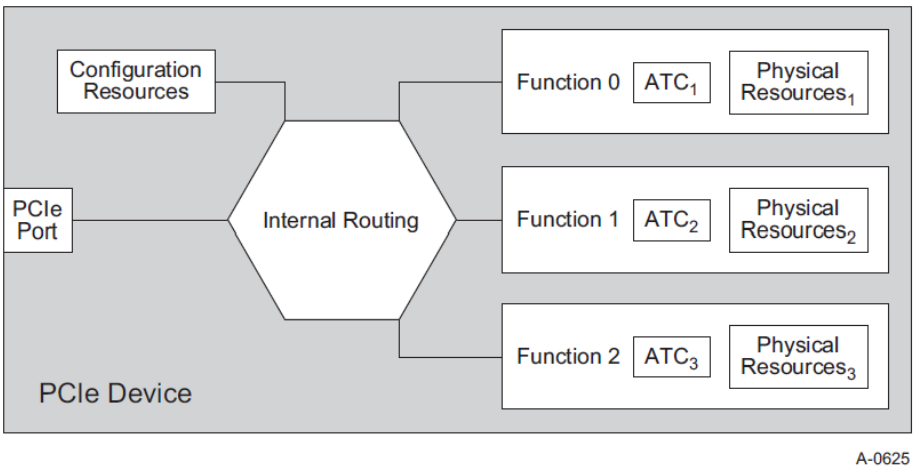


Figure 9-4: Example Multi-Function Device

This figure illustrates an example multi-Function PCIe Device with the following characteristics:

- The PCIe Device shares a common PCIe Link. The Link and PCIe functionality shared by all Functions is managed through Function 0.
  - While this figure illustrates only three Functions, with the use of the Alternative Routing Identifier (ARI) capability, a PCIe Device can support up to 256 Functions.
  - All Functions use a single Bus Number captured through the PCI enumeration process.
- In this example, each PCIe Function supports the ATS capability and therefore has an associated ATC to manage ATS obtained translated addresses.
- Each PCIe Function has a set of unique physical resources including a separate configuration space and BAR.
- Each PCIe Function can be assigned to an SI. To prevent one SI from impacting another, all PCIe configuration operations should be intercepted and processed by the VI.

As this figure illustrates, the hardware resources scale with the number of Functions provisioned. Depending upon the complexity and size of the device, the incremental cost per Function will vary. To reduce the incremental hardware cost, a device can be constructed using SR-IOV to support a single PF and multiple VFs as illustrated in Figure 9-5.

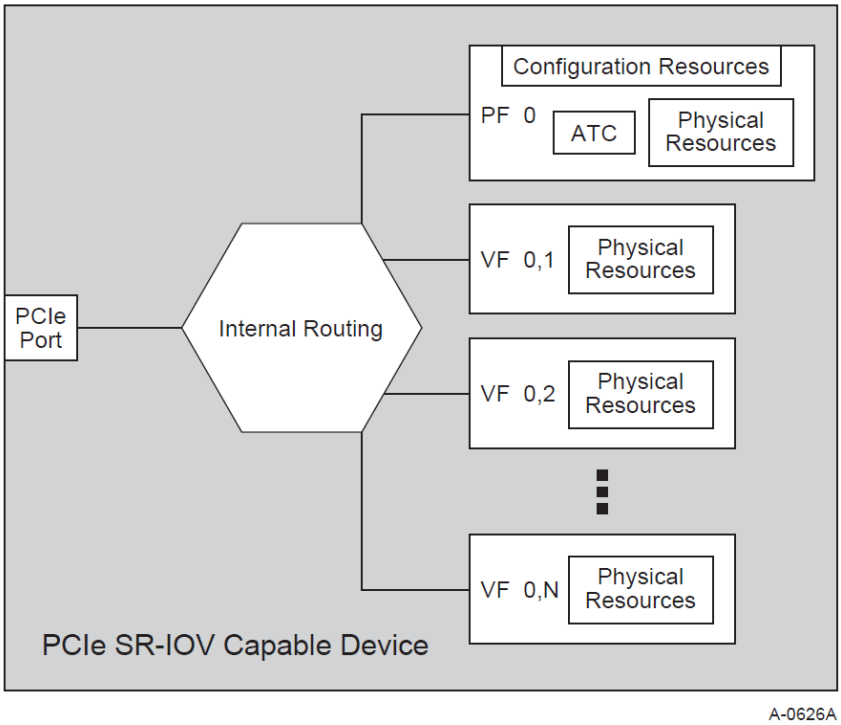


Figure 9-5: Example SR-IOV Single PF Capable Device

The example in Figure 9-5 illustrates a single PF with N VFs. Key observations to note:

- The PF is a PCIe-compliant.
  - Initially and after a conventional reset, a PCIe Function that supports the SR-IOV capabilities defined in this specification shall have the SR-IOV capabilities disabled.
  - To discover the page sizes supported by a PF and its associated VF, the Supported Page Sizes configuration field is read. For additional information on how this field can be used to align PF or VF Memory space apertures on a system page boundary, see Section 9.2.1.1.1.
- PF nomenclature **PF *M*** designates the PF at Function number M.
- VF nomenclature **VF *M,N*** designates the Nth VF associated with PF M. VFs are numbered starting with 1 so the first VF associated with PF M is VF M,1.
- Each VF shares a number of common configuration space fields with the PF; (i.e., where the fields are applicable to all VF and controlled through a single PF. Sharing reduces the hardware resource requirements to implement each VF.)
  - A VF uses the same configuration mechanisms and header types as a PF.
  - All VFs associated with a given PF share a VF BAR set (see Section 9.3.3.14) and share a VF Memory Space Enable (MSE) bit in the SR-IOV extended capability (see Section 9.3.3.3.4) that controls access to the VF Memory space. That is, if the VF MSE bit is Clear, the memory mapped space allocated for all VFs is disabled.
  - The InitialVFs and TotalVFs fields (see Section 9.3.3.5 and Section 9.3.3.6) are used to discover the maximum number of VFs that can be associated with a PF.

- If the Device does not support VF migration, TotalVFs and InitialVFs shall contain the same value. If the Device supports VF migration, when TotalVFs is read, the PF must return the number of VFs that can be assigned to the PF. For such a Device, when InitialVFs is read, the PF must return the initial number of VFs assigned to the PF.

- Each Function, PF, and VF is assigned a unique Routing ID. The Routing ID for each PF is constructed as per Section 2.2.4.2. The Routing ID for each VF is determined using the Routing ID of its associated PF and fields in that PF's SR-IOV Capability.
- All PCIe and SR-IOV configuration access is assumed to be through a trusted software component such as a VI or an SR-PCIM.
- Each VF contains a non-shared set of physical resources required to deliver Function-specific services, (e.g., resources such as work queues, data buffers, etc.) These resources can be directly accessed by an SI without requiring VI or SR-PCIM intervention.
- One or more VF may be assigned to each SI. Assignment policies are outside the scope of this specification.
- While this example illustrates a single ATC within the PF, the presence of any ATC is optional. In addition, this specification does not preclude an implementation from supporting an ATC per VF within the Device.
- Internal routing is implementation specific.
- While many potential usage models exist regarding PF operation, a common usage model is to use the PF to bootstrap the device or platform strictly under the control of a VI. Once the SR-IOV capability is configured enabling VF to be assigned to individual SI, the PF takes on a more supervisory role. For example, the PF can be used to manage device-specific functionality such as internal resource allocation to each VF, VF arbitration to shared resources such as the PCIe Link or the Function-specific Link (e.g., a network or storage Link), etc. These policy, management, and resource allocation operations are outside the scope of this specification.

Another example usage model is illustrated in Figure 9-6. In this example, the device supports multiple PFs each with its own set of VFs.

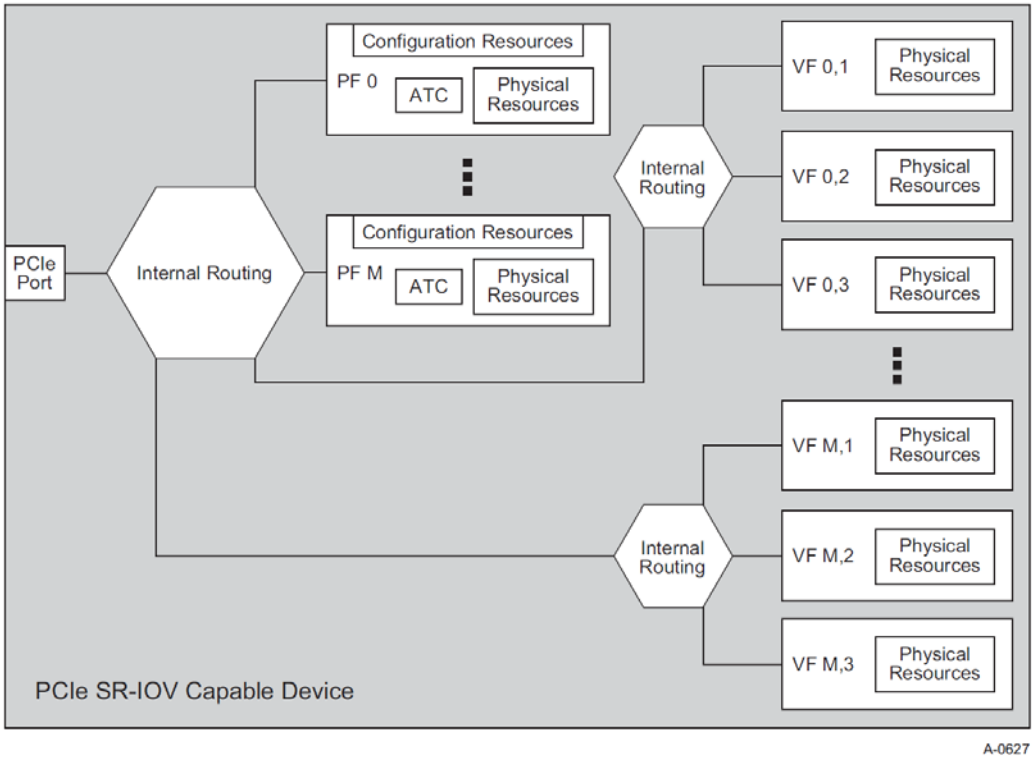


Figure 9-6: Example SR-IOV Multi-PF Capable Device

Key observations to note:

- Each PF can be assigned zero or more VFs. The number of VFs per PF is not required to be identical for all PFs within the device.
- The ARI capability enables Functions to be assigned to Function Groups and defines how Function Group arbitration can be configured. PFs and VFs can be assigned to Function Groups and take advantage of the associated arbitration capabilities. Within each Function Group, though, arbitration remains implementation specific.
- Internal routing between PFs and VFs is implementation specific.
- For some usage models, all PFs may be the same device type; (e.g., all PFs deliver the same network device or all deliver the same storage device functionality.) For other usage models, each PF may represent a different device type; (e.g., in Figure 9-6, one PF might represent a network device while another represents an encryption device.)
    - In situations where there is a usage model dependency between device types, such as for each VF that is a network device type, each SI also requires a VF that is an encryption device type. The SR-IOV capability provides a method to indicate these dependencies. The policies used to construct these dependencies as well as assign dependent sets of VF to a given SI are outside the scope of this specification.

As seen in the prior example, the number of PF and VF can vary based on usage model requirements. To support a wide range of options, an SR-IOV Device can support the following number and mix of PF and VF:

- Using the Alternative Routing Identifier (ARI) capability, a device may support up to 256 PFs. Function Number assignment is implementation specific and may be sparse throughout the 256 Function Number space.
- A PF can only be associated with the Device's captured Bus Number as illustrated in Figure 9-7.
- SR-IOV Devices may consume more than one Bus Number. A VF can be associated with any Bus Number within the Device's Bus Number range – the captured Bus Number plus any additional Bus Numbers configured by software. See Section 9.2.1.2 for details.
    - Use of multiple Bus Numbers enables a device to support a very large number of VFs – up to the size of the Routing ID space minus the bits used to identify intervening busses.
    - If software does not configure sufficient additional Bus Numbers, then the VFs implemented for the additional Bus Numbers may not be visible.

IMPLEMENTATION NOTE

Function Co-location

The ARI capability enables a Device to support up to 256 Functions – Functions, PFs, or VFs in any combination – associated with the captured Bus Number. If a usage model does not require more than 256 Functions, implementations are strongly encouraged to co-locate all Functions, PFs, and VFs within the captured Bus Number and not require additional Bus Numbers to access VFs.

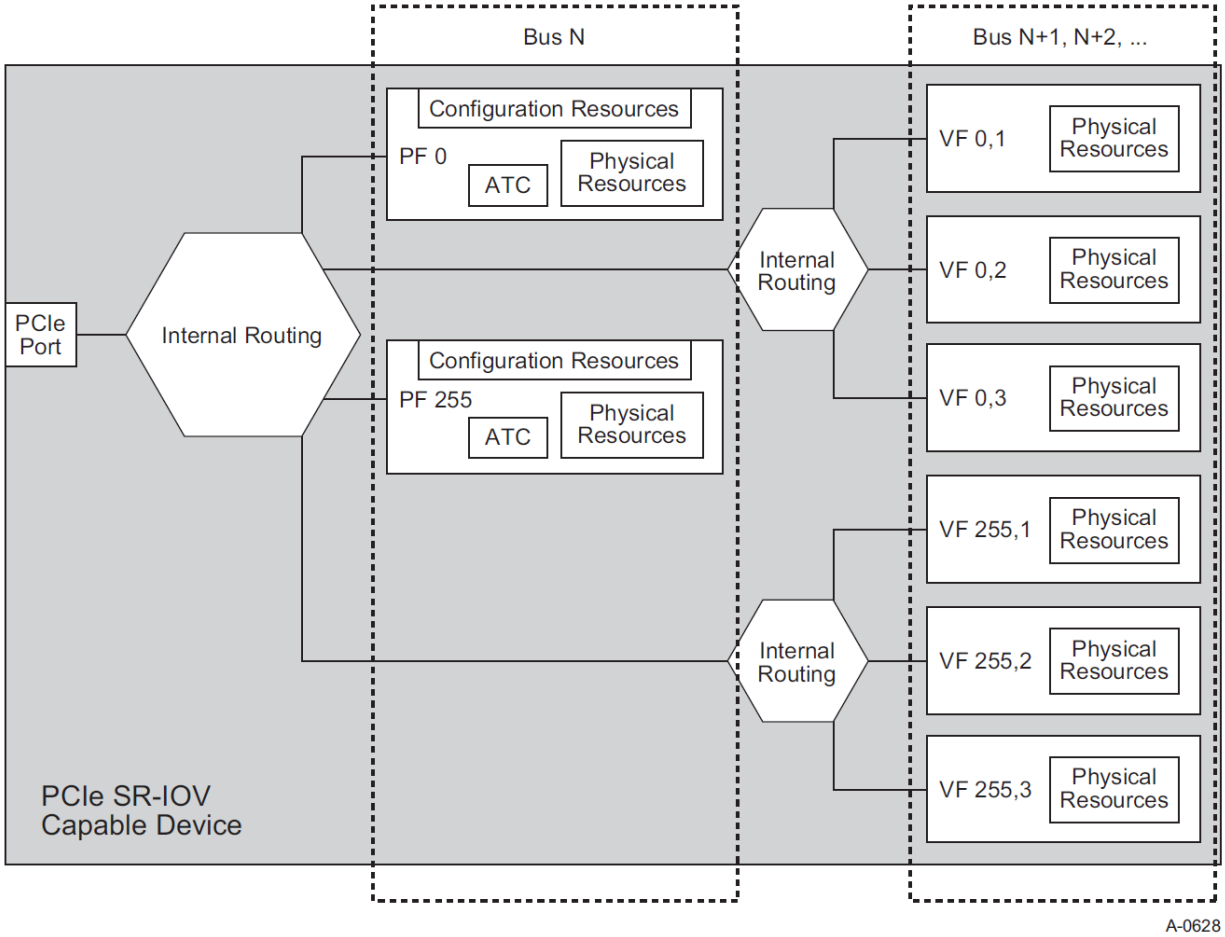


Figure 9-7: Example SR-IOV Device with Multiple Bus Numbers

In this last example, Figure 9-8, a device implementation may mix any number of Functions, PFs, and VFs.

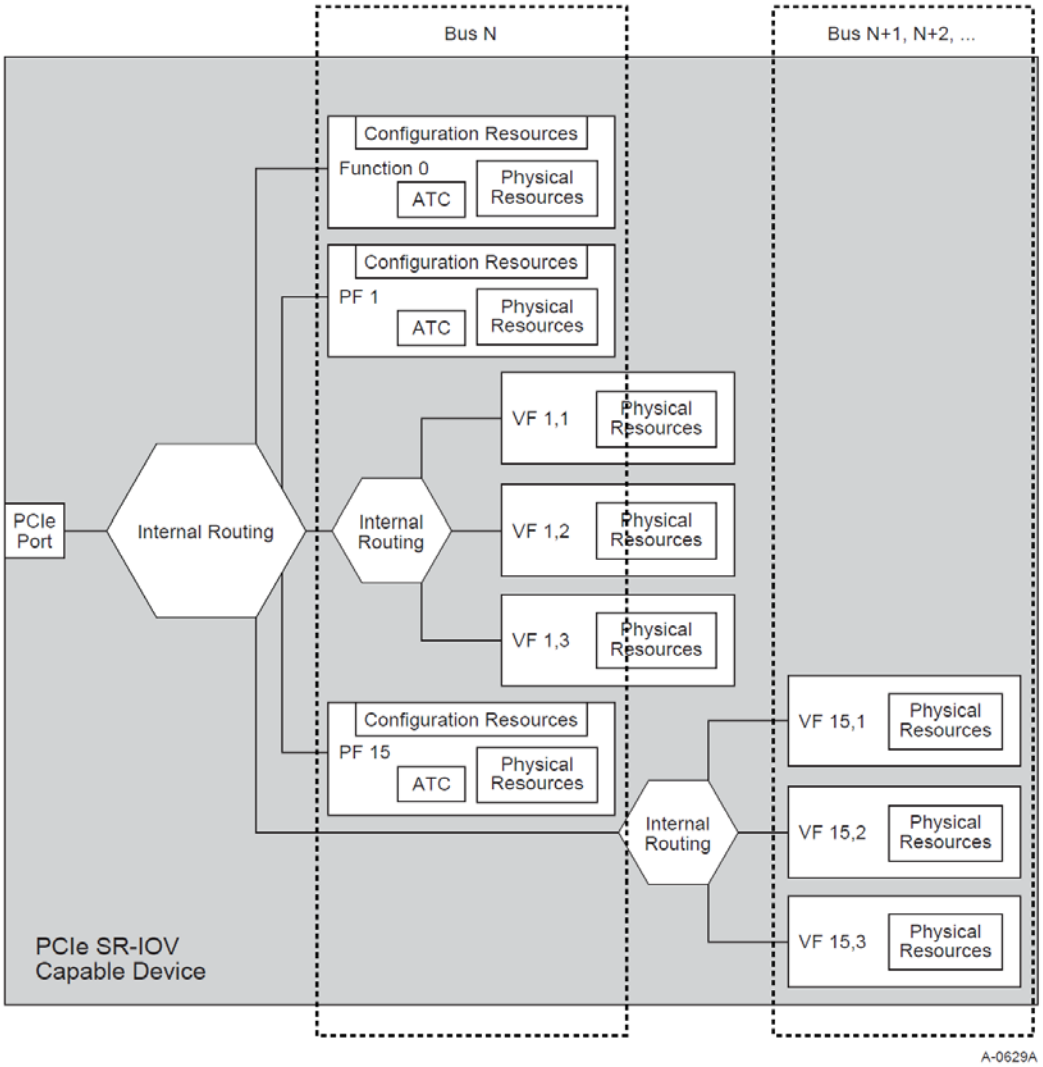


Figure 9-8: Example SR-IOV Device with a Mixture of Function Types

Key observations to note:

- Each Device must contain a Function 0. Function 0 may be a PF (i.e., it may include the SR-IOV extended capability).
- Any mix of Functions can be associated with the captured Bus Number.
  - Non-VFs can only be associated with the captured Bus Number.
- If the ARI capability is supported, Functions can be assigned to Function Groups. The assignment policy is outside the scope of this specification. If the ARI capability is not supported, Functions can still use the Function arbitration capabilities as defined in Section 6.3.3.4.

9.1.1 PCI Technologies Interoperability

Establishing clear interoperability requirements is critical to the success of any technology. To this end, the PCI-SIG I/O virtualization specifications are organized to maximize the interoperability potential for compliant implementations. Conceptually, this is viewed as a set of concentric circles that define how functionality is layered to build an IOV capable component as illustrated in Figure 9-9.

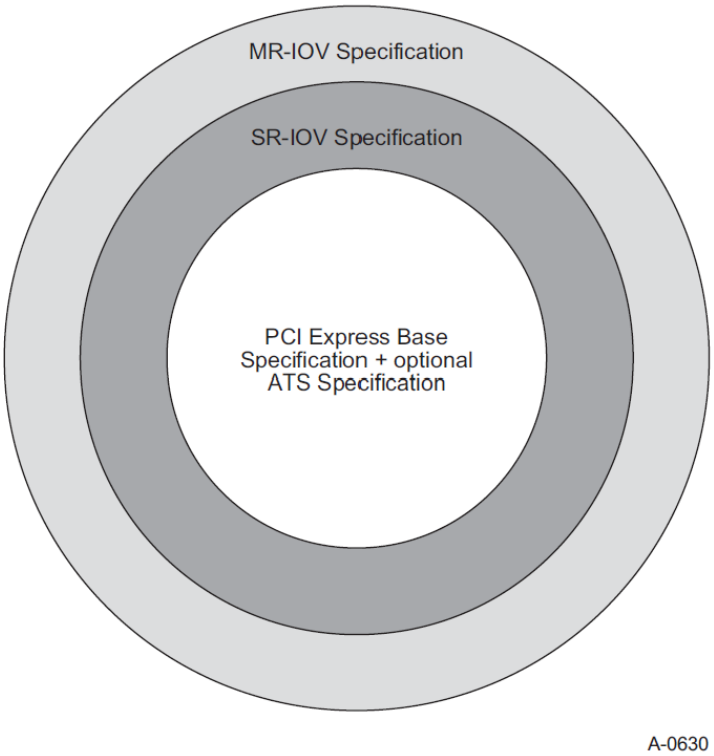


Figure 9-9: I/O Virtualization Interoperability

Key observations to note:

- At their core, I/O virtualization specifications build upon the *PCI Express Base Specification*. All IOV implementations must be compliant with the *PCI Express Base Specification, Revision 1.1* or later versions. Where applicable, the IOV specifications note relevant deltas between these versions.
  - None of the IOV specifications touch the physical layer.
  - SR-IOV does not touch the data Link or transaction layers specified in the *PCI Express Base Specification*.
  - The *Multi-Root I/O Virtualization and Sharing Specification* does not touch the transaction layer specified in the *PCI Express Base Specification*.
  - All I/O virtualization capabilities are communicated through new PCI Express capabilities implemented in PCI Express extended configuration space.
  - I/O virtualization specifications have no impact on PCI or PCI-X specifications.

- A hierarchy can be composed of a mix of PCI Express and PCI Express-to-PCI/PCI-X Bridges.
  - A PCI Express-to-PCI/PCI-X Bridge and PCI/PCI-X Devices can be serially shared by multiple SIs.

- ATS defines optional functionality applicable to any Function. ATS can be supported in SR-IOV components.
- To implement an SR-IOV Device, SR-IOV requires the Device to be fully compliant with the *PCI Express Base Specification*.
  - A hierarchy can be composed of a mix of SR-IOV and non-SR-IOV components. For example, a hierarchy may contain any mix of SR-IOV and non-SR-IOV Endpoint Devices.
- To implement an MR-IOV Device, the *Multi-Root I/O Virtualization and Sharing Specification* requires the Device to be fully compliant with this specification.

9.2 Initialization and Resource Allocation

9.2.1 SR-IOV Resource Discovery

The following sections describe how software determines that a Device is SR-IOV capable and subsequently identifies VF resources through Virtual Function Configuration Space.

9.2.1.1 *Configuring SR-IOV Capabilities*

This section describes the fields that must be configured before enabling a PF's IOV Capabilities. The VFs are enabled by Setting the PF's VF Enable bit (see Section 9.3.3.3.1) in the SR-IOV extended capability.

The NumVFs field (see Section 9.3.3.7) defines the number of VFs that are enabled when VF Enable is Set in the associated PF.

9.2.1.1.1 Configuring the VF BAR Mechanisms

This section describes how the VF BARs are configured to map memory space. VFs do not support I/O Space and thus VF BARs shall not indicate I/O Space.

The System Page Size field (see Section 9.3.3.13) defines the page size the system will use to map the VF's PCIe memory addresses when the PF's IOV Capabilities are enabled. The System Page Size field is used by the PF to align the Memory space aperture defined by each VF BAR to a system page boundary. The value chosen for the System Page Size must be one of the Supported Page Sizes (see Section 9.3.3.12) in the SR-IOV extended capability.

The behavior of VF BARs is the same as the normal PCI Memory Space BARs (see Section 7.5.1.2.1), except that a VF BAR describes the aperture for each VF, whereas a PCI BAR describes the aperture for a single Function. The attributes for some of the bits in the VF BARs are affected by the VF Resizable BAR capability (see Section 9.3.7.5) if it is implemented.

- The behavior described in Section 7.5.1.2.1 for determining the memory aperture of a Function's BAR applies to each VF BAR. That is, the size of the memory aperture required for each VF BAR can be determined by writing all "1"s and then reading the VF BAR. The results read back must be interpreted as described in Section 7.5.1.2.1.
- The behavior for assigning the starting memory space address of each BAR associated with the first VF is also as described in Section 7.5.1.2.1. That is, the address written into each VF BAR is used by the Device for the starting address of the first VF.
- The difference between VF BARs and BARs described in Section 7.5.1.2.1 is that for each VF BAR, the memory space associated with the second and higher VFs is derived from the starting address of the first VF and the memory space aperture. For any given VF (v), the starting address of its Memory space aperture for any implemented BAR (b) is calculated according to the following formula:

  $$BAR_b\ VF_v \text{ starting address} = VF\ BAR_b + (v - 1) \times (VF\ BAR_b \text{ aperture size})$$

  where VF BAR_b aperture size is the size of VF BAR_b as determined by the usual BAR probing algorithm as described in Section 9.3.3.14.

  VF memory space is not enabled until both VF Enable and VF MSE have been Set (see Section 9.3.3.3.1 and Section 9.3.3.3.4). Note that changing System Page Size (see Section 9.3.3.13) may affect the VF BAR aperture size.

Figure 9-10 shows an example of the PF and VF Memory space apertures.

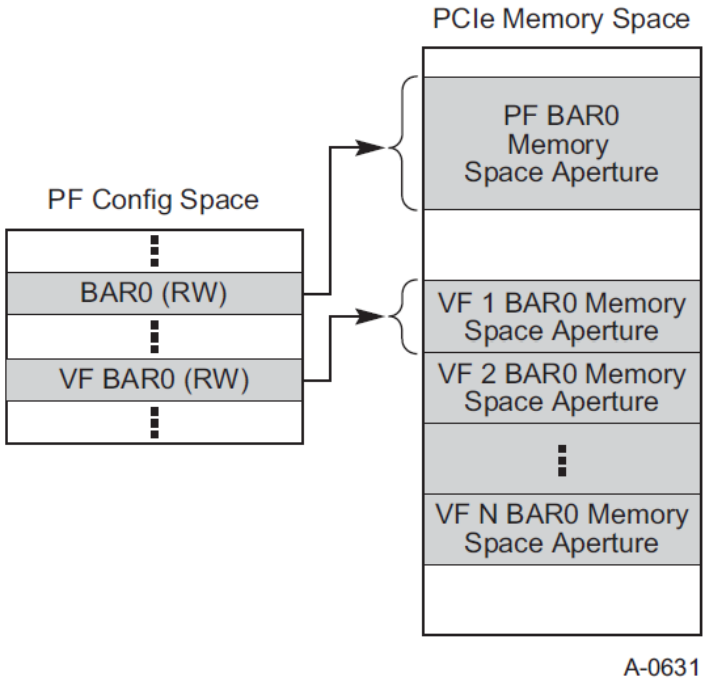


Figure 9-10: BAR Space Example for Single BAR Device

9.2.1.2 *VF Discovery*

The First VF Offset and VF Stride fields in the SR-IOV extended capability are 16-bit Routing ID offsets. These offsets are used to compute the Routing IDs for the VFs with the following restrictions:

- The value in NumVFs in a PF (Section 9.3.3.7) may affect the values in First VF Offset (Section 9.3.3.9) and VF Stride (Section 9.3.3.10) of that PF.
- The value in ARI Capable Hierarchy (Section 9.3.3.3.5) in the lowest-numbered PF of the Device (for example PF_0) may affect the values in First VF Offset and VF Stride in all PFs of the Device.
- NumVFs of a PF may only be changed when VF Enable (Section 9.3.3.3.1) of that PF is Clear.
- ARI Capable Hierarchy (Section 9.3.3.3.5) may only be changed when VF Enable is Clear in all PFs of a Device.

IMPLEMENTATION NOTE

NumVFs and ARI Capable Hierarchy

After configuring NumVFs and ARI Capable Hierarchy where applicable, software may read First VF Offset and VF Stride to determine how many Bus Numbers would be consumed by the PF's VFs. The additional Bus Numbers, if any, are not actually used until VF Enable is Set.

Table 9-1 describes the algorithm used to determine the Routing ID associated with each VF.

Table 9-1: VF Routing ID Algorithm

VF Number	VF Routing ID
VF 1	(PF Routing ID + First VF Offset) Modulo 2^{16}
VF 2	(PF Routing ID + First VF Offset + VF Stride) Modulo 2^{16}
VF 3	(PF Routing ID + First VF Offset + 2 * VF Stride) Modulo 2^{16}
…	…
VF N	(PF Routing ID + First VF Offset + (N-1) * VF Stride) Modulo 2^{16}
…	…
VF NumVFs (last one)	(PF Routing ID + First VF Offset + (NumVFs-1) * VF Stride) Modulo 2^{16}

All arithmetic used in this Routing ID computation is 16-bit unsigned dropping all carries.

All VFs and PFs must have distinct Routing IDs. The Routing ID of any PF or VF must not overlap with the Routing ID of any other PF or VF given any valid setting of NumVFs across all PFs of a device.

VF Stride and First VF Offset are constants. Except as stated earlier in this section, their values may not be affected by settings in this or other Functions of the Device.

VFs may reside on different Bus Number(s) than the associated PF. This can occur if, for example, First VF Offset has the value 0100h. A VF shall not be located on a Bus Number that is

numerically smaller than its associated PF. A VF that is located on the same Bus Number as its associated PF shall not be located on a Device Number that is numerically smaller than the PF[142].

VFs of an SR-IOV RCiEP Device are associated with the same Root Complex Event Collector (if any) as their PF.

As per Section 2.2.6.2, SR-IOV capable Devices that are associated with an Upstream Port capture the Bus Number from any Type 0 Configuration Write Request. SR-IOV capable Devices do not capture the Bus Number from any Type 1 Configuration Write Requests. SR-IOV capable RCiEPs use an implementation specific mechanism to assign their Bus Numbers.

Switch processing of Bus Numbers is unchanged from base PCIe. In base PCIe, the switch sends all Configuration Requests in the range Secondary Bus Number (inclusive) to Subordinate Bus Number (inclusive) to the Device. Type 1 Configuration Requests addressing the Secondary Bus Number are converted into Type 0 Configuration Requests while Type 1 Configuration Requests addressing Bus Numbers between Secondary Bus Number (exclusive) and Subordinate Bus Number (inclusive) are forwarded on to the Device as Type 1 Configuration Requests.

Note: Bus Numbers are a constrained resource. Devices are strongly encouraged to avoid leaving "holes" in their Bus Number usage to avoid wasting Bus Numbers.

All PFs must be located on the Device's captured Bus Number.

IMPLEMENTATION NOTE

VFs Spanning Multiple Bus Numbers

As an example, consider an SR-IOV Device that supports a single PF. Initially, only PF 0 is visible. Software Sets ARI Capable Hierarchy. From the SR-IOV Capability it determines: InitialVFs is 600, First VF Offset is 1 and VF Stride is 1.

- If software sets NumVFs in the range [0 .. 255], then the Device uses a single Bus Number.
- If software sets NumVFs in the range [256 .. 511], then the Device uses two Bus Numbers.
- If software sets NumVFs in the range [512 .. 600], then the Device uses three Bus Numbers.

PF 0 and VF 0,1 through VF 0,255 are always on the first (captured) Bus Number. VF 0,256 through VF 0,511 are always on the second Bus Number (captured Bus Number plus 1). VF 0,512 through VF 0,600 are always on the third Bus Number (captured Bus Number plus 2).

Software should configure Switch Secondary and Subordinate Bus Number fields to route enough Bus Numbers to the Device. If sufficient Bus Numbers are not available, software should reduce a Device's Bus Number requirements by not enabling SR-IOV and/or reducing NumVFs for some or all PFs of the Device prior to enabling SR-IOV.

After VF Enable is Set in some PF *n*, the Device must Enable VF *n*,1 through VF *n*,*m* (inclusive) where *m* is the smaller of InitialVFs and NumVFs. A Device receiving a Type 0 Configuration

[142] SR-IOV Devices immediately below a Downstream Port always have a Device Number of 0 and thus always satisfy this condition.

Request targeting an Enabled VF must process the Request normally. A Device receiving a Type 1 Configuration Request targeting an Enabled VF not located on the captured Bus Number must process the Request normally. A Device receiving a Type 1 Configuration Request targeting the Device's captured Bus Number must follow the rules for handling Unsupported Requests. Additionally, if VF MSE is Set, each Enabled VF must respond to PCIe Memory transactions addressing the memory space associated with that VF.

Functions that are not enabled (i.e., Functions for VFs above *m*) do not exist in the PCI Express fabric. As per Section 2.3.1, addressing Functions that do not exist will result in Unsupported Request (UR) being returned. This includes Functions on additional Bus Numbers.

IMPLEMENTATION NOTE

Multi-Function Devices with PFs and Switch Functions

SR-IOV devices may consume multiple bus numbers. Additional bus numbers beyond the first one are consecutive and immediately follow the first bus number assigned to the device. If an SR-IOV device also contains PCI-PCI Bridges (with Type 1 Configuration Space headers), the SR-IOV usage must be accounted for when programming the Secondary Bus Number for those Bridges. Software should determine the last Bus Number used by VFs first and then configure any co-located Bridges to use Bus Numbers above that value.

9.2.1.3 *Function Dependency Lists*

PCI Devices may have vendor-specific dependencies between Functions. For example, Functions 0 and 1 might provide different mechanisms for controlling the same underlying hardware. In such situations, the Device programming model might require that these dependent Functions be assigned to SIs as a set.

Function Dependency Lists are used to describe these dependencies (or to indicate that there are no Function dependencies). Software should assign PFs and VFs to SIs such that the dependencies are satisfied.

See Section 9.3.3.8 for details.

9.2.1.4 *Interrupt Resource Allocation*

PFs and VFs support either MSI, MSI-X interrupts, or both if interrupt resources are allocated. VFs shall not implement INTx. Interrupts are described in Section 9.5.

9.2.2 Reset Mechanisms

This section describes how PCI-base-defined-reset mechanisms affect Devices that support SR-IOV. It also describes the mechanisms used to reset a single VF and a single PF with its associated VFs.

9.2.2.1 *Conventional Reset*

A Conventional Reset to a Device that supports SR-IOV shall cause all Functions (including both PFs and VFs) to be reset to their original, power-on state as per the rules in Section 6.6.1. Section 9.3 describes the behavior for the fields defined.

Note: Conventional Reset clears VF Enable in the PF. Thus, VFs no longer exist after a Conventional Reset.

9.2.2.2 *FLR That Targets a VF*

VFs must support Function Level Reset (FLR).

Note: Software may use FLR to reset a VF. FLR to a VF affects a VF's state but does not affect its existence in PCI Configuration Space or PCI Bus address space. The VFs BARn values (see Section 9.3.3.14) and VF MSE (see Section 9.3.3.3.4) in the PF's SR-IOV extended capability are unaffected by FLRs issued to the VF.

9.2.2.3 *FLR That Targets a PF*

PFs must support FLR.

FLR to a PF resets the PF state as well as the SR-IOV extended capability including VF Enable which means that VFs no longer exist.

9.2.3 IOV Re-initialization and Reallocation

If VF Enable is Cleared after having been Set, all of the VFs associated with the PF no longer exist and must no longer issue PCIe transactions or respond to Configuration Space or Memory Space accesses. VFs must not retain any state after VF Enable has been Cleared (including sticky bits).

9.2.4 VF Migration

VF Migration is an optional feature in the *Multi-Root I/O Virtualization and Sharing Specification*. Devices that do not support the MR-IOV Extended Capability do not support VF Migration functionality.

In Multi-Root systems, VFs can be migrated between Virtual Hierarchies.

For VF Migration to occur, both VF Migration Capable and VF Migration Enable must be Set. VF Migration Capable indicates to SR-PCIM that VF Migration Hardware is present and that Multi-Root PCI Manager (MR-PCIM) has enabled its use. Hardware support for VF Migration is optional. SR-PCIM support for VF Migration is also optional.

VF Migration Enable indicates to Device hardware and to MR-PCIM that SR-PCIM has also enabled VF Migration.

Supporting VF Migration places the following requirements on SR-PCIM:

- The need to determine if a VF is *Active, Dormant,* or *Inactive*. A VF that is Active is available for use by the Single Root (SR). A VF that is Dormant can be configured by SR but will not issue transactions. A VF that is Inactive is not available for use by SR.
- The need to participate in *Migrate In* operations. A Migrate In operation is used to offer a VF to SR. A Migrate In operation is initiated by MR-PCIM. SR-PCIM can accept a Migrate In operation and move a VF to the *Active.Available* state.
- The need to participate in *Migrate Out* operations. A Migrate Out operation is used to request the graceful removal of an Active VF from use by SR. SR-PCIM can accept the Migrate Out and move the VF to the *Inactive.Unavailable* state.
- The need to handle *Migrate In Retractions*. This is when an offer of a VF to SR is retracted by MR-PCIM and the VF moves back to the *Inactive.Unavailable* state.

9.2.4.1 *Initial VF State*

This section describes the initial values in the VF Migration State Array (see Section 9.3.3.15.1).

If InitialVFs (Section 9.3.3.5) is non-zero, VF_1 through $VF_{InitialVFs}$ are in the Active.Available state. If TotalVFs (Section 9.3.3.6) is greater than InitialVFs, $VF_{InitialVFs+1}$ through $VF_{TotalVFs}$ are in the Inactive.Unavailable state. If VF Migration Enable (Section 9.3.3.3.2) is Clear, VFs above InitialVFs are not used.

If InitialVFs is 0, no VFs are in the Active.Available state. If TotalVFs equals InitialVFs all VFs are in the Active.Available state. If TotalVFs is 0, no VFs are associated with this PF and there is no VF Migration State Array.

Figure 9-11 describes this initial VF State.

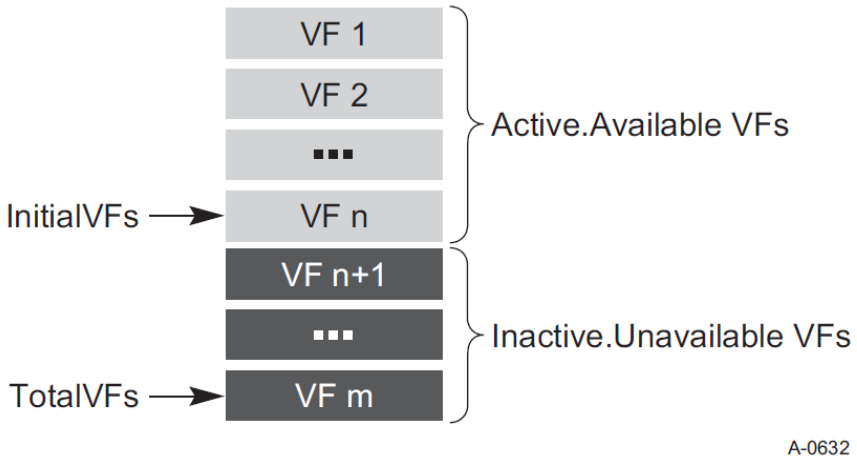


Figure 9-11: Initial VF Migration State Array

9.2.4.2 *VF Migration State Transitions*

VF migration follows the state diagram shown in Figure 9-12. The state values shown are contained in the VF State array entry associated with the VF. State transitions indicated by solid lines are

initiated by MR-PCIM. State transitions indicated by dotted and dashed lines are initiated by SR-PCIM.

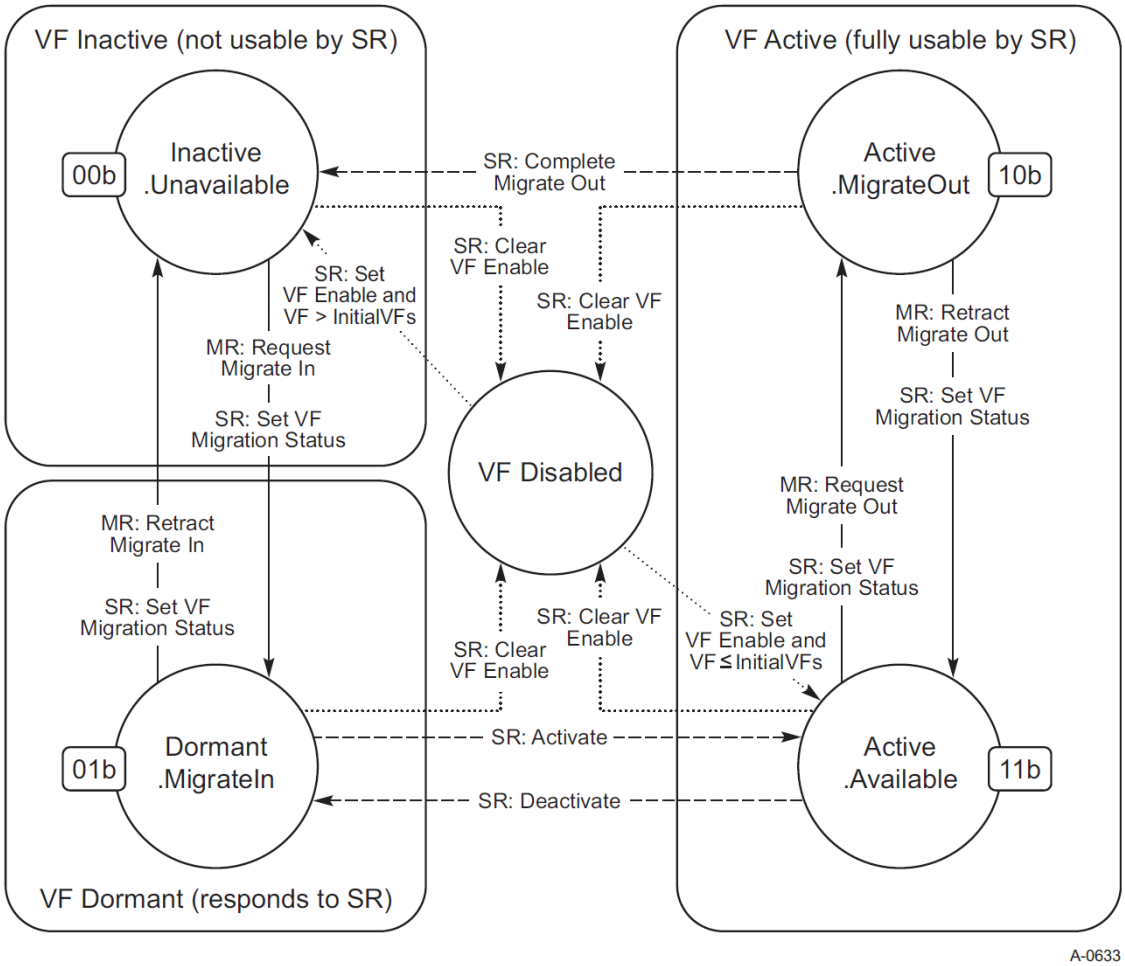


Figure 9-12: VF Migration State Diagram

SR-PCIM initiates a state transition by writing a new value to the VF Migration State Array. Devices ignore write transactions and no state transitions occur when SR-PCIM attempts to initiate any state transition other than in Table 9-2:

Table 9-2: SR-IOV VF Migration State Table

Current VF Enable	Current VF State	Written VF Enable	Written VF State	Meaning
1	Dormant.MigrateIn	1	Active.Available	SR Activate VF.
1	Active.Available	1	Dormant.MigrateIn	SR Deactivate VF.
1	Active.MigrateOut	1	Inactive.Unavailable	SR Complete Migrate Out.
1	Any	0	Any	SR Disables all VFs.
0	Any	1	Any	SR Enables all VFs. VFs transition to either Active.Available or Inactive.Unavailable based on the VF number and InitialVFs.

IMPLEMENTATION NOTE

Software State Migration Change Detection

SR-PCIM will typically need to verify that a state change took effect by re-reading VF Migration State after writing it.

VFs in states Inactive.Unavailable are not usable by software in any way. Requests targeting an Inactive VF receive Unsupported Request (UR). Within 100 ms of transitioning to the Inactive or Dormant states, the Device must ensure that no new transactions will be issued using the indicated Routing ID.

MR-PCIM initiates state transitions by using a different data structure as specified in the *Multi-Root I/O Virtualization and Sharing Specification*. The effects of such transitions are visible in the VF Migration State Array and in the VF Migration Status bit. All state transitions initiated by MR-PCIM cause the VF Migration Status bit to be Set.

This migration state machine exists for every VF that supports VF Migration. Migration state machines are not affected by Function Dependency Lists (see Section 9.2.1.3 and Section 9.3.3.8).

VF Migration State does not affect Function state. If VF state needs to be reset as part of a Migrate Out and/or Migrate In operation, SR-PCIM must issue an FLR to accomplish this. VF behavior when VF Migration occurs without an FLR is undefined.

IMPLEMENTATION NOTE

FLR and VF Migration

VF Migration from system A to system B will typically involve one FLR in system A before completing the MigrateOut operation and a second FLR in system B after accepting the MigrateIn operation but before using the VF. System A uses the first FLR to ensure that none of its data leaks out. System B uses the second FLR to ensure that it starts with a clean slate.

9.3 Configuration

9.3.1 Overview

This section provides SR-IOV-added requirements for implementing PFs and VFs.

PFs are discoverable in configuration space, as with all Functions. PFs contain the SR-IOV Extended Capability described in Section 9.3.3. PFs are used to discover, configure, and manage the VFs associated with the PF and for other things described in this specification.

9.3.2 Configuration Space

PFs that support SR-IOV shall implement the SR-IOV Capability as defined in the following sections. VFs shall implement configuration space fields and capabilities as defined in the following sections.

9.3.3 SR-IOV Extended Capability

The SR-IOV Extended Capability defined here is a PCIe extended capability that must be implemented in each PF that supports SR-IOV. This Capability is used to describe and control a PF's SR-IOV Capabilities.

For Multi-Function Devices, each PF that supports SR-IOV shall provide the Capability structure defined in this section. This Capability structure may be present in any Function with a Type 0 Configuration Space header. This Capability must not be present in Functions with a Type 1 Configuration Space header.

Figure 9-13: shows the SR-IOV Capability structure.

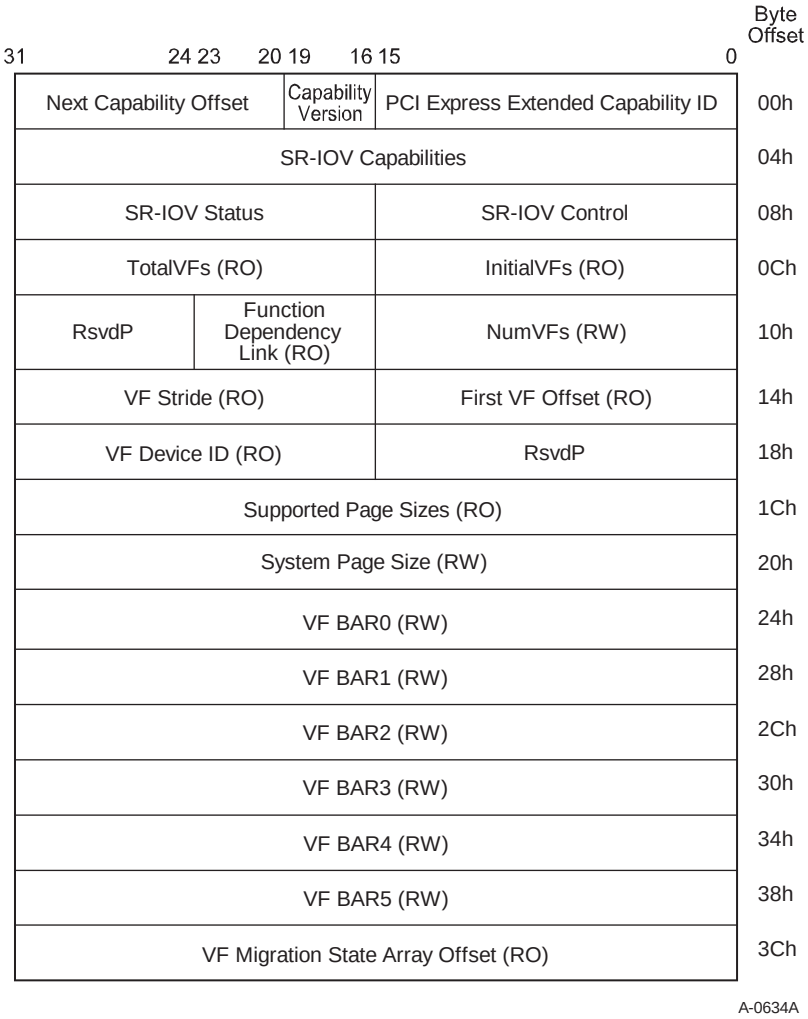


Figure 9-13: SR-IOV Extended Capability

9.3.3.1 *SR-IOV Extended Capability Header (00h)*

Table 9-3: SR-IOV Extended Capability Header defines the SR-IOV Extended Capability header. The Capability ID for the SR-IOV Extended Capability is 0010h.

Table 9-3: SR-IOV Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. The Extended Capability ID for the SR-IOV Extended Capability is 0010h.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.	RO

Bit Location	Register Description	Attributes
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh.	RO

9.3.3.2 *SR-IOV Capabilities (04h)*

Table 9-4: SR-IOV Capabilities defines the layout of the SR-IOV Capabilities field.

Table 9-4: SR-IOV Capabilities

Bit Location	Register Description	Attributes
0	**VF Migration Capable** – If Set, the PF is Migration Capable and operating under a Migration Capable MR-PCIM.	RO
1	ARI Capable Hierarchy Preserved PCI Express Endpoint: If Set, the ARI Capable Hierarchy bit is preserved across certain power state transitions. RCiEP: Not applicable – it is strongly recommended that this bit be hardwired to 0b.	RO
2	**VF 10-Bit Tag Requester Supported** – If Set, all VFs must support 10-Bit Tag Requester capability. If Clear, VFs must not support 10-Bit Tag Requester capability. If the 10-Bit Tag Requester Supported bit in the PF's Device Capabilities 2 Register is Clear, this bit must be Clear.	HwInit
20:3	**Reserved** – These fields are currently Reserved	RsvdP
31:21	**VF Migration Interrupt Message Number** – Indicates the MSI/MSI-X vector used for migration interrupts. The value in this field is undefined if VF Migration Capable is Clear.	RO

9.3.3.2.1 VF Migration Capable

VF Migration Capable is Set to indicate that the PF supports VF Migration. If Clear, the PF does not support VF Migration.

VF Migration support is an optional feature of *Multi-Root I/O Virtualization and Sharing Specification*. If the PF does not implement hardware for VF Migration, this bit shall be implemented as RO value 0b. If the PF implements hardware for VF Migration, this bit is controlled by mechanisms defined in the *Multi-Root I/O Virtualization and Sharing Specification* and indicates the presence of support for VF Migration.

Devices that implement SR-IOV only shall implement this field as RO value zero.

PFs that support VF Migration must implement MSI or MSI-X interrupts (or both).

9.3.3.2.2 ARI Capable Hierarchy Preserved

ARI Capable Hierarchy Preserved is Set to indicate that the PF preserves the ARI Capable Hierarchy bit across certain power state transitions (see Section 9.3.3.3.5). Components must either Set this bit or Set the No_Soft_Reset bit (see Section 9.6.2). It is recommended that components set this bit even if they also set No_Soft_Reset.

ARI Capable Hierarchy Preserved is only present in the lowest-numbered PF of a Device (for example PF0). ARI Capable Hierarchy Preserved is Read Only Zero in other PFs of a Device.

ARI Capable Hierarchy Preserved does not apply to RCiEPs, and its value is undefined (see Section 9.3.3.3).

9.3.3.2.3 VF 10-Bit Tag Requester Supported

If a PF supports 10-Bit Tag Requester capability, its associated VFs are permitted to support 10-Bit Tag Requester capability as well, but this is optional. Especially for usage models where the bulk of the traffic is spread across several VFs concurrently, it may not be necessary for individual VFs to use 10-Bit Tags so they can support >256 outstanding Non-Posted Requests each.

For a given PF, it is required that either all or none of its associated VFs support 10-Bit Tag Requester capability. This avoids unnecessary implementation and management complexity. See Section 10.3.5.9.

VFs that support 10-Bit Tag Requester capability have additional requirements beyond other Function types in order to simplify error handling and reduce the possibility of 10-Bit Tag related errors with one VF impacting other traffic.

- If the VF 10-Bit Tag Requester Enable bit in the SR-IOV Control register is Set, then each VF must use 10-Bit Tags for all Non-Posted Requests that it generates.
- For each outstanding 10-Bit Tag Request, if the VF receives a Completion that matches the outstanding Request other than Tag[9:8] being 00b, the VF must prevent that Request from (eventually) generating a Completion Timeout error, and instead handle the error via a device-specific mechanism that avoids data corruption.

It is strongly recommended that software not configure Unexpected Completion errors to be handled as Uncorrectable Errors. This avoids them triggering System Errors or hardware error containment mechanisms like Downstream Port Containment (DPC).

IMPLEMENTATION NOTE

No VF 10-Bit Tag Completer Supported Bit

There is no VF 10-Bit Tag Completer Supported bit. If a PF supports 10-Bit Tag Completer capability, then all of its associated VFs are required support 10-Bit Tag Completer capability as stated in Section 10.3.5.9. This helps avoid the complexity of PCIe hierarchies where some Completers support 10-Bit Tag capability and some do not.

9.3.3.2.4 VF Migration Interrupt Message Number

VF Migration Interrupt Message Number must contain the MSI or MSI-X vector number used for the interrupt message generated in association with certain VF migration events.

For MSI, VF Migration Interrupt Message Number must indicate the MSI message number [0 .. 31] used to reference certain SR events described in this section. The Function is required to update this field so that it is correct if the number of MSI Messages assigned to the Function changes when software writes to the Multiple Message Enable field in the MSI Message Control register.

For MSI-X, VF Migration Interrupt Message Number must indicate the MSI-X Table entry [0 .. 2047] used to generate the interrupt message.

If both MSI and MSI-X are implemented, they are permitted to use different vectors, though software must enable only one mechanism at a time. If both MSI and MSI-X interrupts are enabled, this field is undefined.

If VF Migration Capable is Clear, this field is undefined.

IMPLEMENTATION NOTE

Migration and MSI

If a PF implements MSI and software sets Multiple Message Enable to a value less than Multiple Message Capable, some sharing of MSI vectors may be occurring. This can create complicated software structures since a single vector might need to be directed to both SR-PCIM and the PF Device Driver.

9.3.3.3 *SR-IOV Control (08h)*

Table 9-5 defines the layout of the SR-IOV Control fields.

Table 9-5: SR-IOV Control

Bit Location	Register Description	Attributes
0	**VF Enable** – Enables/Disables VFs. Default value is 0b.	RW
1	**VF Migration Enable** – Enables/Disables VF Migration Support. Default value is 0b. See Section 9.3.3.3.2.	RW or RO (see description)
2	**VF Migration Interrupt Enable** – Enables/Disables VF Migration State Change Interrupt. Default value is 0b.	RW
3	**VF MSE** – Memory Space Enable for Virtual Functions. Default value is 0b.	RW
4	**ARI Capable Hierarchy** – **PCI Express Endpoint:** This bit must be RW in the lowest-numbered PF of the Device and hardwired to 0b in all other PFs. If the value of this bit is 1b, the Device is permitted to locate VFs in Function Numbers 8 to 255 of the captured Bus Number. Otherwise, the Device must locate VFs as if it were a non-ARI Device. This bit is not affected by FLR of any PF or VF. Default value is 0b. **RCiEP:** Not applicable – this bit must be hardwired to 0b. Within the Root Complex, VFs are always permitted to be assigned to any Function Number allowed by First VF Offset and VF Stride rules (see Sections 9.3.3.8 and 9.3.3.9)**.**	RW or RO (see description)
5	**VF 10-Bit Tag Requester Enable** – If Set, all VFs must use 10-Bit Tags for all Non-Posted Requests they generate. If Clear, VFs must not use 10-Bit Tags for Non-Posted Requests they generate. See Section 10.3.3.2.3. If software changes the value of this bit while any VFs have outstanding Non-Posted Requests, the result is undefined. If the VF 10-Bit Tag Requester Supported bit in the SR-IOV Capabilities register is Clear, this bit is permitted to be hardwired to 0b. Default value is 0b.	RW or RO
15:6	**Reserved** – These fields are currently Reserved.	RsvdP

9.3.3.3.1 VF Enable

VF Enable manages the assignment of VFs to the associated PF. If VF Enable is Set, the VFs associated with the PF are accessible in the PCI Express fabric. When Set, VFs respond to and may issue PCI Express transactions following the rules for PCI Express Endpoint Functions.

If VF Enable is Clear, VFs are disabled and not visible in the PCI Express fabric; requests to these VFs shall receive UR and these VFs shall not issue PCI Express transactions.

To allow components to perform internal initialization, after changing the VF Enable bit from Cleared to Set, the system is not permitted to issue Requests to the VFs which are enabled by that VF Enable bit until one of the following is true:

- At least 100 ms has passed
- An FRS Message has been received from the PF with a Reason Code of VF Enabled
- At least VF Enable Time has passed. VF Enable Time is either (1) the Reset Time value in the Readiness Time Reporting capability associated with the VF, or (2) a value determined by system software / firmware[143].

The Root Complex and/or system software must allow at least 1.0 s after Setting the VF Enable bit, before it may determine that a VF which fails to return a Successful Completion Status for a valid Configuration Request is broken. After Setting the VF Enable bit, the VFs enabled by that VF Enable bit are permitted to return a CRS status to configuration requests up to the 1.0 s limit, if they are not ready to provide a Successful Completion Status for a valid Configuration Request. After a PF transmits an FRS Message with a Reason Code of VF Enabled, no VF associated with that PF is permitted to return CRS without an intervening VF disable or other valid reset condition. After returning a Successful Completion to any Request, no VF is permitted to return CRS without an intervening VF disable or other valid reset condition.

Since VFs don't have an MSE bit (MSE in VFs is controlled by the VF MSE bit in the SR-IOV capability in the PF), it's possible for software to issue a Memory Request before the VF is ready to handle it. Therefore, Memory Requests must not be issued to a VF until at least one of the following conditions has been met:

- The VF has responded successfully (without returning CRS) to a Configuration Request.
- After issuing an FLR to the VF, at least one of the following is true:
    - At least 1.0 s has passed since the FLR was issued.
    - The VF supports FRS and, after the FLR was issued, an FRS Message has been received from the VF with a Reason Code of FLR Completed.
    - At least FLR Time has passed since the FLR was issued. FLR Time is either (1) the FLR Time value in the Readiness Time Reporting capability associated with the VF or (2) a value determined by system software / firmware[143].
- After Setting VF Enable in a PF, at least one of the following is true:
    - At least 1.0 s has passed since VF Enable was Set.

---

[143] For example, ACPI tables.

- The PF supports FRS and, after VF Enable was Set, an FRS Message has been received from the PF with a Reason Code of VF Enabled.
- At least VF Enable Time has passed since VF Enable was Set. VF Enable Time is either (1) the Reset Time value in the Readiness Time Reporting capability associated with the VF or (2) a value determined by system software / firmware[143].

The VF is permitted to silently drop Memory Requests after an FLR has been issued to the VF or VF Enable has been Set in the associated PF's SR-IOV capability until the VF responds successfully (without returning CRS) to any Request.

Clearing VF Enable effectively destroys the VFs. Setting VF Enable effectively creates VFs. Setting VF Enable after it has previously been Cleared shall result in a new set of VFs. If the PF is in the D0 power state, the new VFs are in the $D0_{uninitialized}$ state. If the PF is in a lower power state behavior is undefined (see Sections 9.6.1 and 9.6.2).

When Clearing VF Enable, a PF that supports FRS shall send an FRS Message with FRS Reason VF Disabled to indicate when this operation is complete. The PF is not permitted to send this Message if there are outstanding Non-Posted Requests issued by the PF or any of the VFs associated with the PF. The FRS Message may only be sent after these Requests have completed (or timed out).

After VF Enable is Cleared no field in the SR-IOV Extended Capability or the VF Migration State Array (see Section 9.3.3.15.1) may be accessed until either:

- ❑ At least 1.0 s has elapsed after VF Enable was Cleared.
- ❑ The PF supports FRS and after VF Enable was Cleared, an FRS Message has been received from the PF with a Reason Code of VF Disabled.

Section 9.3.3.7 NumVFs, Section 9.3.3.5 InitialVFs, Section 9.3.3.6 TotalVFs, Section 9.3.3.9 First VF Offset, Section 9.3.3.13 System Page Size, and Section 9.3.3.14 VF BARx describe additional semantics associated with this field.

9.3.3.3.2 VF Migration Enable

VF Migration Enable must be Set to allow VF Migration on this PF. See the *Multi-Root I/O Virtualization and Sharing Specification* for details.

If VF Migration Capable is Set, this bit is RW and the default value is 0b. Otherwise, this bit is RO Zero.

This field is RO when VF Enable is Set.

9.3.3.3.3 VF Migration Interrupt Enable

An MSI or MSI-X interrupt message must be sent every time the logical AND of the following conditions transitions from FALSE to TRUE:

- ❑ The VF's Bus Master Enable bit is Set.
- ❑ The associated vector is unmasked.
- ❑ The VF Migration Interrupt Enable bit is Set.
- ❑ The VF Migration Status bit is Set.

The MSI or MSI-X vector used is indicated by the VF Migration Interrupt Message Number field.

If VF Migration Capable is Clear, this field is undefined.

Note: VF Migration events are described in Section 9.2.4.

9.3.3.3.4 VF MSE (Memory Space Enable)

VF MSE controls memory space enable for all Active VFs associated with this PF, as with the Memory Space Enable bit in a Function's PCI Command register. The default value for this bit is 0b.

When VF Enable is Set, VF memory space will respond only when VF MSE is Set. VFs shall follow the same error reporting rules as defined in the *PCI Express Base Specification* if an attempt is made to access a Virtual Function's memory space when VF Enable is Set and VF MSE is Clear.

IMPLEMENTATION NOTE

VF MSE and VF Enable

VF memory space will respond with Unsupported Request when VF Enable is Clear. Thus, VF MSE is "don't care" when VF Enable is Clear; however, software may choose to Set VF MSE after programming the VF BARn registers, prior to Setting VF Enable.

9.3.3.3.5 ARI Capable Hierarchy

For Devices associated with an Upstream Port, ARI Capable Hierarchy is a hint to the Device that ARI has been enabled in the Root Port or Switch Downstream Port immediately above the Device. Software should set this bit to match the ARI Forwarding Enable bit in the Root Port or Switch Downstream Port immediately above the Device.

ARI Capable Hierarchy is only present in the lowest-numbered PF of a Device (for example PF_0) and affects all PFs of the Device. ARI Capable Hierarchy is Read Only Zero in other PFs of a Device.

A Device may use the setting of ARI Capable Hierarchy to determine the values for First VF Offset (see Section 9.3.3.9) and VF Stride (see Section 9.3.3.10). The effect of changing ARI Capable Hierarchy is undefined if VF Enable is Set in any PF. This bit must be set to its default value upon Conventional Reset. This bit is not affected by FLR of any PF or VF. If either ARI Capable Hierarchy Preserved is Set (see Section 9.3.3.2.2) or No_Soft_Reset is Set (see Section 9.6.2), a power state transition of this PF from $D3_{hot}$ to D0 does not affect the value of this bit (see Section 9.6.2).

ARI Capable Hierarchy does not apply to RCiEPs.

IMPLEMENTATION NOTE

ARI Capable Hierarchy

For a Device associated with an Upstream Port, that Device has no way of knowing whether ARI has been enabled. If ARI is enabled, the Device can conserve Bus Numbers by assigning VFs to Function Numbers greater than 7 on the captured Bus Number. ARI is defined in the *PCI Express Base Specification.*

Since RCiEPs are not associated with an Upstream Port, ARI does not apply, and VFs may be assigned to any Function Number within the Root Complex permitted by First VF Offset and VF Stride (see Sections 9.3.3.8 and 9.3.3.9).

9.3.3.4 *SR-IOV Status (0Ah)*

Table 9-69: defines the layout of the SR-IOV Status field.

Table 9-6: SR-IOV Status

Bit Location	Register Description	Attributes
0	**VF Migration Status** – Indicates a VF Migration In or Migration Out Request has been issued by MR-PCIM. To determine the cause of the event, software may scan the VF State Array. Default value is 0b.	RW1C
15:1	**Reserved** – These fields are currently Reserved.	RsvdZ

9.3.3.4.1 VF Migration Status

VF Migration Status is Set when VF Enable, VF Migration Capable, and VF Migration Enable are Set and certain state changes occur in a VF Migration State Array entry (see Section 9.2.4 and Section 9.3.3.15.1 for details).

This setting of VF Migration Status is not affected by the value of VF Migration Interrupt Enable or by any controls for MSI or MSI-X.

9.3.3.5 *InitialVFs (0Ch)*

InitialVFs indicates to SR-PCIM the number of VFs that are initially associated with the PF.

The minimum value of InitialVFs is 0.

For Devices operating in Single-Root mode, this field is HwInit and must contain the same value as TotalVFs.

For Devices operating in Multi-Root mode, the value of this field may be changed by MR-PCIM when VF Enable is Clear.

Note: The mechanism used by MR-PCIM to affect this field is described in the *Multi-Root I/O Virtualization and Sharing Specification.*

If VF Migration Enable is Set and VF Enable is Cleared and then Set, the value of InitialVFs may change. This is necessary since some VFs may have been migrated to other PFs and may no longer be available to this PF.

9.3.3.6 *TotalVFs (0Eh)*

TotalVFs indicates the maximum number of VFs that could be associated with the PF.

The minimum value of TotalVFs is 0.

For Devices operating in Single-Root mode, this field is HwInit and must contain the same value as InitialVFs.

For Devices operating in Multi-Root mode, the value of this field may be changed by MR-PCIM.

Note: The mechanism used by MR-PCIM to affect this field is described in the *Multi-Root I/O Virtualization and Sharing Specification*.

9.3.3.7 *NumVFs (10h)*

NumVFs controls the number of VFs that are visible. SR-PCIM sets NumVFs as part of the process of creating VFs. This number of VFs shall be visible in the PCI Express fabric after both NumVFs is set to a valid value and VF Enable is Set. A visible VF has a Function number reserved for it, but might not exist. If the VF exists, it shall respond to PCI Express transactions targeting them, and shall follow all rules defined by this specification. A VF exists if either:

- VF Migration Capable is Clear and the VF number is less than or equal to TotalVFs.
- VF Migration Capable is Set and the associated VF is in either the Active.Available or Dormant.MigrateIn state (see Section 9.2.4)

The results are undefined if NumVFs is set to a value greater than TotalVFs.

NumVFs may only be written while VF Enable is Clear. If NumVFs is written when VF Enable is Set, the results are undefined.

The initial value of NumVFs is undefined.

9.3.3.8 *Function Dependency Link (12h)*

The programming model for a Device may have vendor-specific dependencies between sets of Functions. The Function Dependency Link field is used to describe these dependencies.

This field describes dependencies between PFs. VF dependencies are the same as the dependencies of their associated PFs.

If a PF is independent from other PFs of a Device, this field shall contain its own Function Number.

If a PF is dependent on other PFs of a Device, this field shall contain the Function Number of the next PF in the same Function Dependency List. The last PF in a Function Dependency List shall contain the Function Number of the first PF in the Function Dependency List.

If PF *p* and PF *q* are in the same Function Dependency List, than any SI that is assigned VF *p,n* shall also be assigned to VF *q,n*.

IMPLEMENTATION NOTE

Function Dependency Link Example

Consider the following scenario:

SR-IOV Field	PF 0	PF 1	PF 2
Function Dependency Link	1	0	2
NumVFs	4	4	6
First VF Offset	4	4	4
VF Stride	3	3	3

Function Number	Description	Independent
0	PF 0	No
1	PF 1	No
2	PF 2	Yes
3	Function not present	
4	VF 0,1 (aka PF 0 VF 1)	No
5	VF 1,1 (aka PF 1 VF 1)	No
6	VF 2,1 (aka PF 2 VF 1)	Yes
7	VF 0,2	No
8	VF 1,2	No
9	VF 2,2	Yes
10	VF 0,3	No
11	VF 1,3	No
12	VF 2,3	Yes
13	VF 0,4	No
14	VF 1,4	No
15	VF 2,4	Yes
16 to 17	Functions not present	
18	VF 2,5	Yes
19 to 20	Functions not present	
21	VF 2,6	Yes
22 to 255	Functions not present	

In this example, Functions 4 and 5 must be assigned to the same SI. Similarly, Functions 7 and 8, 10 and 11, and 13 and 14 must be assigned together. If PFs are assigned to SIs, Functions 0 and 1 must be assigned together as well. Functions 2, 6, 9, 12, 15, 18, and 21 are independent and may be assigned to SIs in any fashion.

All PFs in a Function Dependency List shall have the same values for the InitialVFs, TotalVFs, and VF Migration Capable fields.

SR-PCIM shall ensure that all PFs in a Function Dependency List have the same values for the NumVFs, VF Enable, and VF Migration Enable fields before any VF in that Function Dependency List is assigned to an SI.

VF Migration and VF Mapping operations occur independently for every VF. SR-PCIM shall not assign a VF to an SI until it can assign all dependent VFs. For example, using the scenario above, if both VF 0,2 (Function 7) and VF 1,2 (Function 8) are in the Inactive.Unavailable or

Dormant.MigrateIn states, SR-PCIM shall not assign either VF to an SI until both VFs reach the Active.Available state.

Similarly, SR-PCIM shall not remove a VF from an SI until it can remove all dependent VFs. For example, using the scenario above, both VF 0,2 and VF 1,2 shall be removed from an SI only when they both reach the Active.MigrateOut state. SR-PCIM shall not transition the Functions to Inactive.Unavailable until the SI has stopped using all dependent Functions.

9.3.3.9 *First VF Offset (14h)*

First VF Offset is a constant and defines the Routing ID offset of the first VF that is associated with the PF that contains this Capability structure. The first VF's 16-bit Routing ID is calculated by adding the contents of this field to the Routing ID of the PF containing this field ignoring any carry, using unsigned, 16-bit arithmetic.

A VF shall not be located on a Bus Number that is numerically smaller than its associated PF.

This field may change value when the lowest-numbered PF's ARI Capable Hierarchy value changes or when this PF's NumVFs value changes.

Note: First VF Offset is unused if NumVFs is 0. If NumVFs is greater than 0, First VF Offset must not be zero.

9.3.3.10 *VF Stride (16h)*

VF Stride defines the Routing ID offset from one VF to the next one for all VFs associated with the PF that contains this Capability structure. The next VF's 16-bit Routing ID is calculated by adding the contents of this field to the Routing ID of the current VF, ignoring any carry, using unsigned 16-bit arithmetic.

This field may change value when the lowest-numbered PF's ARI Capable Hierarchy value changes or when this PF's NumVFs value changes.

Note: VF Stride is unused if NumVFs is 0 or 1. If NumVFs is greater than 1, VF Stride must not be zero.

9.3.3.11 *VF Device ID (1Ah)*

This field contains the Device ID that should be presented for every VF to the SI.

VF Device ID may be different from the PF Device ID. A VF Device ID must be managed by the vendor. The vendor must ensure that the chosen VF Device ID does not result in the use of an incompatible device driver.

9.3.3.12 *Supported Page Sizes (1Ch)*

This field indicates the page sizes supported by the PF. This PF supports a page size of 2^{n+12} if bit *n* is Set. For example, if bit 0 is Set, the PF supports 4-KB page sizes. PFs are required to support 4-KB, 8-KB, 64-KB, 256-KB, 1-MB, and 4-MB page sizes. All other page sizes are optional.

The page size describes the minimum alignment requirements for VF BAR resources as described in Section 9.3.3.13.

IMPLEMENTATION NOTE

Non-pre-fetch Address Space

Non-pre-fetch address space is limited to addresses below 4 GB. Pre-fetch address space in 32-bit systems is also limited. Vendors are strongly encouraged to utilize the System Page Size feature to conserve address space while also supporting systems with larger pages.

9.3.3.13 *System Page Size (20h)*

This field defines the page size the system will use to map the VFs' memory addresses. Software must set the value of the System Page Size to one of the page sizes set in the Supported Page Sizes field (see Section 9.3.3.12). As with Supported Page Sizes, if bit *n* is Set in System Page Size, the VFs associated with this PF are required to support a page size of 2^{n+12}. For example, if bit 1 is Set, the system is using an 8-KB page size. The results are undefined if System Page Size is zero. The results are undefined if more than one bit is set in System Page Size. The results are undefined if a bit is Set in System Page Size that is not Set in Supported Page Sizes.

When System Page Size is set, the VF associated with this PF is required to align all BAR resources on a System Page Size boundary. Each VF BAR*n* or VF BAR*n* pair (see Section 9.3.3.14) shall be aligned on a System Page Size boundary. Each VF BAR*n* or VF BAR*n* pair defining a non-zero address space shall be sized to consume an integer multiple of System Page Size bytes. All data structures requiring page size alignment within a VF shall be aligned on a System Page Size boundary.

VF Enable must be zero when System Page Size is written. The results are undefined if System Page Size is written when VF Enable is Set.

Default value is 0000 0001h (i.e., 4 KB).

9.3.3.14 *VF BAR0, VF BAR1, … VF BAR5 (24h … 38h)*

These fields must define the VF's Base Address Registers (BARs). These fields behave as normal PCI BARs, as described in Section 7.5.1. They can be sized by writing all 1s and reading back the contents of the BARs as described in Section 7.5.1.2.1, complying with the low order bits that define the BAR type fields.

These fields may have their attributes affected by the VF Resizable BAR capability (see Section 9.3.7.5) if it is implemented.

The amount of address space decoded by each BAR shall be an integral multiple of System Page Size.

Each VF BAR*n*, when "sized" by writing 1s and reading back the contents, describes the amount of address space consumed and alignment required by a single Virtual Function, per BAR. When written with an actual address value, and VF Enable and VF MSE are Set, the BAR maps NumVFs

BARs. In other words, the base address is the address of the first VF BAR*n* associated with this PF and all subsequent VF BAR*n* address ranges follow as described below.

VF BARs shall only support 32-bit and 64-bit memory space. PCI I/O Space is not supported in VFs. Bit 0 of any implemented VF BARx must be RO 0b except for a VF BARx used to map the upper 32 bits of a 64-bit memory VF BAR pair.

The alignment requirement and size read is for a single VF, but when VF Enable is Set and VF MSE is Set, the BAR contains the base address for all (NumVFs) VF BAR*n*.

The algorithm to determine the amount of address space mapped by a VF BAR*n* differs from the standard BAR algorithm as follows:

1. Resize the BAR via the VF Resizable BAR capability (see Section9.3.7.5) if it is implemented.
2. After reading the low order bits to determine if the BAR is a 32-bit BAR or 64-bit BAR pair, determine the size and alignment requirements by writing all 1s to VF BAR*n* (or VF BAR*n* and VF BAR*n+1* for a 64-bit BAR pair) and reading back the contents of the BAR or BAR pair. Convert the bit mask returned by the read(s) to a size and alignment value as described in Section 7.5.1.2.1. This value is the size and alignment for a single VF.
3. Multiply the value from step 1 by the value set in NumVFs to determine the total amount of space the BAR or BAR pair will map after VF Enable and VF MSE are Set.

For each VF BAR*n* field, *n* corresponds to one of the VFs BAR spaces. Table 9-7 shows the relationship between *n* and a Function's BAR.

Table 9-7: BAR Offsets

n	BAR Offset in a Type 0 Header
0	10h
1	14h
2	18h
3	1Ch
4	20h
5	24h

The contents of all VF BAR*n* registers are indeterminate after System Page Size is changed.

9.3.3.15 *VF Migration State Array Offset (3Ch)*

If VF Migration Capable (Section 9.3.3.2.1) is Set and TotalVFs (Section 9.3.3.6) is not zero, this register shall contain a PF BAR relative pointer to the VF Migration State Array. This register is RO Zero if VF Migration Capable is Clear. The VF Migration State Array is defined in Section 9.3.3.15.1. The layout of the VF Migration State Array Offset is defined in Table 9-8.

Table 9-8: VF Migration State Array Offset

Bit Location	Register Description	Attributes
2:0	**VF Migration State BIR** – Indicates which one of a Function's Base Address registers, located beginning at 10h in Configuration Space, is used to map the Function's VF Migration State Array into Memory Space. **BIR Value BAR** 0 BAR0 10h 1 BAR1 14h 2 BAR2 18h 3 BAR3 1Ch 4 BAR4 20h 5 BAR5 24h 6 Reserved 7 Reserved For a 64-bit BAR, the VF Migration State BIR indicates the lower DW. This field is undefined if TotalVFs is 0.	RO
31:3	**VF Migration State Offset** – Used as an offset from the address contained by one of the Function's Base Address registers to point to the base of the VF Migration State Array. The lower three VF Migration State BIR bits are masked off (set to zero) by software to form a 32-bit QW-aligned offset. This field is undefined if TotalVFs is 0.	RO

If a BAR that maps address space for the VF Migration State Array also maps other usable address space not associated with VF Migration, locations used in the other address space must not share any naturally aligned 8-KB address range with one where the VF Migration State Array resides.

9.3.3.15.1 VF Migration State Array

The VF Migration State Array is located using the VF Migration State Array Offset register of the SR-IOV Capability block.

The VF Migration State Array has a VF Migration State Entry for each VF. The total size of the VF Migration State array is NumVFs bytes. The VF Migration State Array shall not exist if TotalVFs is 0. Table 9-9 defines the layout of a VF Migration State Array entry.

Table 9-9: VF Migration State Entry

Bit Location	Register Description	Attributes
1:0	**VF Migration State** – State of the associated VF. This field is undefined when VF Enable is Clear. The initial values of the VF Migration State Array are described in Section 9.2.4.1.	RW
7:2	**Reserved** – These fields are currently Reserved.	RsvdP

VF Migration State contains the values shown in Table 9-10.

Table 9-10: VF Migration State Descriptions

VF State	VF Exists	Description
00b	No	**Inactive.Unavailable** – VF does not exist to SR nor is it being migrated in or out.
01b	No	**Dormant.MigrateIn** – VF is available for use by SR. VF exists but cannot initiate transactions.
10b	Yes	**Active.MigrateOut** – SR has been requested to relinquish use of the VF.
11b	Yes	**Active.Available** – Fully functional. Could be assigned to an SI.

The initial values of the VF Migration State Array are described in Section 9.2.4.1. The VF Migration State Array is returned to a configuration as described in Section 9.2.4.1 within 1.0 s after VF Enable is Cleared.

Software initiates a state transition by writing a new state value to the entry.

Valid state transitions are listed in Table 9-11 and Table 9-12 and shown in Figure 9-12. Only changes in Table 9-11 can be requested by SR. Changes in Table 9-12 are initiated by MR-PCIM using mechanisms described in the *Multi-Root I/O Virtualization and Sharing Specification* and have SR visible effects described here. Any transition issued by SR-PCIM and not listed in Table 9-11 is ignored and does not change the VF State.

Table 9-11: SR-PCIM Initiated VF Migration State Transitions

Current State	New State	Change Initiated By	SR Visible Effects of Change
Dormant.MigrateIn	Active.Available	SR-PCIM	**VF Activate** VF now exists.
Active.Available	Dormant.MigrateIn	SR-PCIM	**VF Deactivate** VF no longer exists.
Active.MigrateOut	Inactive.Unavailable	SR-PCIM	**VF Migrate Out Complete** VF no longer exists.

Table 9-12: MR-PCIM Initiated VF Migration State Transitions

Current State	New State	Change Initiated By	SR Visible Effects of Change
Active.Available	Active.MigrateOut	MR-PCIM	VF Migrate Out Request VF continues to exist. Sets VF Migration Status.
Inactive.Unavailable	Dormant.MigrateIn	MR-PCIM	VF Migrate In Request VF remains non-existent. Sets VF Migration Status.
Dormant.MigrateIn	Inactive.Unavailable	MR-PCIM	VF Migrate In Retract VF remains non-existent. Sets VF Migration Status.
Active.MigrateOut	Active.Available	MR-PCIM	VF Migrate Out Retract VF continues to exist. Sets VF Migration Status.

9.3.4 PF/VF Configuration Space Header

This section defines the requirements on PF and VF configuration space fields.

The register definitions listed throughout this chapter establish the mapping between existing PCI-SIG specifications and the PF/VF definitions for a PCIe SR-IOV-capable device.

9.3.4.1 *Type 0 Configuration Space Header*

Figure 9-14 details allocation for register fields of PCI Express Type 0 Configuration Space Header.

31			0	Byte Offset
Device ID		Vendor ID		00h
Status		Command		04h
Class Code			Revision ID	08h
BIST	Header Type	Master Latency Timer	Cache Line Size	0Ch
Base Address Registers				10h 14h 18h 1Ch 20h 24h
Cardbus CIS Pointer				28h
Subsystem ID		Subsystem Vendor ID		2Ch
Expansion ROM Base Address				30h
Reserved			Capabilities Pointer	34h
Reserved				38h
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	3Ch

OM14316

Figure 9-14: Type 0 Configuration Space Header

The PCI Express-specific interpretation of registers specific to PCI Express Type 0 Configuration Space Header is defined in this section.

Error Reporting fields are described in more detail in Section 9.4.

9.3.4.1.1 Vendor ID (Offset 00h)

This Read Only field identifies the manufacturer of the Device.

This field in all VFs returns FFFFh when read. VI software should return the Vendor ID value from the associated PF as the Vendor ID value for the VF.

9.3.4.1.2 Device ID (Offset 02h)

This Read Only field identifies the particular Device.

This field in all VFs returns FFFFh when read. VI software should return the VF Device ID (see Section 9.3.3.11) value from the associated PF as the Device ID for the VF.

IMPLEMENTATION NOTE

Legacy PCI Probing Software

Returning FFFFh for Device ID and Vendor ID values allows some legacy software to ignore VFs. See Section 7.5.1.1.1.

9.3.4.1.3 Command (Offset 04h)

PF and VF functionality is defined in Section 7.5.1.1.3 except where noted in Table 9-13. For VF fields marked RsvdP, the PF setting applies to the VF.

Table 9-13: Command Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
0	**I/O Space Enable** – Does not apply to VFs. Must be hardwired to 0b for VFs.	Base	0b
1	**Memory Space Enable** – Does not apply to VFs. Must be hardwired to 0b for VFs. VF Memory Space is controlled by the VF MSE bit in the VF Control register.	Base	0b
2	**Bus Master Enable** – Transactions for a VF that has its Bus Master Enable Set must not be blocked by transactions for VFs that have their Bus Master Enable Cleared.	Base	Base
6	**Parity Error Response**	Base	RsvdP
8	**SERR# Enable**	Base	RsvdP
10	**Interrupt Disable** – Does not apply to VFs.	Base	0b

9.3.4.1.4 Status (Offset 06h)

PF and VF functionality is defined in Section 7.5.1.1.4 except where noted in Table 9-14.

Table 9-14: Status Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
3	**Interrupt Status** – Does not apply to VFs.	Base	0b

9.3.4.1.5 Revision ID (Offset 08h)

This register specifies a device specific revision identifier.

The value reported in the VF may be different than the value reported in the PF.

9.3.4.1.6 Class Code (Offset 09h)

The Class Code register is read-only and is used to identify the generic function of the device and, in some cases, a specific register level programming interface. The field in the PF and associated VFs must return the same value when read.

9.3.4.1.7 Cache Line Size (Offset 0Ch)

This field is implemented by PCI Express devices as a read-write field for legacy compatibility purposes but has no effect on any PCI Express device behavior. Physical Functions continue to implement this field as RW. For Virtual Functions, this field is RO Zero.

9.3.4.1.8 Latency Timer (Offset 0Dh)

This field does not apply to PCI Express. This register must be RO Zero.

9.3.4.1.9 Header Type (Offset 0Eh)

This byte identifies the layout of the second part of the predefined header (beginning at byte 10h in Configuration Space) and also whether or not the device contains multiple Functions. Bit 7 in this register is used to identify a Multi-Function Device. For an SR-IOV device, bit 7 in this register is only Set if there are multiple Functions. VFs do not affect the value of bit 7. Bits 6 through 0 identify the layout of the second part of the predefined header. For VFs, this register must be RO Zero.

9.3.4.1.10 BIST (Offset 0Fh)

VFs shall not support BIST and must define this field as RO Zero.

If VF Enable is turned on in any PF of a Device, then software must not invoke BIST in any Function associated with that Device.

9.3.4.1.11 Base Address Registers (Offset 10h, 14h, ... 24h)

For VFs, the values in these registers are RO Zero.

Note: See also Section 9.2.1.1.1 and Section 9.3.3.14.

9.3.4.1.12 Cardbus CIS Pointer (Offset 28h)

For VFs, this register is not used and shall be RO Zero.

9.3.4.1.13 Subsystem Vendor ID (Offset 2Ch)

This Read Only field identifies the manufacturer of the subsystem. The field in the PF and associated VFs must return the same value when read.

9.3.4.1.14 Subsystem ID (Offset 2Eh)

This Read Only field identifies the particular subsystem. The Device may have a different value in the PF and the VF.

9.3.4.1.15 Expansion ROM BAR (Offset 30h)

The Expansion ROM BAR may be implemented in PFs. The expansion ROM BAR is RO Zero in VFs. The VI may choose to provide access to the PF Expansion ROM BAR for VFs via emulation.

ROM BAR shared decoding, defined in Section 7.5.1.2.4, is not permitted in any SR-IOV Device.

9.3.4.1.16 Capabilities Pointer (Offset 34h)

No differences from the description in Section 7.5.1.1.11.

9.3.4.1.17 Interrupt Line (Offset 3Ch)

This field does not apply to VFs. This field must be RO Zero.

9.3.4.1.18 Interrupt Pin (Offset 3Dh)

This field does not apply to VFs. This field must be RO Zero.

9.3.4.1.19 Min_Gnt/Max_Lat (Offset 3Eh/3Fh)

These registers do not apply to PCI Express. They must be RO Zero.

9.3.5 PCI Express Capability

PCI Express defines a Capability structure (see Section 7.5.2) for identification of a PCI Express device and indicates support for new PCI Express features.

Figure 7-20details allocation of register fields in the PCI Express Capability Structure. PFs and VFs are required to implement this capability subject to the exceptions and additional requirements described below.

9.3.5.1 *PCI Express Capability List Register (Offset 00h)*

The PCI Express Capability List register is described in Section 7.5.3.2 and the functionality described there applies to both the PF and VF.

9.3.5.2 *PCI Express Capabilities Register (Offset 02h)*

The PCI Express Capabilities register identifies PCI Express device type and associated capabilities.

The PF and VF functionality is defined in Section 7.5.3.2.

9.3.5.3 *Device Capabilities Register (Offset 04h)*

The Device Capabilities register identifies PCI Express device specific capabilities.
Figure 7-23details allocation of register fields in the Device Capabilities register; Table 9-15 provides the respective bit definitions.

PF and VF functionality is defined in Section 7.5.3.3 except where noted in Table 9-15.

Table 9-15: Device Capabilities Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
4:3	**Phantom Functions Supported** – When the VF Enable is Set, use of Phantom Function numbers by this PF and associated VFs is not permitted and this field must return 00b when read.	Base	00b
25:18	**Captured Slot Power Limit Value**	Base	undefined
27:26	**Captured Slot Power Limit Scale**	Base	undefined
28	**Function Level Reset Capability** – Required for PFs and for VFs.	1b	1b

9.3.5.4 *Device Control Register (Offset 08h)*

The Device Control register controls PCI Express device specific parameters. Figure 7-24 details allocation of register fields in the Device Control register; Table 9-16 provides the respective bit definitions.

PF and VF functionality is defined in Section 7.5.3.4 except where noted in Table 9-16. For VF fields marked RsvdP, the PF setting applies to the VF.

Table 9-16: Device Control Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
0	**Correctable Error Reporting Enable** See Section 9.4 for details on error reporting.	Base	RsvdP
1	**Non-Fatal Error Reporting Enable** See Section 9.4 for details on error reporting.	Base	RsvdP
2	**Fatal Error Reporting Enable** See Section 9.4 for details on error reporting.	Base	RsvdP
3	**Unsupported Request Reporting Enable** See Section 9.4 for details on error reporting.	Base	RsvdP
4	**Enable Relaxed Ordering**	Base	RsvdP
7:5	**Max_Payload_Size**	Base	RsvdP
8	**Extended Tag Field Enable**	Base	RsvdP
9	**Phantom Functions Enable**	Base	RsvdP
10	**Aux Power PM Enable**	Base	RsvdP
11	**Enable No Snoop**	Base	RsvdP
14:12	**Max_Read_Request_Size**	Base	RsvdP
15	**Initiate Function Level Reset** – Required for PFs and for VFs. **Note:** Setting Initiate Function Level Reset in a PF resets VF Enable which means that VFs no longer exist after the FLR is complete.	Base	Base

9.3.5.5 *Device Status Register (Offset 0Ah)*

The Device Status register provides information about PCI Express device specific parameters. Figure 7-25 details allocation of register fields in the Device Status register; Table 9-17 provides the respective bit definitions.

PF and VF functionality is defined in Section 7.5.3.5 except where noted in Table 9-17.

Table 9-17: Device Status Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
4	AUX Power Detected	Base	0b
6	Emergency Power Reduction Detected	Base	0b

9.3.5.6 *Link Capabilities Register (Offset 0Ch)*

The Link Capabilities register identifies PCI Express Link specific capabilities. Figure 7-26 details allocation of register fields in the Link Capabilities register.

PF and VF functionality is defined in Section 7.5.3.6.

9.3.5.7 *Link Control Register (Offset 10h)*

The Link Control register controls PCI Express Link specific parameters. Figure 7-27 details allocation of register fields in the Link Control register.

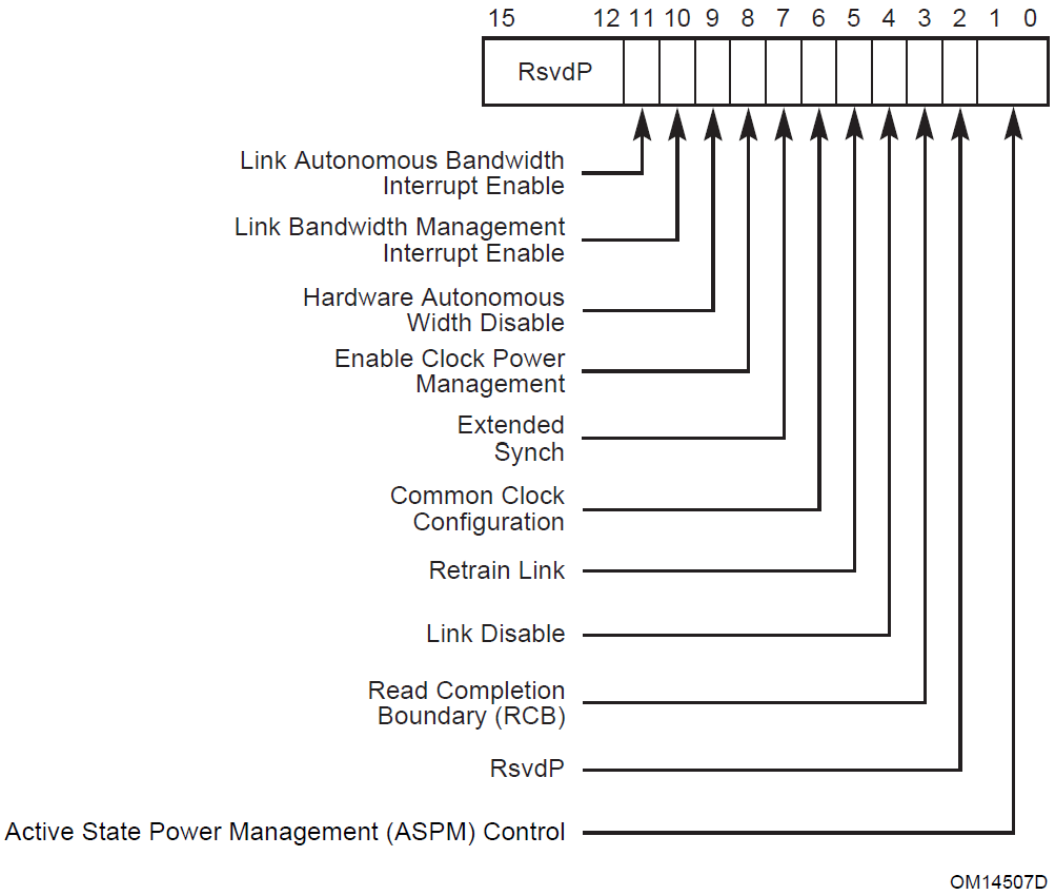


Figure 9-15: Link Control Register

PF and VF functionality is defined in Section 7.5.3.7 except where noted in Table 9-18: . For VF fields marked RsvdP, the PF setting applies to the VF.

Table 9-18: Link Control Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
1:0	**Active State Power Management (ASPM) Control**	Base	RsvdP

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
3	**Read Completion Boundary (RCB)** Must be hardwired to 0b for VFs.	Base	RsvdP
6	**Common Clock Configuration**	Base	RsvdP
7	**Extended Synch**	Base	RsvdP
8	**Enable Clock Power Management**	Base	RsvdP
9	**Hardware Autonomous Width Disable**	Base	RsvdP

9.3.5.8 *Link Status Register (Offset 12h)*

The Link Status register provides information about PCI Express Link specific parameters. Figure 7-28 details allocation of register fields in the Link Status register.

PF functionality is defined in Section 7.5.3.8. For the VF, all fields in this register are RsvdZ and the PF setting applies to the VF.

9.3.5.9 *Device Capabilities 2 Register (Offset 24h)*

PF and VF functionality is defined in Section 7.5.3.15 except as noted in Table 9-19.

Table 9-19: Device Capabilities 2 Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
3:0	**Completion Timeout Ranges Supported** VF value must be identical to PF value.	Base	Base
4	**Completion Timeout Disable Supported** VF value must be identical to PF value.	Base	Base
6	**AtomicOp Routing Supported** Not applicable to Endpoints.	RsvdP	RsvdP
7	**32-bit AtomicOp Completer Supported** VF value must be identical to PF value.	Base	Base
8	**64-bit AtomicOp Completer Supported** VF value must be identical to PF value.	Base	Base
9	**128-bit CAS Completer Supported** VF value must be identical to PF value.	Base	Base
16	**10-Bit Tag Completer Supported** VF value must be identical to PF value.	Base	Base

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
17	**10-Bit Tag Requester Supported** VF value must equal the value of the VF 10-Bit Tag Requester Supported bit in the SR-IOV Capabilities register. See Section 9.3.3.2.3	Base	Base
25:24	**Emergency Power Reduction Supported** VF value must be identical to PF value.	Base	Base
26	**Emergency Power Reduction Initialization Required** VF value must be identical to PF value.	Base	Base

9.3.5.10 *Device Control 2 Register (Offset 28h)*

PF and VF functionality is defined in Section 7.5.3.16 except as noted in Table 9-20.

Table 9-20: Device Control 2 Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
3:0	**Completion Timeout Value** The PF value applies to all associated VFs.	Base	RsvdP
4	**Completion Timeout Disable** The PF value applies to all associated VFs.	Base	RsvdP
6	**AtomicOp Requester Enable** The PF value applies to all associated VFs.	Base	RsvdP
8	**IDO Request Enable** The PF value applies to all associated VFs.	Base	RsvdP
9	**IDO Completion Enable** The PF value applies to all associated VFs.	Base	RsvdP
11	**Emergency Power Reduction Request** This bit is only present in one Function associated with an Upstream Port. That Function can never be a VF.	Base	RsvdP
12	**10-Bit Tag Requester Enable** The value in the VF 10-Bit Tag Requester Enable bit in the SR-IOV Control register applies to all associated VFs.	Base	RsvdP

9.3.5.11 *Device Status 2 Register (Offset 2Ah)*

PF and VF functionality is defined in Section 7.5.3.17.

9.3.5.12 *Link Capabilities 2 Register (Offset 2Ch)*

PF and VF functionality is defined in Section 7.5.3.18.

9.3.5.13 *Link Control 2 Register (Offset 30h)*

PF and VF functionality is defined in Section 7.5.3.19.

9.3.5.14 *Link Status 2 Register (Offset 32h)*

PF and VF functionality is defined in Section 7.5.3.20 except where noted in Table 9-21. The VF fields marked RsvdZ use the value of the associated PF.

Table 9-21. Link Status 2 Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
0	**Current De-emphasis Level**	Base	RsvdZ

9.3.6 PCI Standard Capabilities

SR-IOV usage of PCI Standard Capabilities is described in **Table** 9-22. Items marked n/a are not applicable to PFs or VFs.

Table 9-22: SR-IOV Usage of PCI Standard Capabilities

Capability ID	Description	PF Attributes	VF Attributes
00h	Null Capability	Base	Base
01h	PCI Power Management Interface	Base	Optional. See Section 9.6.
02h	AGP	n/a	n/a
03h	VPD	Base	Optional. See Section 9.3.6.1.
04h	Slot Identification	n/a	n/a
05h	MSI	Base	See Section 9.5.1.1.
06h	CompactPCI Hot Swap	n/a	n/a
07h	PCI-X	n/a	n/a
08h	HyperTransport	n/a	n/a
09h	Vendor-specific	Base	Base
0Ah	Debug Port	Base	Base
0Bh	CompactPCI Central Resource Control	n/a	n/a
0Ch	PCI Hot Plug	Base	n/a
0Dh	PCI Bridge Subsystem ID	n/a	n/a

Capability ID	Description	PF Attributes	VF Attributes
0Eh	AGP 8x	n/a	n/a
0Fh	Secure Device	n/a	n/a
10h	PCI Express	Base	See Section 9.3.5.
11h	MSI-X	See Section 9.5.1.2 and Section 9.5.1.3.	See Section 9.5.1.2 and Section 9.5.1.3.
12h	Serial ATA Data/Index Configuration	Base	n/a
13h	Advanced Features	n/a	n/a
14h	Enhanced Allocation	Base	Must not implement.
29h	Native PCIe Enclosure Management (NPEM)	Base	Not implemented

9.3.6.1 *VPD Capability*

The VPD Capability is optional in PCI. It remains optional in SR-IOV.

VFs and PFs that implement the VPD Capability must ensure that there can be no "data leakage" between VFs and/or PFs via the VPD Capability.

9.3.7 PCI Express Extended Capabilities

SR-IOV usage of PCI Express Extended Capabilities is described in Table 9-23. Items marked n/a are not applicable to PFs or VFs (e.g., for Capabilities only present in Root Complexes or only present in Function 0).

Table 9-23: SR-IOV Usage of PCI Express Extended Capabilities

Extended Capability ID	Description	PF Attributes	VF Attributes
0000h	Null Capability	Base	Base
0001h	Advanced Error Reporting (AER)	Base	See Section 9.4.2.
0002h	Virtual Channel (02h)	Base	Must not implement. See Section 9.3.7.1.
0003h	Device Serial Number	Base	See Section 9.3.7.2
0004h	Power Budgeting	Base	Must not implement. See Section 9.3.7.3
0005h	Root Complex Link Declaration	n/a	n/a
0006h	Root Complex Internal Link Control	n/a	n/a
0007h	Root Complex Event Collector Endpoint Association	n/a	n/a

Extended Capability ID	Description	PF Attributes	VF Attributes
0008h	Multi Function Virtual Channel	Base	Must not implement. See Section 9.3.7.1.
0009h	Virtual Channel (09h)	Base	Must not implement. See Section 9.3.7.1.
000Ah	Root Complex Register Block Header (RCRB)	n/a	n/a
000Bh	Vendor-specific	Base	Base
000Ch	Configuration Access Correlation	n/a	n/a
000Dh	Access Control Services (ACS)	See Section 9.3.7.6.	See Section 9.3.7.6.
000Eh	Alternative Routing ID Interpretation (ARI)	See Section 9.3.7.7.	See Section 9.3.7.7.
000Fh	Address Translation Services (ATS)	See Section 9.3.7.8.	See Section 9.3.7.8.
0010h	Single Root I/O Virtualization (SR-IOV)	See Section 9.3.3.	Must not implement. See Section 9.3.3
0011h	Multi-Root I/O Virtualization (MR-IOV)	Must not implement. See Section 9.3.7.9.	Must not implement. See Section 9.3.7.9.
0012h	Multicast	See Section 9.3.7.10.	See Section 9.3.7.10.
0013h	Page Request Interface (PRI)	See Section 9.3.7.11.	See Section 9.3.7.11
0014h	Reserved for AMD	Base	Base
0015h	Resizable BAR	Base	Must not implement. See Section 9.3.7.4
0016h	Dynamic Power Allocation (DPA)	See Section 9.3.7.12.	Must not implement. See Section 9.3.7.12.
0017h	TLP Processing Hints (TPH)	See Section 9.3.7.13.	See Section 9.3.7.13
0018h	Latency Tolerance Reporting (LTR)	Base	Must not implement. LTR is controlled using Function 0 which is never a VF.
0019h	Secondary PCI Express	Base	Must not implement. 8.0 GT/s is controlled using Function 0 which is never a VF.
001Ah	Protocol Multiplexing (PMUX)	Base	Must not implement. PMUX is controlled using Function 0 which is never a VF.
001Bh	Process Address Space ID (PASID)	Base	See Section 9.3.7.14
001Ch	LN Requester (LNR)	Base	Base
001Dh	Downstream Port Containment (DPC)	n/a	n/a.

Extended Capability ID	Description	PF Attributes	VF Attributes
001Eh	L1 PM Substates	Base	Must not implement. L1 PM Substates is controlled using Function 0 which is never a VF.
001Fh	Precision Time Measurement (PTM)	Base	Must not implement. PTM controls the Port and must not be implemented in a VF.
0020h	PCI Express over M-PHY (M-PCIe)	Base	Must not implement. M-PHY is controlled using Function 0 which is never a VF.
0021h	FRS Queueing	n/a	n/a
0022h	Readiness Time Reporting	Base	See Section 9.3.7.15
0023h	Designated vendor-specific	Base	Base
0024h	VF Resizable BAR	See Section 9.3.7.5	Not Implemented. See Section 9.3.7.5
0025h	Data Link Feature	Base	Must not implement.
0026h	Physical Layer 16.0 GT/s	Base	Must not implement.
0027h	Physical Layer 16.0 GT/s Margining	Base	Must not implement.
0028h	Hierarchy ID	Base	Base

9.3.7.1 *Virtual Channel/MFVC*

VFs use the Virtual Channels of the associated PFs. As such, VFs do not contain any Virtual Channel Capabilities.

VFs shall not contain the MFVC Capability. This Capability is only present in Function 0 which shall never be a VF.

9.3.7.2 *Device Serial Number*

The Device Serial Number Extended Capability may be present in PFs. If a PF contains the capability, its value applies to all associated VFs. VFs are permitted but not recommended to implement this capability. VFs that implement this capability must return the same Device Serial Number value as that reported by their associated PF.

9.3.7.3 *Power Budgeting*

The Power Budgeting Extended Capability may be present in PFs, but VFs must not implement it. If a PF contains the capability, it must report values that cover all associated VFs.

9.3.7.4 *Resizable BAR*

The Resizable BAR Extended Capability may be present in PFs. Since VFs do not implement standard BARs the capability must not be present in a VF. The PF's Resizable BAR settings do not affect any settings in the SR-IOV capability.

9.3.7.5 *VF Resizable BAR Capability*

The VF Resizable BAR capability is permitted to be implemented only in PFs that implement at least one VF BAR, and affects the size and base of a PF's VF BARs. Since VFs do not implement the BARs themselves the capability must not be present in a VF. A PF may implement both a VF Resizable BAR capability and a Resizable BAR capability, as each capability operates independently.

The VF Resizable BAR capability is an optional capability that permits PFs to be able to have their VF's BARs resized. The VF Resizable BAR capability permits hardware to communicate the resource sizes that are acceptable for operation via the VF Resizable BAR Capability and Control registers and system software to communicate the optimal size back to the hardware via the VF BAR Size field of the VF Resizable BAR Control register.

Hardware immediately reflects the size inference in the read-only bits of the appropriate VF BAR. The size inferred is the greater of the values decoded from the System Page Size and VF BAR Size fields. Hardware must Clear any bits that change from read-write to read-only, so that subsequent reads return zero. Software must clear the VF MSE bit in the SR-IOV Control register before writing the VF BAR Size field. After writing the VF BAR Size field, the contents of the corresponding VF BAR are undefined. To ensure that it contains a valid address after resizing the VF BAR, system software must reprogram the VF BAR, and Set the VF MSE bit (unless the resource is not allocated).

The VF Resizable BAR Capability and Control registers are permitted to indicate the ability to operate at 4 GB or greater only if the associated VF BAR is a 64-bit BAR.

It is strongly recommended that a Function not advertise any supported VF BAR sizes in its VF Resizable BAR Capability and Control registers that are larger than the space it would effectively utilize if allocated.

IMPLEMENTATION NOTE

Using the Capability During Resource Allocation

System software uses this capability in a similar way to the Resizable BAR capability. System software must first configure the System Page Size register (see Section 9.2.1.1.1). Potential usable memory aperture sizes are reported by the PF, and read, from the VF Resizable BAR Capability and Control registers. It is intended that the software allocate the largest of the reported sizes that it can, since allocating less address space than the largest reported size can result in lower performance. Software then writes the size to the VF Resizable BAR Control register for the appropriate VF BAR for the Function. Following this, the base address is written to the VF BAR.

For interoperability reasons, it is possible that hardware will set the default size of the VF BAR to a low size; a size lower than the largest reported in the VF Resizable BAR Capability register. Software that does not use this capability to size resources will likely result in sub-optimal resource allocation, where the resources are smaller than desirable, or not allocatable because there is no room for them. It is recommended that system software responsible for allocating resources in a resource constrained environment distribute the limited address space to all memory-mapped hardware, including system RAM, appropriately.

The VF Resizable BAR Capability structure defines a PCI Express Extended Capability which is located in PCI Express Extended Configuration Space, that is, above the first 256 bytes, and is shown below in Figure 9-16. This structure allows PFs with this capability to be identified and controlled. A Capability register and a Control register are implemented for each VF BAR that is resizable. Since a maximum of 6 VF BARs may be implemented by any PF, the VF Resizable BAR Capability structure can range from 12 bytes long (for a single VF BAR) to 52 bytes long (for all 6 VF BARs).

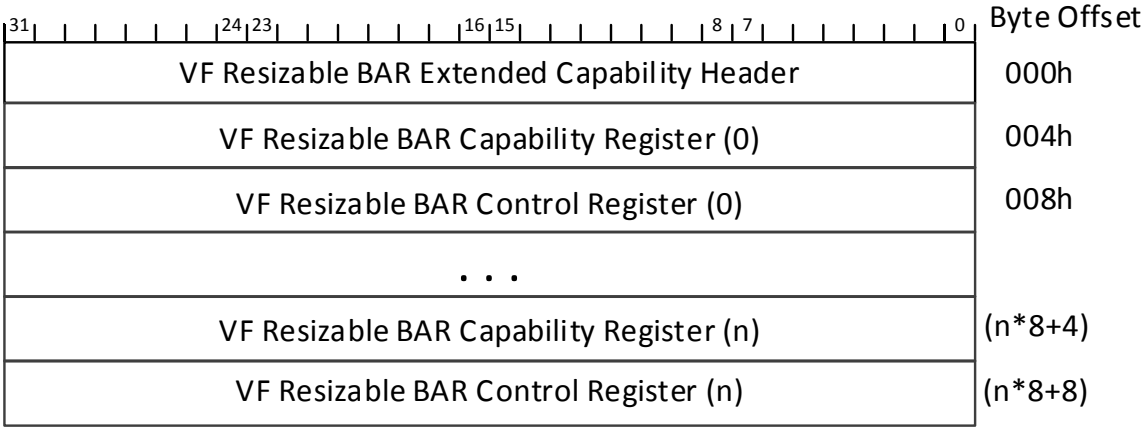


Figure 9-16: VF Resizable BAR Capability

9.3.7.5.1 VF Resizable BAR Extended Capability Header (Offset 00h)

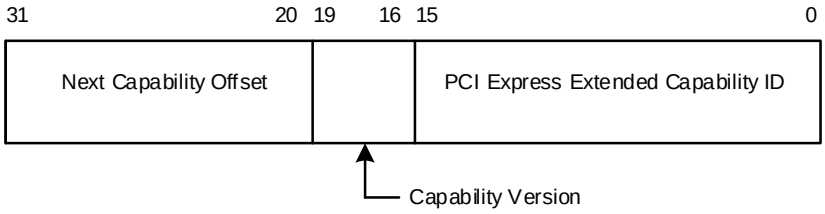


Figure 9-17: VF Resizable BAR Extended Capability Header

Table 9-24: VF Resizable BAR Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the extended capability. PCI Express Extended Capability ID for the VF Resizable BAR Capability is 0024h.	RO
19.16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the capability structure present. Must be 1h for this version of the specification.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express Extended Capability structure or 000h if no other items exist in the linked list of capabilities	RO

9.3.7.5.2 VF Resizable BAR Capability Register (Offset 04h)

The VF Resizable BAR Capability register field descriptions are the same as the definitions in the Resizable BAR Capability register in Table 7-102. Where those descriptions say 'BAR', this register's description is for 'VF BAR'. Where those descriptions say 'Function', this register's description is for 'PF'. Otherwise the field descriptions, the number of bits, their positions, and their attributes are the same. Consequently Figure 7-129 similarly allocates the register fields in this register.

9.3.7.5.3 VF Resizable BAR Control Register (Offset 08h)

The VF Resizable BAR Control register bits 31:16 follow the same definitions as the Resizable BAR Control register in Table 7-103.

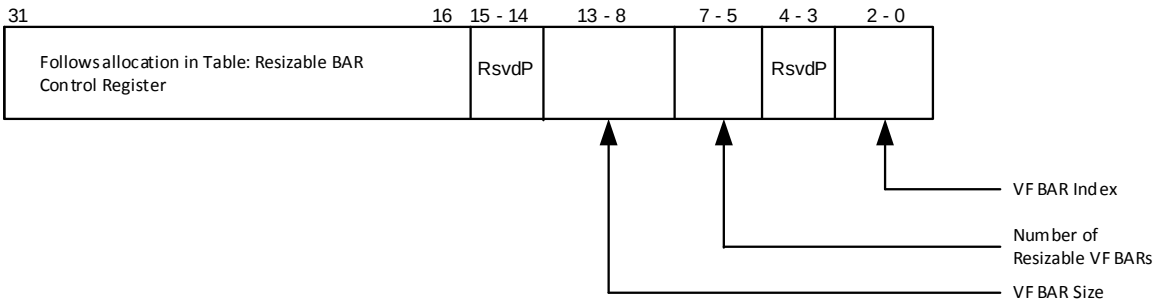


Figure 9-18: VF Resizable BAR Control Register

Table 9-25: VF Resizable BAR Control Register

Bit Location	Register Description	Attributes
2:0	**VF BAR Index** – This encoded value points to the beginning of this particular VF BAR located in the SR-IOV Capability. 0 = VF BAR located at offset 24h 1 = VF BAR located at offset 28h 2 = VF BAR located at offset 2Ch 3 = VF BAR located at offset 30h 4 = VF BAR located at offset 34h 5 = VF BAR located at offset 38h All other encodings are reserved. For a 64-bit Base Address register, the VF BAR Index indicates the lower DWORD. This value indicates which VF BAR supports a negotiable size.	RO
7:5	**Number of VF Resizable BARs** – Indicates the total number of resizable VF BARs in the capability structure for the Function. See Figure 9-16. The value of this field must be in the range of 01h to 06h. The field is valid in VF Resizable BAR Control register (0) (at offset 008h), and is RsvdP for all others.	RO/RsvdP
13:8	**VF BAR Size** – This is an encoded value. 0 = 1 MB (2^{20}) 1 = 2 MB (2^{21}) 2 = 4 MB (2^{22}) 3 = 8 MB (2^{23}) … 43 = 8 EB (2^{63}) The default value of this field is equal to the default size of the address space that the VF BAR resource is requesting via the VF BAR's read-only bits. Software must only write values that correspond to those indicated as supported in the VF Resizable BAR Capability and Control registers. Writing an unsupported value will produce undefined results. When this register field is programmed, the value is immediately reflected in the size of the resource, as encoded in the number of read-only bits in the VF BAR.	RW
31:16	The descriptions for these bits are the same as the definitions in the Resizable BAR Control register in Table 7-103. Where those descriptions say 'BAR', this register's description is for 'VF BAR'. Where those descriptions say 'Function', this register's description is for 'PF'.	See Table 7-103

9.3.7.6 *Access Control Services (ACS) Extended Capability*

ACS is an optional extended capability. If an SR-IOV Capable Device other than one in a Root Complex implements internal peer-to-peer transactions, ACS is required with additional requirements described below.

PF and VF functionality is defined in Section 7.7.7.2 except where noted in Table 9-26.

All Functions in SR-IOV Capable Devices (Devices that implement at least one PF) other than RCiEPs that support peer-to-peer transactions within the Device shall implement ACS Egress Control.

Implementation of ACS in RCiEPs is permitted but not required. It is explicitly permitted that, within a single Root Complex, some RCiEPs implement ACS and some do not. It is strongly recommended that Root Complex implementations ensure that all accesses originating from RCiEPs (PFs and VFs) without ACS capability are first subjected to processing by the Translation Agent (TA) in the Root Complex before further decoding and processing. The details of such Root Complex handling are outside the scope of this specification.

Table 9-26: ACS Capability Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
5	**ACS P2P Egress Control (E)** – Required to be 1b for Functions that are not RCiEPs if peer-to-peer transactions within the Device are supported.	Base	Base

IMPLEMENTATION NOTE

Access Control Services in Systems that Support Direct Assignment of Functions

General-purpose VIs typically have separate address spaces for each SI and for the VI itself. If such a VI also supports direct assignment of a Function to an SI, Untranslated Memory Request transactions issued by directly assigned Functions are under the complete control of software operating within the associated SI and typically reference the address space associated with that SI. In contrast, Memory Request transactions issued by the Host (MMIO requests) and by Functions that are not directly assigned are under the control of the VI and typically reference one or more system address spaces (e.g., the PCIe physical address space, the address space associated with the VI, or the address space associated with some designated SI). General-purpose VIs are not expected to establish a dependency between these various address spaces. Consequently, these address spaces may freely overlap which could lead to unintended routing of TLPs by Switches. For example, Upstream Memory Request TLPs originated by a directly assigned Function and intended for main memory could instead be routed to a Downstream Port if the address in the Request falls within the MMIO address region associated with that Downstream Port. Such unintended routing poses a threat to SI and/or VI stability and integrity.

To guard against this concern, vendors are strongly recommended to implement ACS in platforms that support general purpose VIs with direct assignment of Functions. Such support should include:

- In Switches or Root Complexes located below the TA, the level of ACS support should follow the guidelines established in this document for Downstream Switch Ports that implement an ACS Extended Capability structure.

Note: Components located above the TA only see Translated Memory Requests, consequently this concern does not apply to those components.

- In SR-IOV devices that are capable of peer-to-peer transactions, ACS support is required.
- In Multi-Function Devices that are capable of peer-to-peer transactions, vendors are strongly recommended to implement ACS with ACS P2P Egress Control.

Additionally, platform vendors should test for the presence of ACS and enable it in Root Complexes and Switches on the path from the TA to a Function prior to directly assigning that Function. If the Function is peer-to-peer capable, ACS should be enabled in the Function as well.

9.3.7.7 *Alternative Routing ID Interpretation Extended Capability (ARI)*

ARI is not applicable to RCiEPs; all other SR-IOV Capable Devices (Devices that include at least one PF) shall implement the ARI Capability in each Function. PF and VF functionality is defined in Section 7.8.7.2 except where noted in Table 9-27.

Table 9-27: ARI Capability Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
0	**MFVC Function Groups Capability (M)** – Additional requirements described below.	Base	Base
1	**ACS Function Groups Capability (A)** – Additional requirements described below.	Base	Base
15:8	**Next Function Number** – VFs are located using First VF Offset (see Section 9.3.3.10) and VF Stride (see Section 9.3.3.11).	Base	Undefined

Any Device that implements the MFVC Capability with the optional Function Arbitration Table and consumes more than one Bus Number shall implement the MVFC Function Groups Enable (M) capability bit as 1b in Function 0.

Any Device that implements the ACS Capability with the optional Egress Control Vector and consumes more than one Bus Number shall implement the ACS Function Groups Enable (A) capability bit as 1b in Function 0.

9.3.7.8 *Address Translation Services Extended Capability (ATS)*

ATS support is optional in SR-IOV devices. If a VF implements an ATS capability, its associated PF must implement an ATS capability. The ATS Capabilities in VFs and their associated PFs may be enabled independently.

PF and VF functionality is defined in Section 10.5.1 except where noted in Table 9-28 and Table 9-29. Attributes shown as ATS are the same as specified in Address Translation Services (Chapter 10).

Table 9-28: ATS Capability Register

Bit Location	PF and VF Register Differences From ATS	PF Attributes	VF Attributes
4:0	Invalidate Queue Depth Hardwired to 0 for VFs. Depth of shared PF input queue.	ATS	RO

Table 9-29: ATS Control Register

Bit Location	PF and VF Register Differences From ATS	PF Attributes	VF Attributes
4:0	Smallest Translation Unit (STU) Hardwired to 0 for VFs. PF value applies to all VFs.	ATS	RO

ATS behavior is specified in Address Translation Services (Chapter 10). ATS requests target the Function being invalidated. ATS responses will have a Routing ID field matching the targeted Function. Each Function is required to implement sufficient queuing to ensure it can hold the maximum number of outstanding Invalidation Requests from a TA (using either input or output queuing).

However, all VFs associated with a PF share a single input queue in the PF. To implement Invalidation flow control, the TA must ensure that the total number of outstanding Invalidate Requests to the shared PF queue (targeted to the PF and its associated VFs) does not exceed the value in the PF Invalidate Queue Depth field.

9.3.7.9 *MR-IOV*

PFs and VFs shall not contain an MR-IOV Capability.

If present, the MR-IOV Capability is contained in other Functions of the Component. See the *Multi-Root I/O Virtualization and Sharing Specification* for details.

9.3.7.10 *Multicast*

Multicast support is optional in SR-IOV devices. If a VF implements a Multicast capability, its associated PF must implement a Multicast capability. PF and VF functionality is defined in Section 7.9.11 except where noted in **Table 9-30**, **Table 9-31**, and **Table 9-32**.

Table 9-30: Multicast Capability Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
5:0	**MC_Max_Group** – PF value applies to all VFs.	Base	RsvdP
13:8	**MC_Window_Size_Requested** – PF value applies to all VFs.	Base	RsvdP

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
15	**MC_ECRC_Regeneration_Supported** – Not applicable for Endpoints.	Base	RsvdP

Table 9-31: Multicast Control Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
5:0	**MC_Num_Group** – PF value applies to all VFs.	Base	RsvdP

Table 9-32: Multicast Base Address Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
5:0	**MC_Index_Position** – PF value applies to all VFs.	Base	RsvdP
63:12	**MC_Base_Address** – PF value applies to all VFs.	Base	RsvdP

9.3.7.11 *Page Request Interface (PRI)*

Page Request Interface functionality is defined in Address Translation Services (Chapter 10).

A PF is permitted to support the Page Request Interface. The Page Request Interface of the PF is also used by the associated VFs. The PF is permitted to implement the Page Request Interface capability and the VFs shall not implement it.

Even though the Page Request Interface is shared between PFs and VFs, it sends the requesting Function's ID (PF or VF) in the Requester ID field of the Page Request Message and expects the requesting Function's ID in the Destination Device ID field of the resulting PRG Response Message.

9.3.7.12 *Dynamic Power Allocation (DPA)*

VFs may not implement the Dynamic Power Allocation Capability.

Power allocation for VFs is managed using the PF's DPA Capability, if implemented.

9.3.7.13 *TLP Processing Hint (TPH)*

The TPH Extended Capability may be present in VFs, PFs, both or neither. If any VF associated with a given PF contains the capability, all VFs associated with that PF must also support TPH.

All TPH Extended Capability fields in PFs and VFs operate as defined in Section 7.9.13.

For fields in the TPH Requester Capability Register (offset 04h), a PF and its associated VFs may have different values, but all VFs associated with the same PF must have the same values in all fields.

9.3.7.14 *PASID*

An Endpoint device is permitted to support PASID. The PASID configuration of the single function (Function or PF) representing the device is also used by all VFs in the device. A PF is permitted to implement the PASID capability, but VFs must not implement it.

Even though the PASID configuration is shared between Functions, PFs and VFs, the device sends the requesting Function's ID (Function, PF or VF) in the Requester ID field of the TLP containing PASID.

9.3.7.15 *Readiness Time Reporting Extended Capability*

The Readiness Time Reporting Extended Capability may be present in VFs, PFs, both or neither. If any VF associated with a given PF contains the capability, all VFs associated with that same PF must also support Readiness Time Reporting.

The Reset Time field contains the time required following setting of VF Enable (see Section 9.6.1).

The DL_Up Time field is RsvdP.

9.4 All VFs associated with the same PF shall report the same time values.Error Handling

SR-IOV devices utilize the Error Reporting mechanism defined in Section 6.2. Errors that are defined as non-function-specific are only logged in the PF.

Section 6.2 defines two error reporting paradigms: the baseline Capability and the Advanced Error Reporting Capability. The baseline error reporting capabilities are required of all PCI Express devices and define the minimum error reporting requirements. The Advanced Error Reporting Capability is optional and is defined for more robust error reporting and is implemented with a specific PCI Express Capability structure.

9.4.1 Baseline Error Reporting

All SR-IOV devices must support the baseline error reporting capabilities, with some modifications to account for the goal of reduced cost and complexity of implementation.

These control bits are only meaningful in the PF. VFs shall use the error reporting control bits in the associated PF when making decisions on generating error Messages.

These following fields are RsvdP in VFs:

- Command register (see Section 9.3.4.1.3)
  - SERR# Enable
  - Parity Error Response
- Device Control register (see Section 9.3.5.4)
  - Correctable Reporting Enable

- Non-Fatal Reporting Enable
- Fatal Reporting Enable
- Unsupported Request (UR) Reporting Enable

Each VF shall implement a mechanism to provide error status independent of any other Function. This is necessary to provide SI isolation for errors that are Function-specific.

The following baseline error reporting status bits must be implemented in each VF:

- Status register (see Section 9.3.4.1.4)
  - Master Data Parity Error
  - Signaled Target Abort
  - Received Target Abort
  - Received Master Abort
  - Signaled System Error
  - Detected Parity Error
- Device Status register (see Section 9.3.5.5)
  - Correctable Error Detected
  - Non-Fatal Error Detected
  - Fatal Error Detected
  - Unsupported Request Detected

Each VF shall use its own Routing ID when signaling errors.

9.4.2 Advanced Error Reporting

The Advanced Error Reporting Capability is optional. If AER is not implemented in the PF, it must not be implemented in the associated VFs. If AER is implemented in the PF, it is optional in the VFs.

Section 6.2.4 classifies errors as Function-specific and non-Function-specific. Each VF that implements the Advanced Error Reporting Capability must maintain its own error reporting status for function-specific errors.

9.4.2.1 *VF Header Log*

A Device that implements AER in the VFs may share Header Log Registers among VFs associated with a single PF. See Section 9.4.2.10 for details.

Header logging Registers for the PF are independent of its associated VFs and must be implemented with dedicated storage space.

When implementing a reduced set of Header Log Registers, a Function may not have room to log a header associated with an error. In this case, the Function shall update the Uncorrectable Error Status Register and Advanced Error Capabilities and Control register as required by Section 6.2.4;

however, when the Header Log Register is read, it shall return all 1s to indicate an overflow condition and no header was logged.

9.4.2.2 *Advanced Error Reporting Capability*

Figure 7-107 describes the AER extended capability structure.

9.4.2.3 *Advanced Error Reporting Extended Capability Header (Offset 00h)*

This register contains the PCI Express Extended Capability ID, Capability Version, and Next Capability Offset. These fields are unchanged and are described in Section 7.8.4.1.

9.4.2.4 *Uncorrectable Error Status Register (Offset 04h)*

The Uncorrectable Error Status register indicates error detection status of individual errors. Errors that are defined as non-Function-specific are logged in the PF. Only Function-specific errors are logged in the VFs.

PF and VF functionality is defined in Section 7.8.4.2 except where noted in Table 9-33.

Table 9-33: Uncorrectable Error Status Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
4	Data Link Protocol Error Status	Base	0b
5	Surprise Down Error Status	Base	0b
13	Flow Control Protocol Error Status	Base	0b
17	Receiver Overflow Status	Base	0b
18	Malformed TLP Status	Base	0b
19	ECRC Error Status	Base	0b

9.4.2.5 *Uncorrectable Error Mask Register (Offset 08h)*

PF and VF functionality is defined in Section 7.8.4.3 except where noted in Table 9-34. For VF fields marked RsvdP, the PF setting applies to the VF. For VF fields marked 0b, the error is not applicable to a VF.

Table 9-34: Uncorrectable Error Mask Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
4	Data Link Protocol Error Mask	Base	0b
5	Surprise Down Error Mask	Base	0b
12	Poisoned TLP Mask	Base	RsvdP
13	Flow Control Protocol Error Mask	Base	0b
14	Completion Timeout Mask	Base	RsvdP

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
15	Completer Abort Mask	Base	RsvdP
16	Unexpected Completion Mask	Base	RsvdP
17	Receiver Overflow Mask	Base	0b
18	Malformed TLP Mask	Base	0b
19	ECRC Error Mask	Base	0b
20	Unsupported Request Error Mask	Base	RsvdP
21	ACS Violation Mask	Base	RsvdP

9.4.2.6 *Uncorrectable Error Severity Register (Offset 0Ch)*

PF and VF functionality is defined in Section 7.8.4.4 except where noted in Table 9-35. For VF fields marked RsvdP, the PF setting applies to the VF. For VF fields marked 0b, the error is not applicable to a VF.

Table 9-35: Uncorrectable Error Severity Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
4	**Data Link Protocol Error Severity**	Base	0b
5	**Surprise Down Error Severity**	Base	0b
12	**Poisoned TLP Severity**	Base	RsvdP
13	**Flow Control Protocol Error Severity**	Base	0b
14	**Completion Timeout Error Severity**	Base	RsvdP
15	**Completer Abort Error Severity**	Base	RsvdP
16	**Unexpected Completion Error Severity**	Base	RsvdP
17	**Receiver Overflow Severity**	Base	0b
18	**Malformed TLP Severity**	Base	0b
19	**ECRC Error Severity**	Base	0b
20	**Unsupported Request Error Severity**	Base	RsvdP
21	**ACS Violation Severity**	Base	RsvdP

9.4.2.7 *Correctable Error Status Register (Offset 10h)*

The Correctable Error Status register indicates error detection status of individual correctable errors. Errors that are defined as non-Function-specific are logged in the PF. Only Function-specific errors are logged in the VFs.

PF and VF functionality is defined in Section 7.8.4.5 except where noted in Table 9-36.

Table 9-36: Correctable Error Status Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
0	**Receiver Error Status**	Base	0b
6	**Bad TLP Status**	Base	0b
7	**Bad DLLP Status**	Base	0b
8	**REPLAY_NUM Rollover Status**	Base	0b
12	**Replay Timer Timeout Status**	Base	0b
15	**Header Log Overflow Status** If the VF implements Header Log sharing (see Section 9.4.2.1), this bit is hardwired to 0b.	Base	Base / 0b

9.4.2.8 *Correctable Error Mask Register (Offset 14h)*

PF and VF functionality is defined in Section 7.8.4.6 except where noted in Table 9-37. For VF fields marked RsvdP, the PF setting applies to the VF.

Table 9-37: Correctable Error Mask Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
0	**Receiver Error Mask**	Base	RsvdP
6	**Bad TLP Mask**	Base	RsvdP
7	**Bad DLLP Mask**	Base	RsvdP
8	**REPLAY_NUM Rollover Mask**	Base	RsvdP
12	**Replay Timer Timeout Mask**	Base	RsvdP
13	**Advisory Non-Fatal Error Mask**	Base	RsvdP
15	**Header Log Overflow Mask** If the VF implements Header Log sharing (see Section 9.4.2.1), this bit is RsvdP.	Base	Base / RsvdP

9.4.2.9 *Advanced Error Capabilities and Control Register (Offset 18h)*

PF and VF functionality is defined in Section 7.8.4.7 except where noted in Table 9-38. For VF fields marked RsvdP, the PF setting applies to the VF.

Table 9-38: Advanced Error Capabilities and Control Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
6	**ECRC Generation Enable**	Base	RsvdP

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
8	**ECRC Check Enable**	Base	RsvdP
9	**Multiple Header Recording Capable** If the VF implements Header Log sharing (see Section 9.4.2.1), this bit is hardwired to 0b.	Base	Base / 0b
10	**Multiple Header Recording Enable** If the VF implements Header Log sharing (see Section 9.4.2.1), this bit is RsvdP.	Base	Base / RsvdP
11	**TLP Prefix Log Present** If the VF implements Header Log Sharing (see Section 9.4.2.1), this bit is 0b when the Header Log contains all 1s due to an overflow condition.	Base	Base (see description)

9.4.2.10 *Header Log Register (Offset 1Ch)*

The Header Log register captures the header for the TLP corresponding to a detected error. See also Section 9.4.2.1.

A Device that implements AER in the VFs may share Header Log Registers among VFs associated with a single PF. A shared header log must have storage for at least one header.

Header logging Registers for the PF is independent of its associated VFs and must be implemented with dedicated storage space.

When an error is detected in a VF, the error shall be logged as specified in Section 6.2. If a shared set of Header Log Registers is implemented, a VF may not have room to log a header. In this case, the VF shall update its Uncorrectable Error Status Register and Advanced Error Capabilities and Control register as required in Section 6.2; however, when that VF's Header Log Register is read, it shall return all 1s to indicate an overflow condition.

The VF's header log entry shall be locked and remain valid while that VF's First Error Pointer is valid. As defined in Section 6.2, the First Error Pointer register is valid when the corresponding bit of the Uncorrectable Error Status register is Set. While the header log entry is locked, additional errors shall not overwrite the locked entry for this or any other VF. When a header entry is unlocked, it shall be available to record a new error for any VF sharing the header logs.

PF and VF functionality is defined in Section 7.8.4.8, except where noted in Table 9-39.

Table 9-39: Header Log Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
127:0	Header of TLP associated with error (additional requirements are described above)	Base	Base

9.4.2.11 *Root Error Command Register (Offset 2Ch)*

This register is not applicable to Devices.

9.4.2.12 *Root Error Status Register (Offset 30h)*

This register is not applicable to Devices.

9.4.2.13 *Correctable Error Source Identification Register (Offset 34h)*

This register is not applicable to Devices.

9.4.2.14 *Error Source Identification Register (Offset 36h)*

This register is not applicable to Devices.

9.4.2.15 *TLP Prefix Log Register (Offset 38h)*

For PFs and VFs, if End-End TLP Prefixes are supported, this register is implemented.

For a VF, if a shared set of Header Log registers is implemented (Section 9.4.2.1), this register's contents are undefined when the Header Log contains all 1s due to an overflow condition.

9.5 Interrupts

SR-IOV capable devices utilize the same Interrupt signaling mechanisms defined in Section 6.1.

9.5.1 Interrupt Mechanisms

There are three methods of signaling interrupts:

- INTx
- MSI
- MSI-X

PFs may implement INTx. VFs must not implement INTx. PFs and VFs shall implement MSI or MSI-X or both if interrupt resources are requested. Each PF and VF must implement its own unique interrupt capabilities.

9.5.1.1 *MSI Interrupts*

MSI Capability and PF and VF functionality are defined in Section 7.7 except where noted in Table 9-40.

Table 9-40: MSI Capability: Message Control

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
8	Per-Vector Masking Capable	1b	1b

9.5.1.2 *MSI-X Interrupts*

The MSI-X Capability is defined in Section 7.7 and depicted in Figure 9-19.

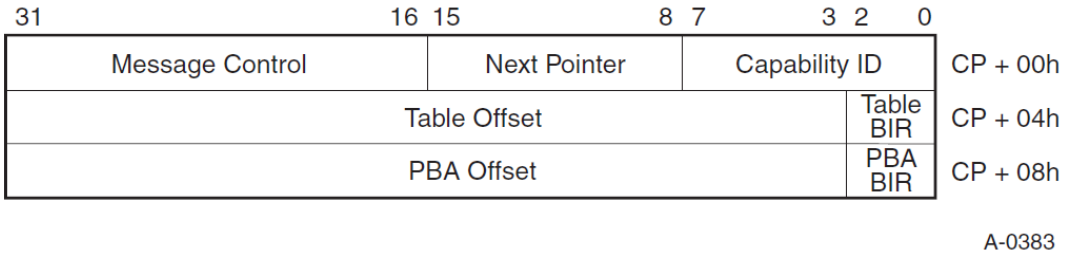


Figure 9-19: MSI-X Capability

PF and VF functionality is identical to that of a Function as defined in Section 7.7.2.

Note that for VFs the Table Offset and PBA Offset values are relative to the VF's Memory address space.

9.5.1.3 *Address Range Isolation*

If a BAR that maps address space for the MSI-X Table or MSI-X PBA also maps other usable address space that is not associated with MSI-X structures, locations (e.g., for CSRs) used in the other address space must not share any naturally aligned System Page Size address range with one where either MSI-X structure resides. The MSI-X Table and MSI-X PBA are permitted to co-reside within a naturally aligned System Page Size address range, though they must not overlap with each other.

9.6 Power Management

This section defines the PCI Express SR-IOV power management capabilities and protocols.

The Power Management Capability is required for PFs as described in Chapter 5.

For VFs, the Power Management Capability is optional.

9.6.1 VF Device Power Management States

If a VF does not implement the Power Management Capability, then the VF behaves as if it had been programmed into the equivalent power state of its associated PF.

If a VF implements the Power Management Capability, the functionality is defined in Section 7.5 except as noted in Section 9.6.4.

If a VF implements the Power Management Capability, the Device behavior is undefined if the PF is placed in a lower power state than the VF. Software should avoid this situation by placing all VFs in lower power state before lowering their associated PF's power state.

A VF in the D0 state is in the $D0_{active}$ state when the VF has completed its internal initialization and either the VF's Bus Master Enable bit is Set (see Section 9.3.4.1.3) or the VF MSE bit in the SR-IOV Control (see Section 9.3.3.3) Extended Capability is Set. The VF's internal initialization must have completed whenany of the following conditions have occurred:

- The VF has responded successfully (without returning CRS) to a Configuration Request.
- After issuing an FLR to the VF, one of the following is true:
  - At least 1.0 s has passed since the FLR was issued.
  - The VF supports Function Readiness Status and, after the FLR was issued, an FRS Message from the VF with Reason Code FLR Completed has been received.
  - At least FLR time has passed since the FLR was issued. FLR Time is either (1) the FLR Time value in the Readiness Time Reporting capability associated with the VF or (2) a value determined by system software / firmware6[144].
- After Setting VF Enable in a PF, at least one of the following is true:
  - At least 1.0 s has passed since VF Enable was Set.
  - The PF supports Function Readiness Status and, after VF Enable was Set, an FRS Message from the PF with Reason Code VF Enabled has been received.
- After transitioning a VF from $D3_{hot}$ to D0, at least one of the following is true:
  - At least 10 ms has passed since the request to enter D0 was issued.
  - The VF supports Function Readiness Status and, after the request to enter D0 was issued, an FRS Message from the VF with Reason Code $D3_{hot}$ to D0 Transition Completed has been received.
  - At least $D3_{hot}$ to D0 Time has passed since the request to enter D0 was issued. $D3_{hot}$ to D0 Time is either (1) the $D3_{hot}$ to D0 Time in the Readiness Time Reporting capability associated with the VF or (2) a value determined by system software / firmware[144].

[144] For example, ACPI tables.

9.6.2 PF Device Power Management States

The PF's power management state (D-state) has global impact on its associated VFs. If a VF does not implement the Power Management Capability, then it behaves as if it is in an equivalent power state of its associated PF.

If a VF implements the Power Management Capability, the Device behavior is undefined if the PF is placed in a lower power state than the VF. Software should avoid this situation by placing all VFs in lower power state before lowering their associated PF's power state.

When the PF is placed into the $D3_{hot}$ state:

- If the No_Soft_Reset bit is Clear then the PF performs an internal reset on the $D3_{hot}$ to D0 transition and all its configuration state returns to the default values.

  Note: Resetting the PF resets VF Enable which means that VFs no longer exist and any VF specific context is lost after the $D3_{hot}$ to D0 transition is complete.

- If the No_Soft_Reset bit is Set then the internal reset does not occur. The SR-IOV extended capability retains state, and associated VFs remain enabled.

When the PF is placed into the $D3_{cold}$ state VFs no longer exist, any VF specific context is lost and PME events can only be initiated by the PF.

IMPLEMENTATION NOTE

No_Soft_Reset Strongly Recommended

It is strongly recommended that the No_Soft_Reset bit be Set in all Functions of a Multi-Function Device. This recommendation applies to PFs.

9.6.3 Link Power Management State

VF Power State does not affect Link Power State.

Link Power State is controlled solely by the setting in the PFs regardless of the VF's D-state.

9.6.4 VF Power Management Capability

The following tables list the requirements for PF and VFs Power Management Capability.

PF and VF functionality is defined in Section 7.5 except where noted in Table 9-41 and Table 9-42.

Table 9-41: Power Management Control/Status (PMCSR)

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
14:13	**Data_Scale**	Base	00b
12:9	**Data_Select**	Base	0000b

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
3	**No_Soft_Reset** –If a VF implements the Power Management Capability, the VF's value of this field must be identical to the associated PF's value.	Base	Base

Table 9-42: Power Management Data Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
7:0	**Data**	Base	00000000b

9.6.5 VF EmergencyPower Reduction State

If the Emergency Power Reduction Supported field in a VF is non-zero, that VF enters and exits the Emergency Power Reduction State at the same time as the associated PF. Software can use the Emergency Power Reduction Detected bit in the PF to emulate the corresponding bit in the VF.

10

10 ATS Specification

10.1 Architectural Overview

Most contemporary system architectures make provisions for translating addresses from DMA (bus mastering) I/O Functions. In many implementations, it has been common practice to assume that the physical address space seen by the CPU and by an I/O Function is equivalent. While in others, this is not the case. The address programmed into an I/O Function is a "handle" that is processed by the Root Complex (RC). The result of this processing is often a translation to a physical memory address within the central complex. Typically, the processing includes access rights checking to insure that the DMA Function is allowed to access the referenced memory location(s).

The purposes for having DMA address translation vary and include:

- Limiting the destructiveness of a "broken" or miss-programmed DMA I/O Function
- Providing for scatter/gather
- Ability to redirect message-signaled interrupts (e.g., MSI or MSI-X) to different address ranges without requiring coordination with the underlying I/O Function
- Address space conversion (32-bit I/O Function to larger system address space)
- Virtualization support

Irrespective of the motivation, the presence of DMA address translation in the host system has certain performance implications for DMA accesses.

Depending on the implementation, DMA access time can be significantly lengthened due to the time required to resolve the actual physical address. If an implementation requires access to a main-memory-resident translation table, the access time can be significantly longer than the time for an untranslated access. Additionally, if each transaction requires multiple memory accesses (e.g., for a table walk), then the memory transaction rate (i.e., overhead) associated with DMA can be high.

To mitigate these impacts, designs often include address translation caches in the entity that performs the address translation. In a CPU, the address translation cache is most commonly referred to as a translation look-aside buffer (TLB). For an I/O TA, the term address translation cache or ATC is used to differentiate it from the translation cache used by the CPU.

While there are some similarities between TLB and ATC, there are important differences. A TLB serves the needs of a CPU that is nominally running one thread at a time. The ATC, however, is generally processing requests from multiple I/O Functions, each of which can be considered a separate thread. This difference makes sizing an ATC difficult depending upon cost models and expected technology reuse across a wide range of system configurations.

The mechanisms described in this specification allow an I/O Device to participate in the translation process and provide an ATC for its own memory accesses. The benefits of having an ATC within a Device include:

- Ability to alleviate TA resource pressure by distributing address translation caching responsibility (reduced probability of "thrashing" within the TA)
- Enable ATC Devices to have less performance dependency on a system's ATC size
- Potential to ensure optimal access latency by sending pretranslated requests to central complex

This specification will provide the interoperability that allows PCIe Devices to be used in conjunction with a TA, but the TA and its Address Translation and Protection Table (ATPT) are treated as implementation-specific and are outside the scope of this specification. While it may be possible to implement ATS within other PCIe Components, this specification is confined to PCIe Devices and PCIe Root Complex Integrated Endpoints (RCiEPs).

Figure 10-1: illustrates an example platform with a TA and ATPT, along with a set of PCIe Devices and RC Integrated Endpoints with integrated ATC. A TA and an ATPT are implementation-specific and can be distinct or integrated components within a given system design.

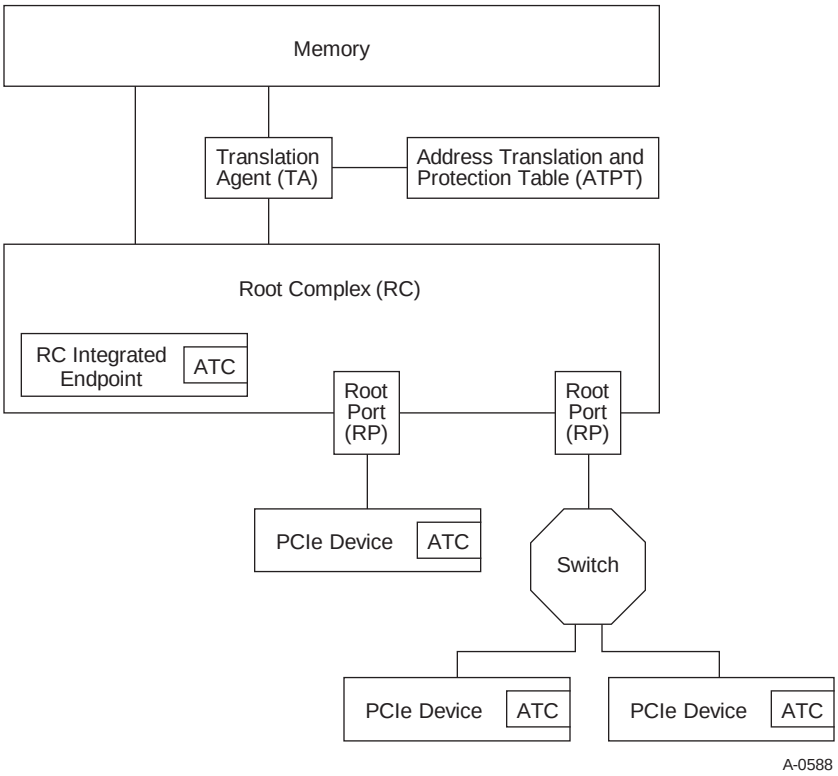


Figure 10-1: Example Illustrating a Platform with TA, ATPT, and ATC Elements

10.1.1 Address Translation Services (ATS) Overview

The ATS chapter provides a new set of TLP and associated semantics. ATS uses a request-completion protocol between a Device[145] and a Root Complex (RC) to provide translation services. In addition, a new AT field is defined within the Memory Read and Memory Write TLP. The new AT field enables an RC to determine whether a given request has been translated or not via the ATS protocol.

Figure 10-2 illustrates the basic flow of an ATS Translation Request operation.

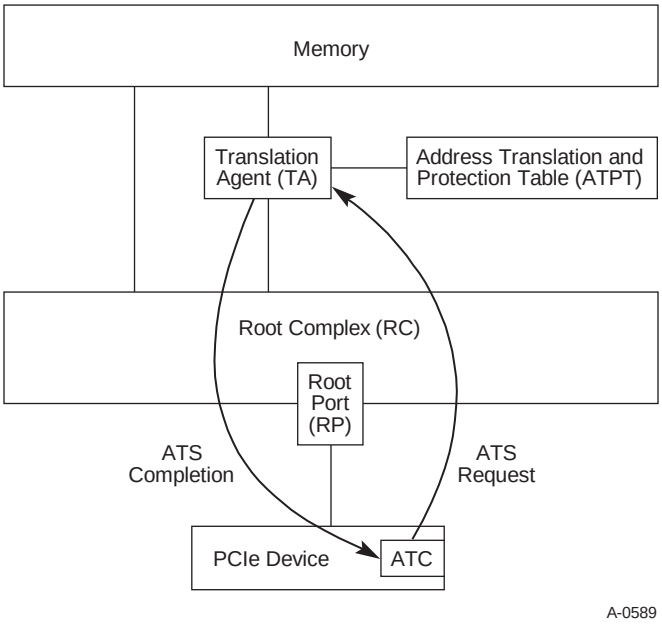


Figure 10-2: Example ATS Translation Request/Completion Exchange

In this example, a Function-specific work request is received by a single-Function PCIe Device. The Function determines through an implementation-specific method that caching a translation within its ATC would be beneficial. There are a number of considerations a Function or software can use in making such a determination; for example:

- Memory address ranges that will be frequently accessed over an extended period of time or whose associated buffer content is subject to a significant update rate
- Memory address ranges, such as work and completion queue structures, data buffers for low-latency communications, graphics frame buffers, host memory that is used to cache Function-specific content, and so forth

Given the variability in designs and access patterns, there is no single criteria that can be applied.

[145] All references within this specification to a Device apply equally to a PCIe Device or a Root Complex Integrated Endpoint. ATS does not delineate between these two types in terms of requirements, semantics, configuration, error handling, etc. From a software perspective, an ATS-capable Root Complex Integrated Endpoint must behave the same as an ATS-capable non-integrated Device.

The Function generates an ATS Translation Request which is sent upstream through the PCIe hierarchy to the RC which then forwards it to the TA. An ATS Translation Request uses the same routing and ordering rules as defined in this specification. Further, multiple ATS Translation Requests can be outstanding at any given time; i.e., one may pipeline multiple requests on one or more TC. Each TC represents a unique ordering domain and defines the domain that must be used by the associated ATS Translation Completion.

Upon receipt of an ATS Translation Request, the TA performs the following basic steps:

1. Validates that the Function has been configured to issue ATS Translation Requests.
2. Determines whether the Function may access the memory indicated by the ATS Translation Request and has the associated access rights.
3. Determines whether a translation can be provided to the Function. If yes, the TA issues a translation to the Function.
   a. ATS is required to support a variety of page sizes to accommodate a range of ATPT and processor implementations.
      i. Page sizes are required to be a power of two and naturally aligned.
      ii. The minimum supported page size is 4096 bytes. ATS capable components are required to support this minimum page size.
   b. A Function must be informed of the minimum translation or invalidate size it will be required to support to provide the Function an opportunity to optimize its resource utilization. The smallest minimum translation size must be 4096 bytes.
4. The TA communicates the success or failure of the request to the RC which generates an ATS Translation Completion and transmits via a Response TLP through a RP to the Function.
   a. An RC is required to generate at least one ATS Translation Completion per ATS Translation Request; i.e., there is minimally a 1:1 correspondence independent of the success or failure of the request.
      i. A successful translation can result in one or two ATS Translation Completion TLPs per request. The Translation Completion indicates the range of translation covered.
      ii. An RC may pipeline multiple ATS Translation Completions; i.e., an RC may return multiple ATS Translation Completions and these ATS Translation Completions may be in any order relative to ATS Translation Requests.
      iii. The RC is required to transmit the ATS Translation Completion using the same TC (Traffic Class) as the corresponding ATS Translation Request.
   b. The requested address may not be valid. The RC is required to issue a Translation Completion indicating that the requested address is not accessible.

When the Function receives the ATS Translation Completion and either updates its ATC to reflect the translation or notes that a translation does not exist. The Function proceeds with processing its work request and generates subsequent requests using either a translated address or an untranslated address based on the results of the Completion.

a. Similar to Read Completions, a Function is required to allocate resource space for each completion(s) without causing backpressure on the PCIe Link.

b. A Function is required to discard Translation Completions that might be "stale". Stale Translation Completions can occur for a variety of reasons.

As one can surmise, ATS Translation Request and Translation Completion processing is conceptually similar and, in many respects, identical to PCIe Read Request and Read Completion processing. This is intentional to reduce design complexity and to simplify integration of ATS into existing and new PCIe-based solutions. Keeping this in mind, ATS requires the following:

- ATS capable components must interoperate with *PCI Express Base Specification, Revision 1.1* compliant components.
- ATS is enabled through a new Capability and associated configuration structure. To enable ATS, software must detect this Capability and enable the Function to issue ATS TLP. If a Function is not enabled, the Function is required not to issue ATS Translation Requests and is required to issue all DMA Read and Write Requests with the TLP AT field set to "untranslated".
- ATS TLPs are routed using either address-based or Requester ID (RID) routing.
- ATS TLPs are required to use the same ordering rules as specified in this specification.
- ATS TLPs are required to flow unmodified through PCIe 1.1-compliant Switches.
- A Function is permitted to intermix translated and untranslated requests.
- ATS transactions are required not to rely upon the address field of a memory request to communicate additional information beyond its current use as defined by the PCI-SIG.

IMPLEMENTATION NOTE

Address Range Overlap

It is likely that the untranslated and translated address range will overlap, perhaps in their entirety. This is not a requirement of ATS but may be an implementation constraint on the TA so that memory requests will be properly routed.

In contrast to the prior example, Figure 10-3 illustrates an example Multi-Function Device. In this example Device, there are three Functions. Key points to note in Figure 10-3 are:

- Each ATC is associated with a single Function. Each ATS-capable Function must be able to source and sink at least one of each ATS Translation Request or Translation Completion type.
- Each ATC is configured and accessed on a per Function basis. A Multi-Function Device is not required to implement ATS on every Function.
- If the ATC implementation shares resources among a set of Functions, then the logical behavior is required to be consistent with fully independent ATC implementations.

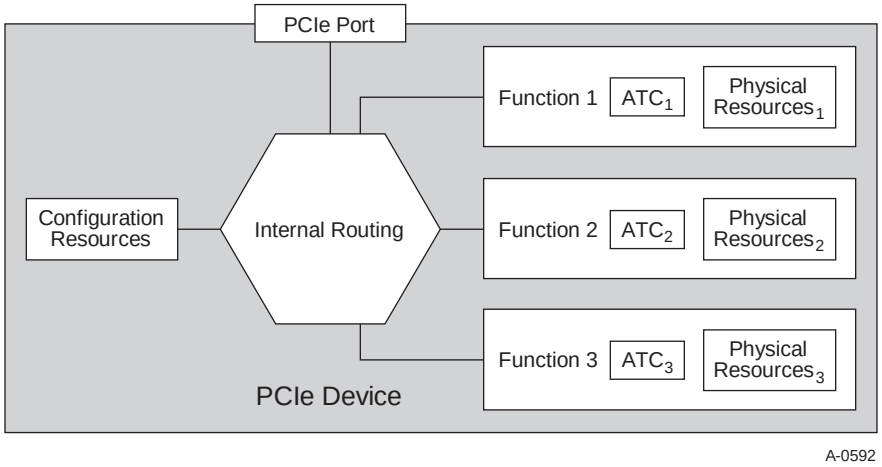


Figure 10-3: Example Multi-Function Device with ATC per Function

Independent of the number of Functions within a Device, the following are required:

- A Function is required not to issue any TLP with the AT field set unless the address within the TLP was obtained through the ATS Translation Request and Translation Completion protocol.
- Each ATC is required to only be populated using the ATS protocol; i.e., each entry within the ATC must be filled via an ATS Translation Completion in response to the Function issuing an ATS Translation Request for a given address.
- Each ATC cannot be modified except through the ATS protocol. That is:
  - Host system software cannot modify the ATC other than through the protocols defined in this specification except to invalidate one or more translations in an ATC. A Device or Function reset would be an example of an operation performed by software to change the contents of the ATC, but a reset is only allowed to invalidate entries not modify their contents.
  - It must not be possible for host system software to use software executing on the Device to modify the ATC.

When a TA determines that a Function should no longer maintain a translation within its ATC, the TA initiates the ATS invalidation protocol. The invalidation protocol consists of a single Invalidation Request and one or more Invalidate Completions.

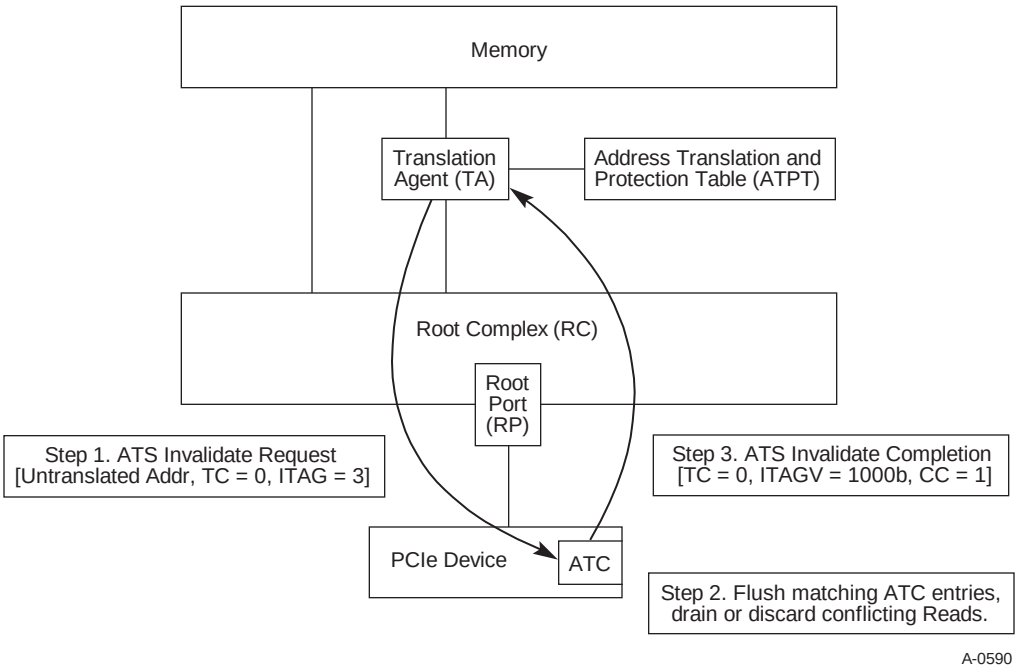


Figure 10-4: Invalidation Protocol with a Single Invalidation Request and Completion

As Figure 10-4 illustrates, there are essentially three steps in the ATS Invalidation protocol:

1. The system software updates an entry in the tables used by the TA. After the table is changed, the TA determines that a translation should be invalidated in an ATC and initiates an Invalidation Request TLP which is transmitted from the RP to the example single-Function Device. The Invalidate Request communicates an untranslated address range, the TC, and an RP unique tag which is used to correlate Invalidate Completions with the Invalidation Request.
2. The Function receives the Invalidate Request and invalidates all matching ATC entries. A Function is not required to immediately flush all pending requests upon receipt of an Invalidate Request. If transactions are in a queue waiting to be sent, it is not necessary for the Function to expunge requests from the queue even if those transactions use an address that is being invalidated.
   a. A Function is required not to indicate the invalidation has completed until all outstanding Read Requests or Translation Requests that reference the associated translated address have been retired or nullified.
   b. A Function is required to ensure that the Invalidate Completion indication to the RC will arrive at the RC after any previously posted writes that use the "stale" address.

3. When a Function has ascertained that all uses of the translated address are complete, it issues one or more ATS Invalidate Completions.
   a. An Invalidate Completion is issued for each TC that may have referenced the range invalidated. These completions act as a flush mechanism to ensure the hierarchy is cleansed of any in-flight transactions which may contain references to the translated address.
      i. The number of Completions required is communicated within each Invalidate Completion. A TA or RC implementation can maintain a counter to ensure that all Invalidate Completions are received before considering the translation to no longer be in use.
      ii. If more than one Invalidation Complete is sent, the Invalidate Completion sent in each TC must be identical in the fields detailed in Section 10.3.2.
   b. An Invalidate Completion contains the ITAG from Invalidate Request to enable the RC to correlate Invalidate Requests and Completions.

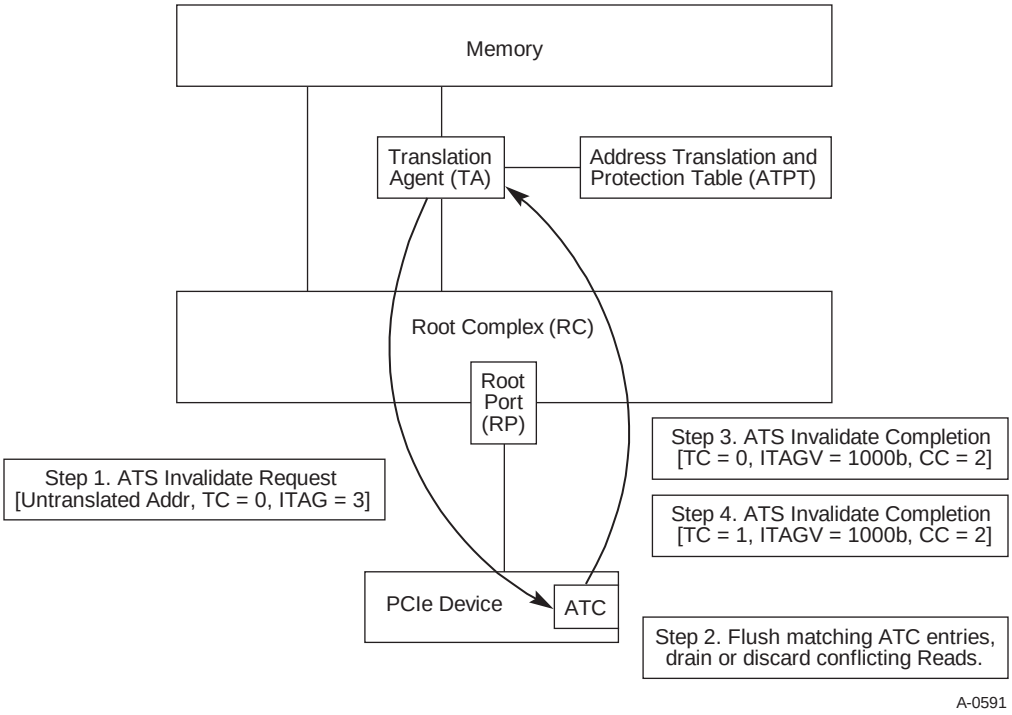


Figure 10-5: Single Invalidate Request with Multiple Invalidate Completions

10.1.2 Page Request Interface Extension

ATS improves the behavior of DMA based data movement. An associated Page Request Interface (PRI) provides additional advantages by allowing DMA operations to be initiated without requiring that all the data to be moved into or out of system memory be pinned.[146] The overhead associated with pinning memory may be modest, but the negative impact on system performance of removing large portions of memory from the pageable pool can be significant.

[146] Locked in place so that it cannot be swapped out by the system's dynamic paging mechanism.

PRI is functionally independent of the other aspects of ATS. That is, a device that supports ATS need not support PRI, but PRI is dependent on ATS's capabilities.

Intelligent I/O devices can be constructed to make good use of a more dynamic memory interface. Pinning will always have the best performance characteristics from a device's perspective–all the memory it wants to touch is guaranteed to be present. However, guaranteeing the residence of all the memory a device might touch can be problematic and force a sub-optimal level of device awareness on a host. Allowing a device to operate more independently (to page fault when it requires memory resources that are not present) provides a superior level of coupling between device and host.[147]

The mechanisms used to take advantage of a Page Request Interface are very device specific. As an example of a model in which such an interface could improve overall system performance, let us examine a high-speed LAN device. Such a device knows its burst rate and need only have as much physical buffer space available for inbound data as it can receive within some quantum. A vector of unpinned virtual memory pages could be made available to the device, that the device then requests as needed to maintain its burst window. This minimizes the required memory footprint of the device and simplifies the interface with the host, both without negatively impacting performance.

The ability to page, begs the question of page table status flag management. Typical TAs associate flags (e.g., dirty and access indications) with each untranslated address. Without any additional hints about how to manage pages mapped to a Function, such TAs would need to conservatively assume that when they grant a Function permission to read or write a page, that Function will use the permission. Such writable pages would need to be marked as dirty before their translated addresses are made available to a Function.

This conservative dirty-on-write-permission-grant behavior is generally not a significant issue for Functions that do not support paging, where pages are pinned and the cost of saving a clean page to memory will seldom be paid. However, Functions that support the Page Request Interface could pay a significant penalty if all writable pages are treated as dirty, since such Functions operate without pinning their accessible memory footprints and may issue speculative page requests for performance. The cost of saving clean pages (instead of just discarding them) in such systems can diminish the value of otherwise attractive paging techniques. This can cause significant performance issues and risk functional issues in circumstances where the backing store is unable to be written, such as a CD-ROM.

The No Write (NW) flag in Translation Requests indicates that a Function is willing to restrict its usage to only reading the page, independent of the access rights that would otherwise have been granted.

If a device chooses to request only read access by issuing a Translation Request with the NW flag Set and later determines that it needs to write to the page, then the device must issue a new Translation Request.

Upon receiving a Translation Request with the NW flag Clear, TAs are permitted to mark the associated pages dirty. It is strongly recommended that Functions not issue such Requests unless they have been given explicit write permission. An example of write permission is where the host issues a command to a Function to load data from a storage device and write that data into memory.

---

[147] The alternative is a private interface between a device and its driver that is used to communicate device state so that the driver can ensure the availability of pinned memory resources.

10.1.3 Process Address Space ID (PASID)

Certain TLPs can optionally be associated with a Process Address Space ID (PASID). This value is conveyed using the PASID TLP Prefix. The PASID TLP Prefix is defined in the *PCI Express Base Specification*.

The PASID TLP Prefix is permitted on:

- Memory Requests (including Untranslated AtomicOp Requests) with Untranslated Addresses
- Address Translation Requests
- Page Request Messages
- ATS Invalidation Requests
- PRG Response Messages

Usage of the PASID TLP Prefix for Untranslated Memory Requests is defined in the *PCI Express Base Specification*. This specification describes PASID TLP Prefix for the remaining TLPs.

When a Request does not have a PASID TLP Prefix, the Untranslated Address represents an address space associated with the Requester ID.

When a Request has a PASID TLP Prefix, the Untranslated Address represents an address space associated with both the Requester ID and the PASID value.

When a Response has a PASID TLP Prefix, the PASID value reflects the address space associated the corresponding Request.

Each Function has an independent set of PASID values. The PASID field is 20 bits wide however the effective width is constrained by the lesser of the width supported by the Root Complex (TA) and the width supported by the Function (ATC). Unused upper bits of the PASID value must be 0b.

For Endpoints in systems where a Virtual Intermediary (VI) is present, Untranslated

Addresses with an associated PASID are typically used to represent Guest Virtual Addresses (GVA) and Untranslated Addresses that are not associated with a PASID represent Guest Physical Addresses (GPA). The TA could be designed so that the VI manages the tables used to perform translations from GPA to Translated Addresses while the individual Guest Operating Systems manage tables used to perform translations from GVA to GPA. When translating an address with an associated PASID, the TA performs both translations and returns the resulting Translated Address (i.e., GVA to GPA followed by GPA to Translated Address). The intermediate GPA value is not visible to the ATC.

When an ATC invalidates a cached GPA mapping, it invalidates the GPA mapping and also invalidates all GVA mappings in the ATC. When the GPA invalidate completes, the VI can safely remove pages backing GPA memory range from a Guest Operating System. The VI does not need to know which GVA mappings involved the GPA mapping.

10.2 ATS Translation Services

A TA does translations. An ATC can cache those translations. If an ATC is separated from the TA by PCIe, the memory request from an ATC will need to be able to indicate if the address in the transaction is translated or not. The modifications to the memory transactions are described in this section, as are the transactions that are used to communicate translations between a remote ATC and a central TA.

10.2.1 Memory Requests with Address Type

A Function with an ATC can send memory read/write Requests that contain either translated or untranslated addresses. As shown in Figure 10-6 and Figure 10-7, the Address Type (AT) field is used to indicate the type of address that is present in the request header.

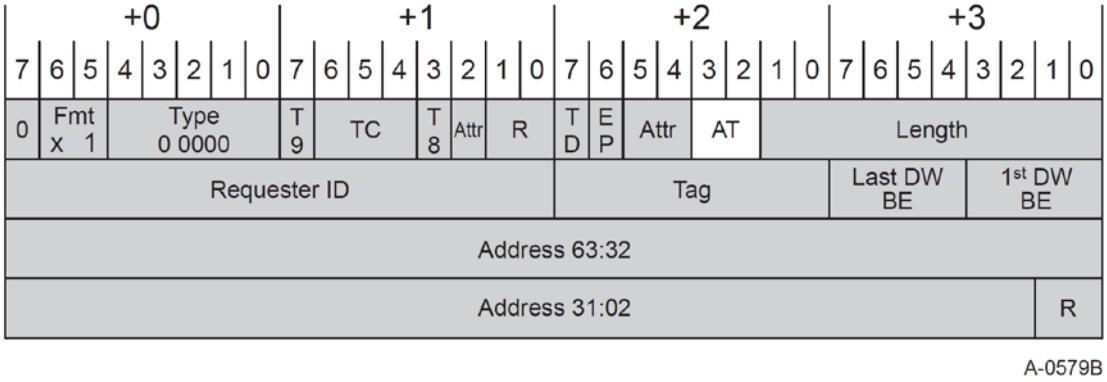


Figure 10-6: Memory Request Header with 64-bit Address

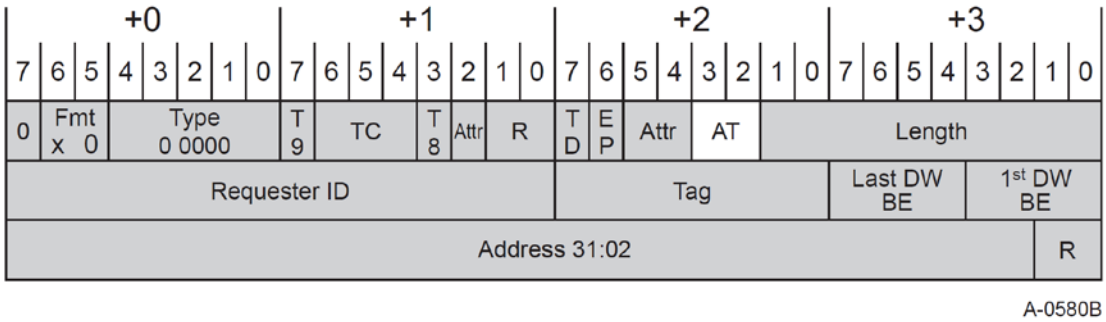


Figure 10-7: Memory Request Header with 32-bit Address

The AT field in the requests is a redefinition of a reserved field in the *PCI Express Base Specification*. Functions that do not implement an ATC will continue to set the AT field to its defined reserved value (00b). Functions that implement an ATC will set the AT field as listed in Table 10-1.

Table 10-1: Address Type (AT) Field Encodings

AT[1:0] Coding	Mnemonic	Meaning
00b	Untranslated	A TA may treat the address as either virtual or physical.
01b	Translation Request	The TA will return the translation of the address contained in the address field of the request as a read completion. This value only has meaning for an explicit Translation Request (see Section 10.2.2). The TA will signal an Unsupported Request (UR) if it receives a TLP with the AT field set to 01b in a Memory Request other than Memory Read.
10b	Translated	The address in the transaction has been translated by an ATC. If the Function associated with the SourceID is allowed to present physical addresses to the system memory, then the TA might not translate this address. If the Function is not allowed to present physical addresses, then the TA may treat this as an UR.
11b	Reserved	The TA will signal an Unsupported Request (UR) if it receives a Memory Request TLP with the AT field set to 11b.

The AT field is only defined for Memory Requests. The field remains reserved for other TLPs.

10.2.2 Translation Requests

A Translation Request has a format that is similar to that of a memory read. The AT field is used to differentiate a Translation Request from a normal memory read.

The request header for a Translation Request has the formats illustrated in Figure 10-8 and Figure 10-9.

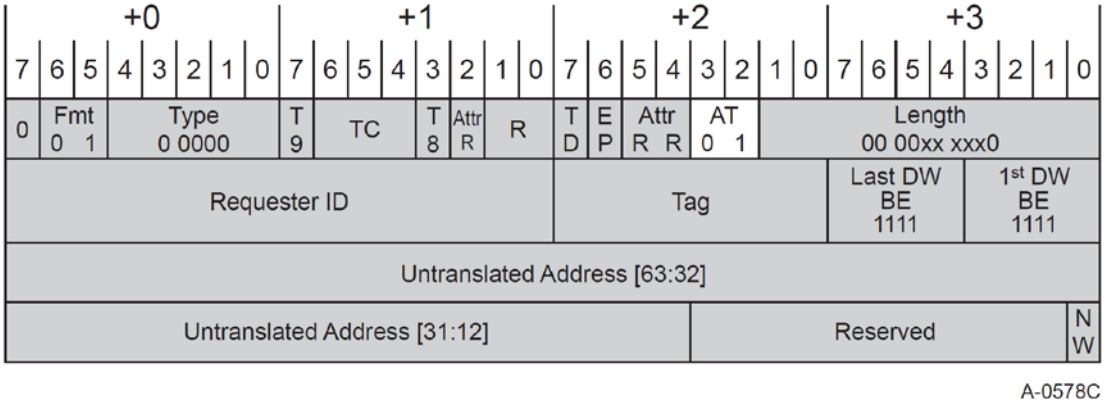


Figure 10-8: 64-bit Translation Request Header

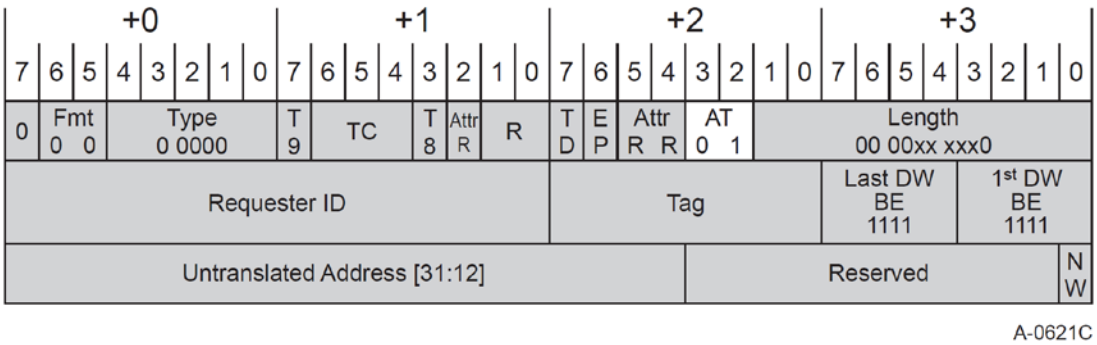


Figure 10-9: 32-bit Translation Request Header

Translation Requests have the same completion timeout intervals as Read Requests.

10.2.2.1 *Attribute Field*

For a Translation Request, the Attr field is reserved for future use. There are no ordering requirements for a Translation Request. A TA may reorder a Translation Request with respect to any other request.

IMPLEMENTATION NOTE

Translation Request Ordering

Because no ordering can be assumed between Translation Requests and other types of Requests, a Translation Request does not make an effective flushing/ordering primitive.

10.2.2.2 *Length Field*

The Length field is set to indicate how many translations may be returned in response to this request. Each translation is 8 bytes in length and represents one or more STUs (Smallest Translation Unit). The maximum setting for the Length field is the RCB. The Length field in a Translation Request must always indicate an even number of DWORDs. If Length is set to indicate a value greater than allowed, or if the least-significant bit of the Length field is non-zero, then the TA will treat the request as a Malformed Packet.

If the Length field has a value greater than two, then the Function is requesting translations for a range of memory greater than a single STU. The additional translations, if provided, are assumed to be for sequentially-increasing, equal-sized, STU-aligned regions, starting at the requested address.

10.2.2.3 *Tag Field*

The Tag field has the same meaning as in a Memory Read Request.

10.2.2.4 *Untranslated Address Field*

A Translation Request includes either a 32-bit or a 64-bit Untranslated Address field. This field indicates the address to be translated. The TA will make decisions about the validity of the request, based on the address in the translation request. The TA is permitted to return fewer translations than requested, but it will not return more.

When multiple translations are requested, the TA will not return a translation if the range of that translation does not overlap the implied range of the Translation Request (this would only apply to translations after the initial value). The implied range of the Translation Request is $[2^{STU+12} * (Length/2)]$ bytes.

The Untranslated Address field in the Translation Request is any address in the range of the first STU. Address bits 11:0 are not present in the Translation Request and are implied to be zero. If a Requester has Page Aligned Request Set (see Section 7.8.8.2), it must ensure that bits 11:2 are zero. If a Requester has Page Aligned Request Clear, it is permitted to supply any value for bits 11:2.[148] The TA must ignore bits 11:2 as well as any low-order bits not required to determine the translation.

For example, if using 64-bit addressing for a Function with the Page Aligned Request bit Set that is programmed with an STU of 1 (i.e., 8192-byte pages), bits 63:13 are significant, bit 12 is ignored by the TA and bits 11:0 are implied to be zero.

10.2.2.5 *No Write (NW) Flag*

The No Write flag, when Set, indicates that the Function is requesting read-only access for this translation.[149]

The TA may ignore the No Write Flag, however, if the TA responds with a translation marked as read-only then the Function must not issue Memory Write transactions using that translation. In this case, the Function may issue another translation request with the No Write flag Clear, which may result in a new translation completion with or without the W (Write) bit Set.

Upon receiving a Translation Request with the NW flag Clear, TAs are permitted to mark the associated pages dirty. It is strongly recommended that Functions not issue such Requests unless they have been given explicit write permission.

10.2.2.6 *PASID TLP Prefix*

If a Translation Request has a PASID TLP Prefix, the Untranslated Address Field is an address within the process address space indicated by the PASID field.

If a Translation Request has a PASID TLP Prefix with either the Privileged Mode Requested or Execute Requested bit Set, these may be used in constructing the Translation Completion Data Entry.

The PASID Extended Capability indicates whether a Function supports and is enabled to send and receive TLPs with the PASID TLP Prefix.

---

[148] Note: The Page Aligned Request bit was added in Revision 1.1 of the ATS Specification.

[149] Note: The No Write Flag was added in Revision 1.1 of the ATS Specification.

10.2.3 Translation Completion

A Translation Completion (either a Cpl or a CplD) is sent by a TA for each Translation Request. This specification describes the meaning of fields in Translation Completions. Fields not defined in this specification have the same meanings proscribed for Read Completions in this specification. The Attr field is reserved for future use.

If the TA was not able to perform the requested translation, a completion with the format shown in Figure 10-10 is used.

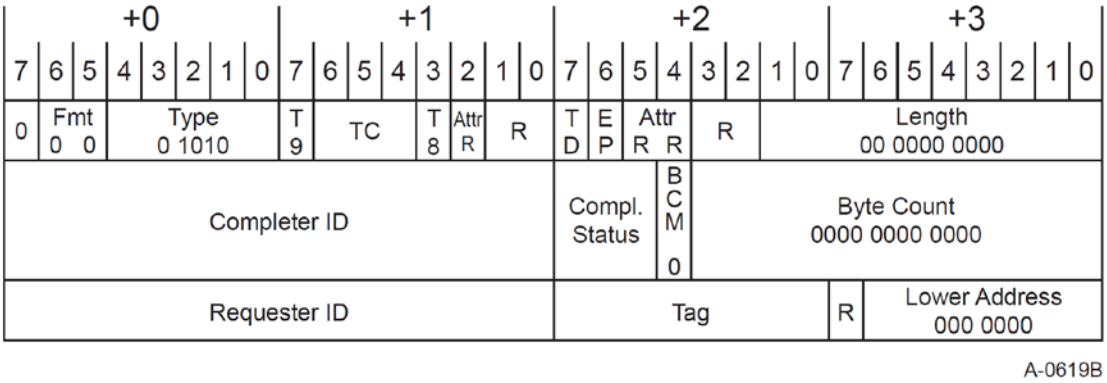


Figure 10-10: Translation Completion with No Data

IMPLEMENTATION NOTE

Byte Count Field for Unsuccessful Translation Completions with No Data

Previous versions of this specification indicated the Byte Count and Lower Address field should be 0000 0000 0000b for Unsuccessful Translation Completions with No Data. It is strongly recommended that implementations do not depend on the Byte Count and Lower Address field being set to any particular value in Unsuccessful Translation Completions with No Data.

The values and meaning for the Completion Status field are listed in Table 2-2.

Table 10-2: Translation Completion with No Data Status Codes

Value	Status	Meaning
000b	Success	This Completion Status has a nominal meaning of “success”. The TA will not return this value in a Cpl.

Value	Status	Meaning
001b	Unsupported Request (UR)	Translation Requests from this Function are not supported by the TA. If a Function receives this Completion code, it must disable its ATC and not send requests using translated addresses until the ATC is re-enabled. For transactions the Function may internally have in flight, the Function may either terminate or complete them. The mechanism a Function receiving this code uses to report this condition is outside the scope of this specification. The TA detecting this error is a "Completer Sending a Completion with UR/CA Status" and shall behave as defined in this specification.
010b	CRS	This value is not allowed in any Completion to a request initiated by a PCI Express Function. If received by a Function, it shall be treated as a Malformed TLP.
100b	Completer Abort (CA)	The TA was not able to translate the address because of an error in the TA. This nominally causes an error to be reported to the device driver associated with the ATC. See AER in this specification.
All others	Reserved	A Translation Completion with a Reserved Completion Status value is treated as if the Completion Status was Unsupported Request (001b).

Note: Return values other than *Success* indicate an error.

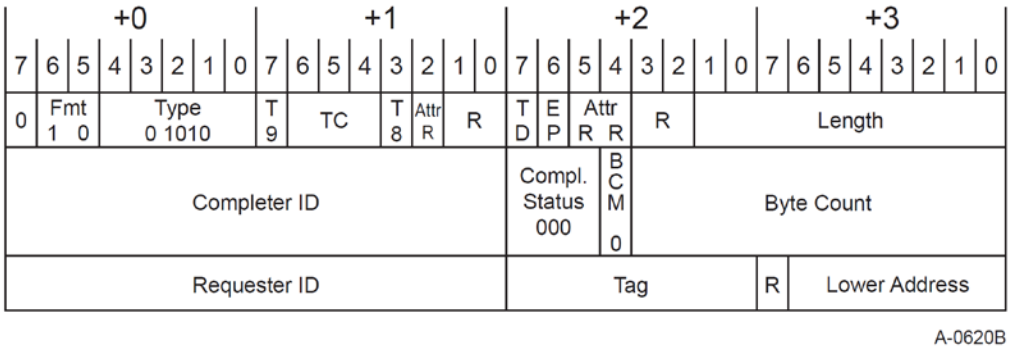


Figure 10-11: Successful Translation Completion

Fields are set in accordance with Sections 2.2.9 and 2.3.1.

Translation Completions must be sent using the same TC as the Translation Request. The Function is not required to verify that the same TC was used.

The Lower Address field will contain a value that will make the packet consistent with RCB semantics. If the result is returned in a single packet, Lower Address is set to RCB minus Byte Count. If the results are returned in multiple packets, the first packet will have a Lower Address field of RCB minus (Total Completion Length * 4) and subsequent packets will have a Lower Address field of 000 0000b. See Section 10.2.4 for additional requirements for multiple packet completions.

If the Completion Status field is 000b, then the translation was successful and a data payload will follow the header. The contents of the data payload are shown in Figure 10-12.

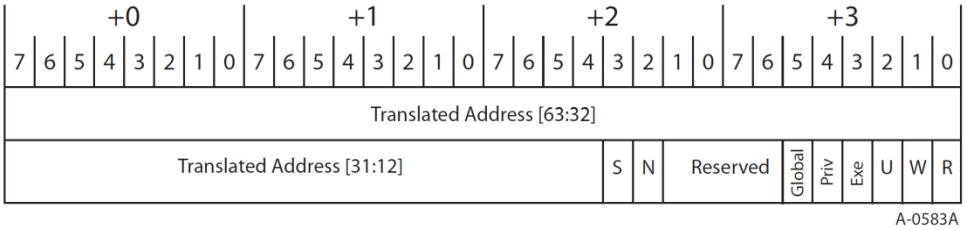


Figure 10-12: Translation Completion Data Entry

Table 10-3: Translation Completion Data Fields

Field	Meaning
S	**Size of translation** – This field is 0b if the translation applies to a 4096-byte range of memory. If this field is 1b, then the translation applies to a range of memory that is larger than 4096 bytes (see Section 10.2.3.1).
N	**Non-snooped accesses** – If this field is 1b, then the read and write requests that use this translation must Clear the No Snoop bit in the Attribute field. If it is 0b, then the Function may use other means to determine if No Snoop should be Set.
Reserved	These bits shall be ignored by the ATC.
Global	**Global Mapping –** If this bit is Set, the ATC is permitted to cache this mapping entry in all PASIDs. If Clear, the ATC is permitted to cache this mapping entry only in the PASID associated with the requesting PASID. This bit may only be Set if the associated Translation Request had a PASID TLP Prefix.
Exe	**Execute Permitted –** If this bit is Set, the requesting Function is permitted to execute code contained in the associated memory range. This bit may be Set only if the associated Translation Request had a PASID TLP Prefix with the Execute Requested bit Set. If this bit is Set, R must also be Set. The Priv bit indicates the Privilege level associated with the Exe bit. If Priv is Set, the Exe bit indicates permissions associated with Privileged Mode entities in the Function. If Priv is Clear, the Exe bit indicates permissions associated with Non- Privileged Mode entities in the Function. This value may be cached if R is Set.
Priv	**Privileged Mode Access –** If this bit is Set, R, W and Exe refer to permissions associated with Privileged Mode entities. If this bit is Clear, R, W and Exe refer to permissions associated with Non-Privileged Mode entities. This bit may only be Set if the associated Translation Request contained a PASID TLP Prefix with the Privileged Mode Requested bit Set. This value must be cached any of the R, W or Exe values are cached.
U	**Untranslated access only** – When this field is Set, the indicated range may only be accessed using untranslated addresses, and the Translated Address field of this Translation Completion Data Entry may not be used in a subsequent Read/Write Request with AT set to Translated. This value may be cached if R or W is Set.

Field	Meaning
R,W	**Read, Write** – These two fields indicate the transaction types that are allowed for requests using the translation. The encodings are: 00bNeither read nor write transactions are allowed. This translation is considered not to be valid. The contents of the Translated Address, N, U, and Exe fields are undefined. A translation with this value may not be cached in the ATC. 01bWrite Requests that target this range are allowed, but Read Requests are not unless they are zero-length reads. 10bRead Requests that target this range are allowed (including zero-length reads), but Write Requests are not. 11bRead and Write Requests that target this range are allowed. The Priv bit indicates the Privilege level associated with R and W. If Priv is Set, R and W indicate permissions associated with Privileged Mode entities in the Function. If Priv is Clear, R and W indicate permissions associated with Non- Privileged Mode entities in the Function.

10.2.3.1 *Translated Address Field*

If the R and W fields are both Clear, or if U is Set, then the Translated Address field may not be used by the Function for any purpose.

If either the R or W field is Set, and the U field is Clear, then the Translated Address field contains an address that can be used by the Function in a Memory Request with the AT field set to Translated and the Function may cache the Translated Address. When cached, the R and W fields must be stored with the same value as the Translation Completion entry. The address that is cached must be a subset of the address range indicated in the Translation Completion (the subset may include the entire range).

While the Translated Address is cached in the Function's ATC, it shall not be possible for the Function to modify the entry other than to delete it. The entry must be deleted from the ATC when an Invalidation Request is received that has an indicated range that overlaps any portion of the cached address.

A Function is not allowed to make an entry into its ATC unless the entry is in a Translation Completion and the E (Enable) field within the ATS Capability is Set. Entries in an ATC cache that are written before the E field is Set must not be used in Memory Request. They must either be invalidated when the E field is Set or ignored and not used.

10.2.3.2 *Translation Range Size (S) Field*

If S is Set, then the translation applies to a range that is larger than 4096 bytes. If S = 1b, then bit 12 of the Translated Address is used to indicate whether or not the range is larger than 8192 bytes. If bit 12 is 0b, then the range size is 8192 bytes, but it is larger than 8192 bytes if Set. If S = 1b and bit 12 = 1b, then bit 13 is used to determine if the range is larger than 16384 bytes or not. If bit 13 is 0b, then the range size is 16384 bytes, but it is larger than 16384 bytes if Set.

Low-order address bits are consumed in sequence to indicate the size of the range associated with the translation.

Note: This encoding method is also used to indicate the size of the memory range being invalidated.

Examples for different translation sizes are shown in Table 10-4.

Table 10-4: Examples of Translation Size Using S Field

Address Bits																						Translation Range Size in Bytes
63:32*	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	S	
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	4 K
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	8 K
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	1	16 K
x	x	x	x	x	x	x	x	x	x	x	x	0	1	1	1	1	1	1	1	1	1	2 M
x	x	x	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1 G
x	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	4 G

*Note: Upper address bits are used to indicate the size for ranges larger than 4 GB.

The size field is set to indicate the range size in multiples of 4096 bytes regardless of the setting of STU. For example, if STU is set to indicate that the minimum translation is 8192 bytes, then S should be Set on all translation returned in a Translation Completion and in all Invalidate Requests. If STU is set to indicate a 16384-byte minimum, then S and bit 12 would both be Set in all translation and invalidate ranges.

If S is Set and bits 63:12 are all 1b, then the behavior is undefined. If S is Set and bit 63 is 0b, and bits 62:12 are all 1b, then the request is to invalidate all translations.

If a Function receives a Translation Completion with a Translation Size field smaller than the Function's programmed STU value, it shall treat the Translation Completion as if it had Completion Status UR.

10.2.3.3 *Non-snooped (N) Field*

This field is Set to indicate that Read and Write Requests that target memory in the range of this translation must Clear the No Snoop Attribute bit in the Request header. When this field is 0b, the Function is allowed to Set the No Snoop Attribute bit in a Function-specific manner.

Note: When this field is Cleared, the Function is not allowed to Set No Snoop in a Memory Request if the Enable No Snoop field in the Device Control register is Cleared.

The N bit may be cached by the ATC if either R or W is Set.

When U is Set, the meaning of this field is undefined, and the TA may set this field to any value. A translation has a single value for the N field that is not affected by privilege level. An ATC is permitted to cache the N field without regard to the value of the Priv bit.

10.2.3.4 *Untranslated Access Only (U) Field*

This field is Set when the Function is not allowed to access the implied range of memory using a translated address (the range is implied by the untranslated address in the Translation Request and the offset of the translation in the Translation Completion). The Function may use untranslated addresses to access the range as long as the accesses are allowed by the R and W fields. The Function may cache this translation value if either R or W is Set. If the U field is Set, the Translated Address field in the translation is not necessarily a valid memory address and the Function may not use the value in a Read or Write Request with AT set to Translated.

> Note: One of the possible uses of this field is to avoid unnecessary invalidations. If a Function uses translated requests for some portions of memory, but not others, then the U field can be used on the portions for which translated requests are not used. When a translation changes if the U field is Set, then it will not necessarily be required that an Invalidate Request be sent to the Function. An example of this use is a Function with a ring buffer that is used for commands. The ring buffer may be allocated for a long period of time and have very high re-use (locality). For this reason, it is useful for the Function to use translated addresses in its memory request that target the command buffer. The same Function might access data buffers that have poor locality and low reuse. Accesses to the data buffers might best be handled by using untranslated Requests. Setting the U field for the data buffer translations ensures that the Function will not attempt to use a translated value to access the data buffer so, when the data buffer mappings are changed, no Invalidation Request is required.
> When U is Set and either R or W is Set, the ATC is permitted to cache U, R, W Exe, and Priv, as well as the Translation Range Size (see Section 10.2.3.2). An Invalidation Request is required if these values change.

10.2.3.5 *Read (R) and Write (W) Fields*

These fields indicate if the returned translation value may be used in a read or write memory request. The ATC may not issue a non-zero read request using the translation value if the R field is Cleared. The ATC may not issue a write request using the translation value if the W field is Cleared. The ATC may not issue any type of request using the translation value if neither the R nor W fields are Set. If both R and W fields are Cleared, the range of the translation is still indicated, but the meaning of the other values in the translation is undefined.

> Note: The range of a Translation entry is indicated even if R = W = 0b in order to allow a "hole" in the Translation Completion. For example, if the Translation Request has a Length of six DWs, then up to three translations could be included in the Translation Completion. The first and third translations may have Set R or W but the second could have R = W = 0b. To avoid ambiguity about the size of the indicated gap, the range of the gap is indicated in the Translation Completion even if R = W = 0b.

The R = 0b, W = 0b state is used to indicate that the address field in the translation may not be used to form a translated address value for a subsequent request.

When the host changes the translation in the TA, to make the translation present, the host is not required to send an invalidation indication to the ATC so that it will know of the change in state of the translation. Since the ATC may not be notified of changes of the translation, a translation value of R = W = 0b may not be cached.

If no table entry is found for the requested address, the TA will return a CplD with a single translation value with R = W = 0b.

Note: Implementations should not assume that receiving a translation response with the R or W bits Set *(independent of the value of the U bit) implies that a subsequent read or write request with the same* ***untranslated*** *address will succeed. Although it may be possible for a device and its controlling software to ensure this property, the method for doing so is outside the scope of this specification.*
The Priv bit is used to qualify R and W. If Priv is Set, R and W indicate permissions granted to Privileged Mode entities in the Function. If Priv is Clear, R and W indicate permissions granted to Non-Privileged Mode entities in the Function. The R and W values for the two privilege levels are independent. The ATC must not assume any correlation between the Privileged Mode and Non-Privileged Mode permissions associated with a translation.

10.2.3.6 *Execute Permitted (Exe)*

If Exe is Set, the requesting Function is permitted to execute code in the implied range of memory. If Exe is Clear, the requesting Function is not permitted to execute code in the implied range of memory.

The definition of what it means for a Function to execute code is outside the scope of this specification. Various system components may have different instruction sets. Behavior within the requesting Function when it attempts to execute code that is not permitted by this bit is outside the scope of this specification.

The Exe bit may only be Set the TA supports Execute permissions, the associated Translation Request had a PASID TLP Prefix with an effective value of 1b for the Execute Requested bit[150] and R is Set in the Translation Completion Data Entry. Otherwise, the Exe bit must be Clear.

This value may be cached if R is Set.

The Priv bit is used to qualify the Exe bit. If Priv is Set, the Exe bit indicates permission granted to Privileged Mode entities in the Function. If Priv is Clear, the Exe bit indicates permission granted to Non-Privileged Mode entities in the Function. The Exe bit values for the two privilege levels are independent. The ATC must not assume any correlation between the Privileged Mode and Non-Privileged Mode permissions associated with a translation.

Functions may optionally check that:

- If the Execute Requested bit is Clear in a Translation Request, the Exe bits in the associated Translation Completion Data Entries are also Clear.
- If Exe is Set, R is also Set.

If either optional check fails, the Function shall signal Unexpected Completion (UC). These checks are independently optional.

[150] The effective value of the bit is 0b unless the bit in the PASID TLP Prefix is 1b and usage of the bit is enabled for the request (see the *PCI Express Base Specification*).

10.2.3.7 *Privileged Mode Access (Priv)*

If Priv is Set, R, W, and Exe refer to permissions granted to entities operating in Privileged Mode in the requesting Function. If Priv is Clear, R, W, and Exe refer to permissions granted to entities operating in Non-Privileged Mode in the requesting Function.

The meaning of Privileged Mode and Non-Privileged Mode and what it means for an entity to be operating as in Privileged Mode or in Non-Privileged Mode depends on the protection model of the system and is outside the scope of this specification.

Behavior is outside the scope of this specification when an entity in the requesting Function attempts to access memory that it is not permitted to access.

The Priv bit may only be Set if the TA supports Privileged Mode and the associated Translation Request had a PASID TLP Prefix with an effective value of 1b for the Privileged Mode Requested[151] bit. Otherwise, the Priv bit must be Clear.

The Privileged and Non-Privileged Mode versions of R, W and Exe are independent. An ATC may cache either or both versions of R, W and Exe. An ATC that receives a translationwith R=W=0b for one privilege level may not assume anything about what it might receive for the other privilege level.

This value may be cached if R or W is Set. This value must be cached when the corresponding R, W, or Exe values are cached.

Note: Since the Priv bit is Set only when the requesting Function Sets the Privileged Mode Requested bit, Functions that never set that bit should always receive the Priv bit Clear and thus don't need to cache it.

Functions may optionally check that when the Privileged Mode Requested bit is Clear in a Translation Request, the Priv bits in the associated Translation Completion Data Entries are also Clear. If this optional check fails, the Function shall signal Unexpected Completion (UC).

IMPLEMENTATION NOTE

Execute Permission and Privilege Mode Enforcement

The requesting Function determines whether a particular Memory Request needs Execute permission or is associated with a Privileged Mode or Non-Privileged Mode entity. The ATC implements the protection checks indicated by the Exe and Priv bits.

10.2.3.8 *Global Mapping (Global)*

If Global is Set, the requesting Function is permitted to create a Global Mapping entry in the ATC for this translation. If Global is Clear, the requesting Function is not permitted to create a Global Mapping entry in the ATC for this translation. Global Mapping entries apply to all PASIDs of the

[151] The effective value of the bit is 0b unless the bit in the prefix is 1b and usage of the bit is enabled for the request (see the *PCI Express Base Specification*).

Function. They permit the ATC to reduce the number of translation requests needed and to reduce the memory needed for caching the results.

A Function is permitted to ignore this bit and always create non-Global Mapping entries in the ATC. This could result in multiple translations being requested for the same Untranslated Address under different PASIDs.

Functions that use this bit must also have the Global Invalidate Supported bit Set (see Section 10.5.1.2).

10.2.4 Completions with Multiple Translations

An ATC is allowed to request that the TA provide translations for a virtually contiguous range of addresses. It does this by setting the Length field in the Translation Request to a value that is two times the number of requested translations as long as the request size (Total Completion Length * 4) is not larger than either Max_Read_Request_Size or RCB.

If multiple translations are requested, the TA may return one or more translations as long as the number of translations does not exceed the number of requested translations. It is not an error for the TA to return fewer translations than requested and no error indication is sent unless there is an error in accessing the data.

If the Translation Completion contains multiple translations, all translations must have the same indicated size. Also, successive translations must apply to the virtual address range that abuts the previous translation in the same completion.

If a translation has both R = 0b and W = 0b, the TA must still set the Size field and the lower bits of the Translated Address field used to encode the completion size to appropriate values.

Each translation in a Translation Completion will have some overlap with the implied memory range of the Translation Request (see Section 10.2.2).

A successful Translation Completion must consist of one or two CplDs.

If a Translation Completion CplD has a Byte Count that is greater than four times the Length field, then additional CplDs are required to complete the transaction.

If a Translation Completion CplD has a Byte Count that is equal to four times the Length field, then the packet completes the request. For such a CplD, if the sum of Byte Count and Lower Address is not a multiple of RCB, then the CplD is the last of a sequence. If no previous CplD for this request has been received, an error has occurred and all translation values should be discarded.

> Note: There are multiple reasons that the TA may truncate the results of the completion. For example, the request might ask for a range of addresses, not all of which are defined. This could occur if the first translation is valid but located at the end of a page of translations. The TA, in looking up the next page of translations, may find that the page is not valid so the addresses are not valid. The range of addresses that are valid would be returned and no error indicated. When truncating a Translation Completion the TA is not allowed to pad the response with invalid entries (R = 0b, W = 0b).

> Note: There are multiple reasons that the TA may break a Translation Completion into multiple TLPs. As an example, if the virtual address of the Translation Completion resolves to a table access that crosses an RCB boundary of the memory system, the completion to the

TA may be broken into multiple completions by the memory. Rather than require that the TA accumulate the results, it is allowed to send each portion of the Translation Completion to a Function when it is received from the system memory.

10.3 ATS Invalidation

ATS uses the messages shown in this section to maintain consistency between the TA and the ATC. This specification assumes there is a single TA associated with each ATC. The TA (in conjunction with its associated software) must ensure that the address translations cached in the ATC are not stale by issuing Invalidate Requests.

10.3.1 Invalidate Request

When a translation is changed in the TA and that translation might be contained within an ATC in a Function, the TA (in conjunction with its associated software) must send an Invalidate Request to the ATC to maintain proper synchronization between the ATPT and the ATC. An Invalidate Request is used to clear a specific subset of the address range from the ATC. Invalidate Requests are constrained to cover power of 2 multiple of 4096-byte pages.

The format of an Invalidate Request is shown in Figure 10-13.

+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
0	Fmt 1 1		Type 1 0010					R	TC			R	Attr R	R		T D	E P	Attr R	R	R				Length 00 0000 0010							
Requester ID																0	0	0	ITag					Message Code 0000 0001							
Device ID																Reserved															
Reserved																															

A-0584A

Figure 10-13: Invalidate Request Message

The Invalidate Request is a MsgD transaction with 64 bits of data. Invalidate Request messages may be sent in any TC. The ITag field is constrained to the values 0 to 31 and is used by the TA to uniquely identify requests it issues. A TA must ensure that once an ITag is used, it is not reused until either released by the corresponding Invalidate Completions or by a vendor-specific timeout mechanism (see below).

The TA may have a single pool of ITag values for all invalidates that it issues or it may have a pool for each Device ID or any other combination. A Device with multiple ATCs on different Functions must manage the ITags separately for each Requester ID.

The address range specified in an Invalidate Request may span one or more STU 4096-byte pages. Invalidation ranges are required to be naturally aligned and may not be smaller than STU 4096-byte pages. Upon receiving an Invalidate Request with a range less than STU an ATC may either (1) signal an Unsupported Request or (2) round the range of the request up to a value greater than or equal to the STU.

IMPLEMENTATION NOTE

Invalidate Completion Timeout

Devices should respond to Invalidate Requests within 1 minute (+50% -0%).Having a bounded time permits an ATPT to implement Invalidate Completion Timeouts and reuse the associated ITag values. ATPT designs are implementation-specific. As such, Invalidate Completion Timeouts and their associated error handling are outside the scope of this specification.

The content of the payload is the untranslated address range to be invalidated. The payload format is shown in Figure 10-14:

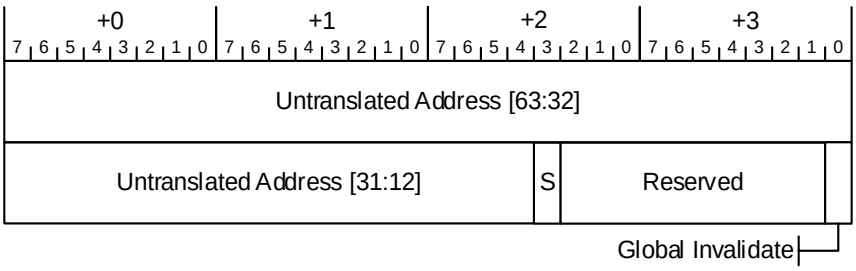


Figure 10-14: Invalidate Request Message Body

The S field is used to indicate if the range being invalidated is greater than 4096 bytes. Its meaning is the same as for the Translation Completion (see Section 10.2.3.1 and Section 10.2.3.2).

The Global Invalidate bit indicates that the Invalidation Request Message affects all PASID values (see Section 10.3.8). This bit is Reserved unless the Invalidation Request has a PASID TLP Prefix. The bit is ignored by the ATC if Global Invalidate Supported bit is Clear (see Section 10.3.8).

10.3.2 Invalidate Completion

When a Function completes an Invalidate operation, it will send one or more Invalidate Completion messages to the TA. These messages must be tagged with information extracted from the Invalidate Request to enable the TA to associate the Invalidate Completions with the Invalidate Request.

The format of the Invalidate Completion message is shown in Figure 10-15.

+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
0	Fmt 0 1		Type 1 0 0 1 0					R	TC			R	Attr R	0	0	T D	E P	Attr 0 0		R		0	0	0	0	0	0	0	0	0	0
Requester ID																Reserved								Message Code 0000 0010							
Device ID																Reserved														CC	
ITag Vector																															

A-0586A

Figure 10-15: Invalidate Completion Message Format

The Invalidate Completion message is a Msg transaction routed by ID. The Requester ID field of the Invalidate Completion message is set to the Requester ID of the Function containing the ATC. The Device ID field of the Invalidate Completion is set to the Requester ID of the TA. The ATC may derive the Requester ID of the TA from the Requester ID field of the corresponding Invalidate Request. Alternatively, since the ATC is only associated with a single TA, the ATC may sample and store the Requester ID from the first Invalidate Request following a Fundamental Reset or FLR. Subsequent Invalidate Completion messages may use this value to set the Device ID field of Invalidate Completion messages.

The Completion Count (CC) field indicates the number of individual Invalidate Completion messages that must be sent for the associated Invalidate Request. Setting the CC field to 0 indicates that eight responses must be sent. The TA is responsible for collecting all the responses associated with a given Tag before considering the corresponding Invalidate Request to be complete.

Invalidate Completion messages may be sent on any TC, independent of the TC the originating Invalidate Request was received. This enables implementations to utilize the Invalidate Completion to push outstanding transactions to the TA to guarantee the required invalidation semantics are met. Implementations that utilize a single Upstream TC are required to send a single Invalidate Completion in the utilized TC.

The ITag Vector field is used to indicate which Invalidate Request has been completed. Each of the 32 possible ITag field values from the Invalidation Request is represented by a single bit in the ITag Vector field. The least significant bit (bit 0; i.e., the right-most bit in the schematic representation of the Invalidate Completion message shown in Figure 10-15) of the ITag Vector field corresponds to the ITag field value of 0. The most significant bit (bit 31) of the ITag Vector field corresponds to the ITag field value of 31. Implementations are allowed to coalesce multiple Invalidate Completions by setting multiple ITag Vector bits in a single message provided the following conditions are met:

- The Invalidate Completions flow in the same TC.
- The Invalidate Completions have the same CC value.
- All fragments of an Invalidate Completion must have identical Request ID, CC, and ITag Vector fields.

A TA that receives an Invalidation Completion for an ITag that has no outstanding Invalidation Request shall report this error using implementation specific mechanisms. One possible such mechanism is to report the Invalidation Completion as an Unexpected Completion (UC).

Functions that do not support ATS will treat an Invalidate Request as UR.

Functions supporting ATS are required to send an Invalidate Completion in response to a Invalidate Request independent of whether the Bus Master Enable bit is Set or not. Note that the above conditions must be satisfied even when Bus Master Enable is Cleared. The method for a device to achieve this is implementation dependent.

IMPLEMENTATION NOTE

Bus Master Enable Change

When Bus Master Enable changes from Set to Clear, no further memory requests should be queued. It is possible that queued write requests are present when BME is Cleared. These requests could block an Invalidate Completion. These requests must be either sent or dropped. This will ensure that all outstanding write transactions that are potentially dependent upon the outstanding invalidation are complete.

10.3.3 Invalidate Completion Semantics

Before an ATC can return an Invalidate Completion for a given Invalidate Request, it must ensure the following conditions are satisfied:

- All new requests initiated by the Function will not utilize stale address translations.
- All outstanding read requests utilizing translated address matching the invalidated range have either completed or been tagged to be discarded (method to discard is implementation specific).
- All outstanding posted writes utilizing a translated address matching the invalidated range have been pushed to the TA. The ATC is required to send a copy of the Invalidate Completion message in each TC in which a posted write has been issued but not known to have been pushed to the TA. The CC field must be set to the same value in each copy of the Invalidate Completion message indicating number of copies sent. The TA is responsible for collecting all sent responses before considering the invalidation to be complete.

IMPLEMENTATION NOTE

Implied TC Flushing

When making the decision as to which TC to send Invalidate Completions, an ATC may infer, in an implementation specific manner, that an issued posted write has been pushed to the TA. For example, a Function that has sent a read transaction to a destination above the TA and received its corresponding response may infer that any preceding posted writes issued in the same TC have been pushed to the TA.

10.3.4 Request Acceptance Rules

In accord with the request acceptance rules enumerated in this section, a Function is not allowed to create a dependency in which the acceptance of a posted transaction is dependent upon the transmission of a posted transaction. Given Invalidate Requests and Invalidate Completions both are posted transactions, Functions must not make the acceptance of an Invalidate Request dependent upon the transmission of an Invalidate Completion. The method for achieving this is implementation specific.

A Function with an ATS capability in its configuration space must be able to accept Invalidate Requests and send Invalidate Completions even if ATS is not enabled.

IMPLEMENTATION NOTE

Invalidate Queue Depth

An ATC is only associated with a single TA. Each TA is limited to a total of 32 outstanding invalidations to any given ATC. This limits the number of outstanding Invalidation Requests active to a single ATC to 32. To avoid a post-to-post dependency, an ATC is required to accept up to 32 Invalidation Requests.

An ATC may choose to implement a maximally sized input queue holding Invalidate Requests. Alternatively, an ATC may choose to implement a maximally sized output queue holding Invalidate Completions. Note that queuing Invalidate Completions requires significantly less state per entry resulting in a potentially more efficient implementation than input queue buffering.

Note that the choice of whether to implement input queuing or output queuing (or a hybrid of both) has no impact on ensuring deadlock free behavior. But implementation choices with regard to queuing may have a significant impact on performance (see Section 10.3.5).

10.3.5 Invalidate Flow Control

Due to the variety of caching architectures and queuing strategies, implementations may vary greatly with respect to invalidation latency and throughput. It is possible that a TA may generate Invalidate Requests at a rate that exceeds the average ATC service rate. When this happens, the credit based flow control mechanisms will throttle the TA issue rate. A side effect of this is congestion spreading to other channels and Links through the credit based flow control mechanism. Depending on the frequency and duration of this congestion, performance may suffer. It is highly recommended that TA and its associated software implement higher level flow control mechanisms.

To assist with the implementation of Invalidate Flow Control, an ATC must publish the number of Invalidate Requests it can buffer before back pressuring the Link. This field applies to all invalidations serviced by the Function, independent of the size of the invalidation. This value is communicated in the Invalidate Queue Depth field in the ATS capability structure (see Section 7.8.8). A value of 0 0000b indicates that invalidate flow control is not necessary to this Function.

IMPLEMENTATION NOTE

Invalidate Flow Control

A Function may indicate that invalidate flow control is not required when one or more of the following is true:

1. The Function can handle invalidations at the maximum arrival rate of Invalidate Requests.
2. The Function will not or very rarely cause Link backpressure (performance loss is negligible).
3. The Function can fully buffer the maximum number of incoming invalidations without back pressuring the Link.

10.3.6 Invalidate Ordering Semantics

Invalidate Requests and Translation Completions may be sent using different TC and are, therefore, unordered with respect to each other (from the Link's perspective). An ATC must ensure that the proper invalidation behavior is maintained when an Invalidate Request bypasses a Translation Completion to an overlapping region.

An ATC must "snoop" its outstanding translation request queue against all arriving Invalidate Requests. When snooping a request for a N*STU sized translation (N is a power of 2), the ATC must snoop the range of addresses starting at the STU aligned region containing the specified address and ending (N-1) STU size pages later.

If an Invalidate Request overlaps the address range in an outstanding Translation Request, the Translation Request must be tagged as invalid and the results of its corresponding Translation Response must be discarded prior to transmission of the Invalidate Completion. If the Translation Response is received before the Invalidate Completion is sent, an implementation is free to issue requests utilizing the translation result provided the Invalidate Completion Semantics (see Section 10.3.3) are satisfied.

IMPLEMENTATION NOTE

Request Range Overlap in Invalidations

In the description above, N is the number of STU sized translations that were requested in the Translation Request. This is equal to (Length field in Translation Request)/2.

As an example:

STU is 00 0010b indicating 16384-byte pages.

An outstanding Translation Request has a Length field of 00 0000 0100b indicating two translations covering a range of 32768 bytes.

The high-order 48 bits of the Translation Request are 0000 0FFF FFFFh.

The low-order 16 bits of the address in the request are 11xx xxxx xxxx xxxxb indicating that the translation request covers a range that overlaps a 32768-byte boundary (in fact, the request crosses a 16-TB boundary).

If two translations are returned, they would cover the two STU sized regions at 0000 0FFF FFFF C000h and 0000 1000 0000 0000h.

An Invalidate Request is received with the high-order 48 bits of 0000 1000 0000h and the low-order 16 bits of 0001 1xxx xxxx xxxxb.

The ATC must detect that a translation associated with a portion of the Translation Request is now invalidated and the Translation Completion associated with the invalidated region must be discarded (for simplification, the ATC is allowed to discard all of the Translation Completion).

It should be noted that, processing of the Invalidate Requests is simplified if Translation Requests do not cross alignment boundaries of the request. The Translation Request from the above example is not aligned to a 32768-byte boundary. If it were broken into two requests, it would be simpler to associate the range of the Invalidate Request with the address in the Translation Request. Breaking the Translation Requests into aligned requests is not a requirement.

10.3.7 Implicit Invalidation Events

The following events will cause the invalidation of all ATC entries:

- Conventional Reset (all forms)
- Function Level Reset
- E field in ATS Capability changes from Clear to Set

The following events will cause the invalidation of all non-Global Mapping ATC entries that were requested using a specific PASID:

- Stopping the use of a PASID as defined inthis specification.

No explicit Invalidate Completion message is sent when these implied invalidate events occur.

IMPLEMENTATION NOTE

Implicit Invalidation and PASID

Software may not change any of the PASID enable bits when the E field in the ATS Capability is Set. The invalidation that occurs when software Sets the E field also invalidates ATC entries with an associated PASID value.

10.3.8 PASID TLP Prefix and Global Invalidate

The requirements in this section apply to Functions that support the PASID TLP Prefix. For Invalidation Requests that have a PASID TLP Prefix, the ATC shall:

- Optionally signal Unsupported Request (UR) if the associated PASID value is greater than or equal to $2^{\text{Max PASID Width}}$. This error may be signaled anytime an out of range PASID value is present, even when the PASID value is ignored (see below).
- Return an Invalidation Completion if PASID Enable is Clear.
- If the Function supports Global Invalidate (see Section 7.8.8.2):
  - If the Global Invalidate bit in the Request is Set, invalidate Global and non-Global Mapping entries in the ATC within the indicated memory range associated with any PASID value and return an Invalidation Completion. The PASID value in the PASID TLP Prefix is ignored.
  - If the Global Invalidate bit in the Request is Clear, invalidate only non-Global Mapping entries in the ATC within the indicated memory range that were requested using the associated PASID value and return an Invalidation Completion.

Global Mapping entries in the ATC for some or all of the indicated memory range may be retained.

- If the Function does not support Global Invalidate (see Section 7.8.8.2), invalidate entries in the ATC within the indicated memory range that were requested using the associated PASID value and return an Invalidation Completion.
- If no matching entries are present in the ATC, invalidate no ATC entries and return an Invalidation Completion.

For Invalidation Requests that do not have a PASID TLP Prefix, the ATC shall:

- Invalidate ATC entries within the indicate memory range that were requested without a PASID value.
- Invalidate ATC entries at all addresses that were requested with any PASID value.

10.4 Page Request Services

The general model for a page request is as follows:

1. A Function determines that it requires access to a page for which an ATS translation is not available.
2. The Function causes the associated Page Request Interface to send a Page Request Message to its RC. A Page Request Message contains a page address and a Page Request Group (PRG) index. The PRG index is used to identify the transaction and is used to match requests with responses.
3. When the RC determines its response to the request (which will typically be to make the requested page resident), it sends a PRG Response Message back to the requesting Function.
4. The Function can then employ ATS to request a translation for the requested page(s).

A Page Request Message is a PCIe Message Request that is Routed to the Root Complex with a Message Code of 4 (0000 0100b). The mechanism employed at the RC to buffer requests is implementation specific. The only requirement is that an RC not silently discard requests.

All Page Request Messages and PRG Response Messages travel in PCIe Traffic Class 0. A Page Request Message or PRG Response Message with a Traffic Class other than 0 shall be treated as Malformed TLPs by the RC or endpoint that receives the same. Intermediate routing elements (e.g., Switches) shall not detect this error.

The Relaxed Ordering and ID Base Ordering bits in the Attr field of Page Request Messages and PRG Response messages may be used. The No Snoop bit in the Attr field is reserved.

The page request service allows grouping of page requests into Page Request Groups (PRGs). A PRG can contain one or more page requests. All pages in a PRG are responded to en mass by the host. Individual pages within a PRG are requested with independent Page Request Messages and are recognized as belonging to a common PRG by sharing the same PRG index. The last request of a PRG is marked as such within its Page Request Message. One request credit is consumed per page request (not per PRG).

A PRG Response Message is a PCIe Message Request that is Routed by ID back to the requesting Function. It is used by system software to alert a Function that the page request(s) associated with the corresponding PRG has (have) been satisfied. The page request mechanism does not guarantee any request completion order and all requests are inherently independent of all other concurrently outstanding requests. If a Function requires that a particular request be completed before another request, the initial request will need to complete before the subsequent request is issued. It is valid for a Function to speculatively request a page without ascertaining its residence state and/or to issue multiple concurrently outstanding requests for the same page.

A Page Request Interface is allocated a specific number of page request message credits. An RC (system software) can divide the available credits in any manner deemed appropriate. Any measures the host chooses to employ to ensure that credits are correctly metered by Page Request Interfaces (a Page Request Interface is not using more than its allocation) is an implementation choice. A Page Request Interface is not allowed to oversubscribe the available number of requests (doing so can result in the page request mechanism being disabled if the buffer limit is exceeded at the root). A

Page Request Interface's page request allocation is static. It is determined when the Page Request Interface is enabled and can only be changed by disabling and then re-enabling the interface.

10.4.1 Page Request Message

A Function uses a Page Request Message to send page requests to its associated host. A page request indicates a page needed by the Function. The Page Request Interface associated with a Function is given a specific Page Request allocation. A Page Request Interface shall not issue page requests that exceed its page request allocation.

A page request contains the untranslated address of the page that is needed, the access permissions needed for that page, and a PRG index. A PRG Index is a 9-bit scalar that is assigned by the Function to identify the associated page request. Multiple pages may be requested using a single PRG index. When more than a single page is to be associated with a given PRG, the Last flag in the Page Request Record is cleared in all the requests except the last request associated with a given PRG (the flag is set in the last request). Page requests are responded to en mass. No response is possible (except for a Response Failure error) until the last request of a PRG has been received by the root. The number of PRGs that a Function can have outstanding at any given time is less than or equal to the associated Page Request Interface's Outstanding Page Request Allocation. It is valid for a request group to contain multiple requests for the same page and for multiple outstanding PRGs to request the same page.

The first two DWs of a Page Request Message contain a standard PCIe message header. The second two DWs of the message contain page request specific data fields.

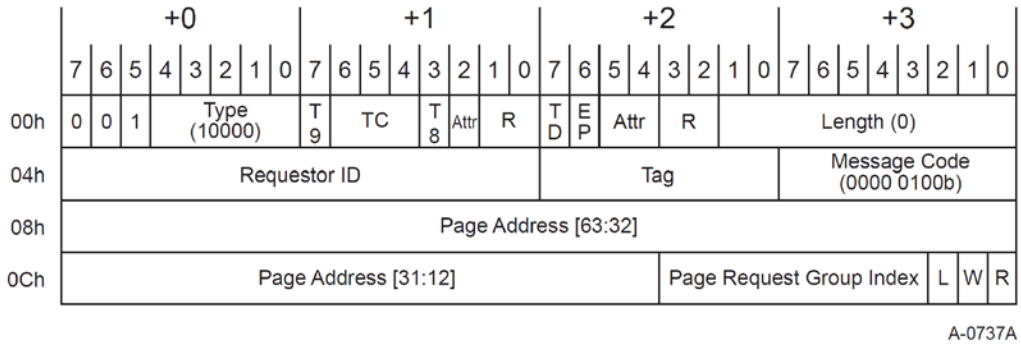


Figure 10-16: Page Request Message

Table 10-5: Page Request Message Data Fields

Field	Meaning
R	**Read Access Requested** – This field, when Set, indicates that the requesting Function seeks read access to the associated page. When Clear, this field indicates that the requesting Function will not read the associated page. The R field must be Set for Page Requests with a PASID TLP Prefix that has the Execute Requested bit Set. If R and W are both Clear and L is Set, this is a Stop Marker (see Section 10.4.1.2.1).

Field	Meaning
W	**Write Access Requested** – This field, when Set, indicates that the requesting Function seeks write access and/or zero-length read access to the associated page. When Clear, this field indicates that the requesting Function will not write to the associated page. Upon receiving a Page Request Message with the W field Set, the host is permitted to mark the associated page dirty. Thus, Functions must not issue such Requests unless the Function has been given explicit write permission. If R and W are both Clear and L is Set, this is a Stop Marker (see Section 10.4.1.2.1).
L	**Last Request in PRG** – This field, when Set, indicates that the associated page request is the last request of the associated PRG. A PRG can have a single entry, in which case the PRG consists of a single request in which this field is Set. When Clear, this field indicates that additional page requests will be posted using this record's PRG Index. If R and W are both Clear and L is Set, this is a Stop Marker (see Section 10.4.1.2.1).
Page Request Group Index	**Page Request Group Index** – This field contains a Function supplied identifier for the associated page request. A Function need not employ the entire available range of PRG index values. A host shall never respond with a PRG Index that has not been previously issued by the Function and that is not currently an outstanding request PRG Index (except when issuing a Response Failure, in which case the host need not preserve the associated request's PRG Index value in the error response).
Page Address	**Page Address** – This field contains the untranslated address of the page to be loaded. For pages larger than 4096 bytes, the least significant bits of this field are ignored. For example, the least significant bit of this field is ignored when an 8096-byte page is being requested.

IMPLEMENTATION NOTE

Last Bit and Relaxed Ordering

If multiple page requests are associated with a single PRG index, the last page request of a PRG should have the Relaxed Ordering attribute bit Clear in addition to having the Last flag Set. All other page request messages may have the Relaxed Ordering attribute bit set to any value.

10.4.1.1 *PASID TLP Prefix Usage*

The PASID Extended Capability indicates whether a Function supports PASID TLP Prefixes and whether it is enabled to send and receive them.

Functions that support the PASID TLP Prefix are permitted to send a PASID TLP Prefix on Page Request Messages. The PASID field contains the process address space of the page being requested and the Execute Requested and Privileged Mode Requested bits indicate the access being requested.

If one Page Request Message in a PRG has a PASID TLP Prefix, all Page Request Messages in that PRG must contain identical PASID TLP Prefixes. Behavior is undefined when the PASID TLP Prefixes are inconsistent.

Functions that support the PASID TLP Prefix and have the PRG Response PASID Required bit Set (see Section 10.5.2.3), expect that PRG Response Messages will contain a PASID TLP Prefix if the associated Page Request Message had a PASID TLP Prefix. For such PRG Response Messages, the Execute Requested and Privileged Mode Requested bits are reserved and the PASID field contains the PASID from the associated Page Request Message.

10.4.1.2 *Managing PASID TLP Prefix Usage*

Ther are rules for stopping and starting the use of a PASID.

This section describes additional rules that apply to Functions that have issued Page Request Messages in a PASID that is being stopped. No additional rules are required to start the usage of the Page Request Interface for a PASID.

When stopping the use of a particular PASID, a Stop Marker Message may be optionally used to avoid waiting for PRG Response Messages before the Function indicates that the stop request for a particular PASID has completed.

To stop without using a Stop Marker Message, the Function shall:

1. Stop queueing new Page Request Messages for this PASID.
2. Finish transmitting any multi-page Page Request Messages for this PASID (i.e. send the Page Request Message with the L bit Set).
3. Wait for PRG Response Messages associated any outstanding Page Request Messages for the PASID.
4. Indicate that the PASID has stopped using a device specific mechanism. This mechanism must indicate that a Stop Marker Message will not be generated.

To stop with the use of a Stop Marker Message the Function shall:

1. Stop queueing new Page Request Messages for this PASID.
2. Finish transmitting any multi-page Page Request Messages for this PASID (i.e. send the Page Request Message with the L bit Set).
3. Internally mark all outstanding Page Request Messages for this PASID as stale. PRG Response Messages associated with these requests will return Page Request Allocation credits and PRG Index values but are otherwise ignored.[152]
4. Indicate that the PASID has stopped using a device specific mechanism. This mechanism must indicate that a Stop Marker Message will be generated.
5. Send a Stop Marker Message to indicate to the host that all subsequent Page Request Messages for this PASID are for a new use of the PASID value.

Note: Steps 4 and 5 may be performed in either order, or in parallel.

[152] Page Request Allocation is shared across all PASIDs of the Function (see Section 1.5.2.5). If PRG Response PASID Required is Clear, PRG Index values are shared across all PASIDs of the Function (see Section 1.5.1.3).

10.4.1.2.1 Stop Marker Messages

A Stop Marker Message indicates that a Function has stopped using the Page Request Interface and has transmitted all pending Page Request Messages for a specific PASID. Stop Marker Messages are strongly ordered with respect to Page Request Messages and serve to push Page Request Messages toward the Host. When the Host receives the Stop Marker Message, this indicated that all Page Request Messages associated with the PASID being stopped have been delivered and that any subsequent Page Request Message with the same PASID value are associated with a new incarnation of that PASID value.

Stop Marker Messages do not have a response. They do not have a PRG Index and do not consume Page Request allocation (see Section 10.5.2.5).

The Stop Marker Message bit layout is shown in Figure 10-17.

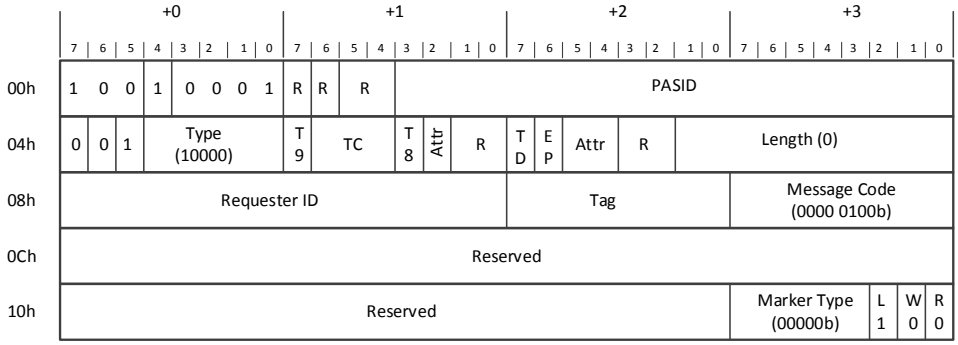


Figure 10-17: Stop Marker Message

A Stop Marker Message is encoded as a Page Request Message that contains a PASID TLP Prefix with the following exceptions:

- The L, W and R fields contain 1b, 0b and 0b respectively.
- The Untranslated Address field and upper bits of the PRG Index field are Reserved.
- The Marker Type field contains 00000b to indicate that this is a Stop Marker Message.

The Execute Requested and Privileged Mode Requested bits in the PASID TLP Prefix are Reserved.

- The Traffic Class must be 0.
- The Relaxed Ordering attribute bit must be Clear.
- The ID Based Ordering attribute bit may be Set.

Behavior is undefined if a Stop Marker Message is received and any of the following are true:

- Marker Type not equal to 00000b.
- No PASID TLP Prefix is present.
- The PASID value does not match an outstanding stop request.
- An incomplete Page Request Message for the PASID is outstanding (i.e. for some PRG Index, the most recently received Page Request Message did not have the L bit Set).

10.4.2 Page Request Group Response Message

System hardware and/or software communicate with a Function's page request interface via PRG Response Messages. A PRG Response Message is used by a host to signal the completion of a PRG, or the catastrophic failure of the interface. A single PRG Response Message is issued in response to a PRG, independent of the number of page requests associated with the PRG. There is no mechanism for indicating a partial request completion or partial request failure. If any of the pages associated with a given PRG cannot be satisfied, then the request is considered to have failed and the reason for the failure is supplied in the PRG Response Message. The host has no obligation to partially satisfy a multi-page request. If one of the requested pages cannot be made resident, then the entire request can, but need not, be discarded. That is, the residence of pages that share a PRG with a failed page request, but that are not associated with the failure, is indeterminate from the Function's perspective.

There are four possible Page Request failures:

1. The requested page is not a valid Untranslated Address.
2. PASID TLP Prefix support exists, the Page Request has a PASID TLP Prefix, and either PASID TLP Prefix usage is not enabled for this request, the PASID value is not valid, or the Execute Requested bit is Set when R is Clear.[153]
3. The requested page does not have the requested access attributes (including Execute permission and/or Privileged Mode access when the Page Request has a PASID TLP Prefix).
4. The system is, for an unspecified reason, unable to respond to the request. This response is terminal (the host may no longer respond to any page requests and may not supply any further replies to the Function until the Function's page request interface has been reset). For example, a request that violates a Function's assigned request limit or overflows the RC's buffering capability may cause this type of failure.

A Function's response to Page Request failure cases 1, 2, and 3 above is implementation dependent. The failure is not necessarily persistent, that is, a failed request may, in some instances succeed if re-issued. The range of possibilities precludes the precise specification of a generalized failure behavior, though on a per Function basis, the response to a failure will be an implementation dependent behavior.

All responses are sent to their associated Functions via PRG Response Messages. A Function must be capable of sinking multiple consecutive messages without losing any information. To avoid deadlock, a Function must able to process PRG Response Messages for all of the Function's outstanding Page Request Messages without depending on the Function sending or receiving any other TLP.[154] A PRG Response Message is an ID routed PCIe message. The only Page Request Interface specific fields in this message are the Response Code and PRG. All other fields are standard PCIe message fields. (Note: these messages are routed based on the ID in bytes 8 and 9; with bytes 4 and 5 containing the host's RID.)

---

[153] Behavior when PASID TLP Prefix support does not exist is defined in the *PCI Express Base Specification*.

[154] For example, processing a PRG Response Message that causes the Function to send a TLP Upstream must not block processing of subsequent downstream TLPs even if the Upstream TLP is delayed by flow control.

Receipt of a PRG Response Message that contains a PRG Index that is not currently outstanding at a Function shall result in the UPRGI flag in the PRI Extended Capability being Set and in the issuance of an Unexpected Response (UR) by the Function containing the PRI Extended Capability. With the exception of setting the UPRGI flag, a Function treats receipt of an unexpected PRG Index in exactly the same manner that it treats receipt of a standard PCIe read completion for which there is no outstanding request.

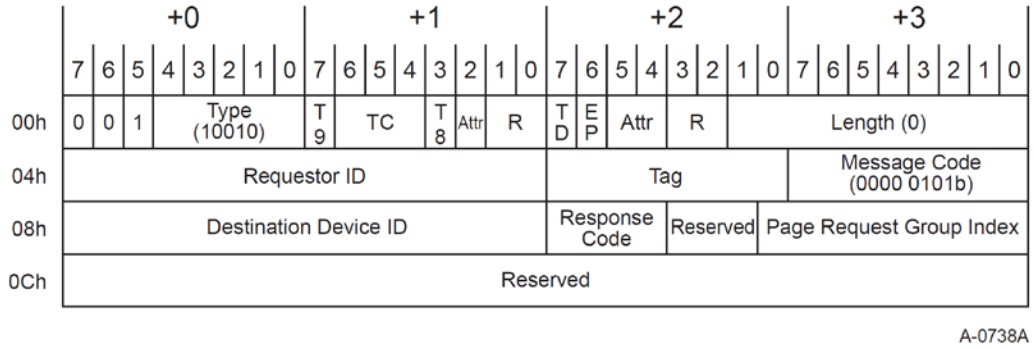


Figure 10-18: PRG Response Message

Table 10-6: PRG Response Message Data Fields

Field	Meaning
Page Request Group Index	**Page Request Group Index** – This field contains a Function supplied index to which the RC is responding. A given PRG Index will receive exactly one response per instance of PRG (with the possible exception of a Response Failure).
Response Code	**Response Code** – This field contains the response type of the associated PRG. The encodings are presented in Section 10.4.2.1.

10.4.2.1 *Response Code Field*

The values and meaning for the Response Code field are listed in Table 10-7.

Table 10-7: Response Codes

Value	Status	Meaning
0000b	Success	All pages within the associated PRG were successfully made resident.
0001b	Invalid Request	One or more pages within the associated PRG do not exist or requests access privilege(s) that cannot be granted. Unless the page mapping associated with the Function is altered, re-issuance of the associated request will never result in success.
1110b:0010b	Unused	Unused Response Code values. A Function receiving such a message shall process it as if the message contained a Response Code of Response Failure.
1111b	Response Failure	One or more pages within the associated request group have encountered/caused a catastrophic error. This response disables the Page Request Interface at the Function. Any pending page requests for other PRGs will be satisfied at the convenience of the host. The Function shall ignore any subsequent PRG Response Messages, pending re-enablement of the Page Request Interface.

10.4.2.2 *PASID TLP Prefix Usage*

If a Page Request has a PASID TLP Prefix, the corresponding PRG Response Message may optionally contain one as well.

If the PRG Response PASID Required bit is Clear, PRG Response Messages do not have a PASID TLP Prefix.

If the PRG Response PASID Required bit is Set, PRG Response Messages have a PASID TLP Prefix if the Page Request also had one. The Function is permitted to use the PASID value from the prefix in conjunction with the PRG Index to match requests and responses.

In PASID TLP Prefixes attached to PRG Response Messages, the Execute Requested and Privileged Mode Requested bits are Reserved and the PASID value is copied from the PASID value of the Page Request.

10.5 Configuration

10.5.1 ATS Extended Capability Structure

Each Function that supports ATS (capable of generating Translation Requests) must have the ATS Extended Capability structure in its extended configuration space. It is permitted to be implemented by Endpoints or Root Complex Integrated Endpoints.

Figure 10-19 details allocation of the register fields in the ATS Extended Capability structure.

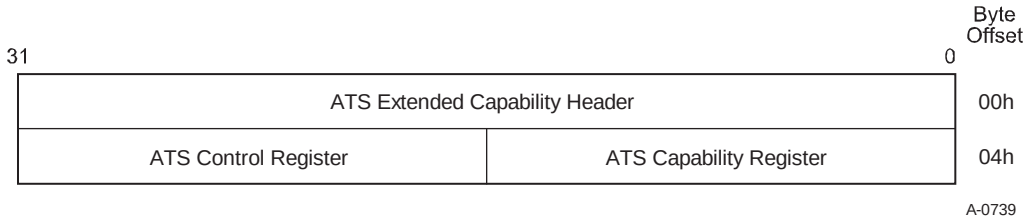


Figure 10-19: ATS Extended Capability Structure

10.5.1.1 *ATS Extended Capability Header*

Figure 10-20 details allocation of the register fields in the ATS Extended Capability header; Table 10-8 provides the respective field definitions.

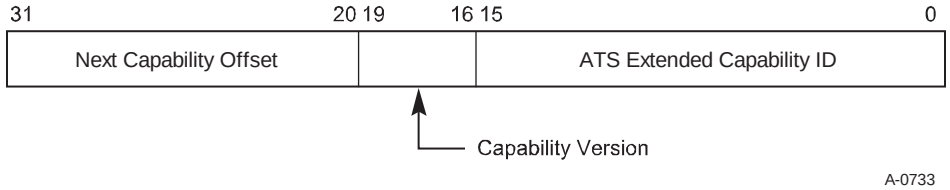


Figure 10-20: ATS Extended Capability Header

Table 10-8: ATS Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**ATS Extended Capability ID** – Indicates the ATS Extended Capability structure. This field must return a Capability ID of 000Fh indicating that this is an ATS Extended Capability structure.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.	RO
31:20	**Next Capability Offset** – The offset to the next PCI Extended Capability structure or 000h if no other items exist in the linked list of capabilities.	RO

10.5.1.2 *ATS Capability Register*

Figure 10-21 details the allocation of register fields of an ATS Capability register; Table 10-9 provides the respective bit definitions.

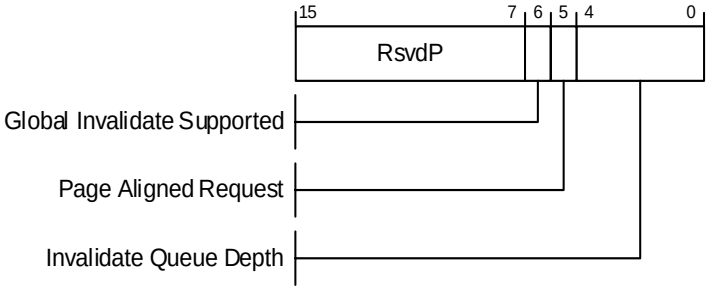


Figure 10-21: ATS Capability Register

Table 10-9: ATS Capability Register

Bit Location	Register Description	Attributes
4:0	**Invalidate Queue Depth** – The number of Invalidate Requests that the Function can accept before putting backpressure on the Upstream connection. If 0 0000b, the Function can accept 32 Invalidate Requests.	RO
5	**Page Aligned Request** – If Set, indicates the Untranslated Address is always aligned to a 4096 byte boundary. Setting this field is recommended. This field permits software to distinguish between implementations compatible with this specification and those compatible with an earlier version of this specification in which a Requester was permitted to supply anything in bits [11:2].	RO
6	**Global Invalidate Supported** – If Set, the Function supports Invalidation Requests that have the Global Invalidate bit Set. If Clear, the Function ignores the Global Invalidate bit in all Invalidate Requests (see Section 10.3.8). This bit is 0b if the Function does not support the PASID TLP Prefix.	RO

10.5.1.3 *ATS Control Register*

Figure 10-22 details the allocation of register fields of an ATS Control register; Table 10-10 provides the respective bit definitions.

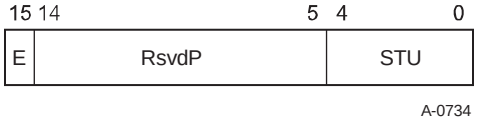


Figure 10-22: ATS Control Register

Table 10-10: ATS Control Register

Bit Location	Register Description	Attributes
4:0	**Smallest Translation Unit (STU)** – This value indicates to the Function the minimum number of 4096-byte blocks that is indicated in a Translation Completions or Invalidate Requests. This is a power of 2 multiplier and the number of blocks is 2^{STU}. A value of 0 0000b indicates one block and a value of 1 1111b indicates 2^{31} blocks (or 8 TB total) Default value is 0 0000b.	RW
15	**Enable (E)** – When Set, the Function is enabled to cache translations. Behavior is undefined if this bit is Set and the value of the PASID Enable, Execute Requested Enable, or Privileged Mode Requested bits are changed. Default value is 0b.	RW

10.5.2 Page Request Extended Capability Structure

A Page Request Extended Capability Structure is used to configure the Page Request Interface mechanism. A Multi-Function Endpoint or Root Complex Integrated Endpoint Device may implement a Page Request Interface and the associated capability on any Function within the Device. For SR-IOV, a single Page Request Interface is permitted for the PF and is shared between the PF and the associated VFs. The PF implements this capability and the VFs do not. Every Page Request Interface mechanism operates independently.

Note: For SR-IOV, even though the Page Request Interface is shared between PFs and VFs, it sends the requesting Function's ID (PF or VF) in the Requester ID field of the Page Request Message and expects the requesting Function's ID in the Destination Device ID field of the resulting PRG Response Message.

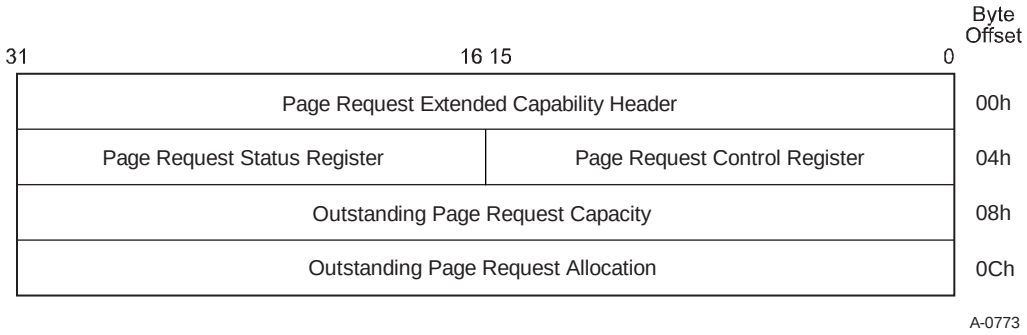


Figure 10-23: Page Request Extended Capability Structure

10.5.2.1 *Page Request Extended Capability Structure*

Figure 10-24: details allocation of the register fields in the Page Request Extended Capability header; Table 10-11 provides the respective field definitions.

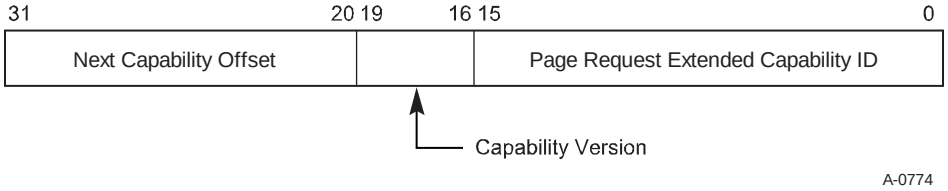


Figure 10-24: Page Request Extended Capability Header

Table 10-11: Page Request Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**Page Request Extended Capability ID** – Indicates that the associated extended capability structure is a Page Request Extended Capability. This field must return a Capability ID of 0013h.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.	RO
31:20	**Next Capability Offset** – The offset to the next PCI Extended Capability structure or 000h if no other items exist in the linked list of capabilities.	RO

10.5.2.2 *Page Request Control Register (04h)*

Figure 10-25: details allocation of the register fields in the Page Request Control Register; Table 10-12 provides the respective field definitions.

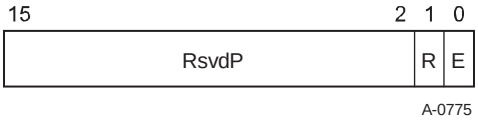


Figure 10-25: Page Request Control Register

Table 10-12: Page Request Control Register

Bit Location	Register Description	Attributes
0	**Enable (E)** – This field, when set, indicates that the Page Request Interface is allowed to make page requests. If this field is Clear, the Page Request Interface is not allowed to issue page requests. If both this field and the Stopped field are Clear, then the Page Request Interface will not issue new page requests, but has outstanding page requests that have been transmitted or are queued for transmission. When the Page Request Interface is transitioned from not-Enabled to Enabled, its status flags (Stopped, Response Failure, and Unexpected Response flags) are cleared. Enabling a Page Request Interface that has not successfully Stopped has indeterminate results. Default value is 0b.	RW
1	**Reset (R)** – When the Enable field is clear, or is being cleared in the same register update that sets this field, writing a 1b to this field, clears the associated implementation dependent page request credit counter and pending request state for the associated Page Request Interface. No action is initiated if this field is written to 0b or if this field is written with any value while the Enable field is Set. Reads of this field return 0b	RW

10.5.2.3 *Page Request Status Register (06h)*

Figure 10-26: details allocation of the register fields in the Page Request Error Register; Table 10-13 provides the respective field definitions.

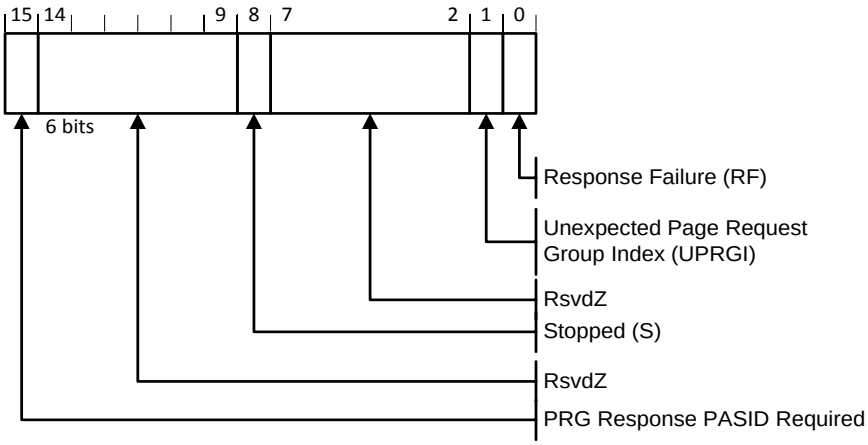


Figure 10-26: Page Request Status Register

Table 10-13: Page Request Status Register

Bit Location	Register Description	Attributes
0	**Response Failure (RF)** – This field, when Set, indicates that the Function has received a PRG Response Message indicating a Response Failure. The Function expects no further responses from the host (any received are ignored). This field is Set by the Function and Cleared when a one is written to the field. For SR-IOV, this field is Set in the PF if any associated Function (PF or VF) receives a PRG Response Message indicating Response Failure. Default value is 0b.	RW1C
1	**Unexpected Page Request Group Index (UPRGI)** – This field, when Set, indicates that the Function has received a PRG Response Message containing a PRG index that has no matching request. This field is Set by the Function and cleared when a one is written to the field. For SR-IOV, this field is Set in the PF if any associated Function (PF or VF) receives a PRG Response Message that does has no matching request. Default value is 0b.	RW1C
8	**Stopped (S)** – When this field is Set, the associated page request interface has stopped issuing additional page requests and that all previously issued Page Requests have completed. When this field is Clear the associated page request interface either has not stopped or has stopped issuing new Page Requests but has outstanding Page Requests. This field is only meaningful if Enable is Clear. If Enable is Set, this field is undefined. When the Enable field is Cleared, after having been previously Set, the interface transitions to the stopping state and Clears this field. After all page requests currently outstanding at the Function(s) have completed, this field is Set and the interface enters the disabled state. If there were no outstanding page requests, this field may be Set immediately when Enable is Cleared. Resetting the interface will cause an immediate transition to the disabled state. While in the stopping state, receipt of a Response Failure message will result in the immediate transition to the disabled state (Setting this field). For SR-IOV, this field is Set only when all associated Functions (PF and VFs) have stopped issuing page requests. Default value is 1b.	RO

Bit Location	Register Description	Attributes
15	**PRG Response PASID Required** – If Set, the Function expects a PASID TLP Prefix on PRG Response Messages when the corresponding Page Requests had a PASID TLP Prefix. If Clear, the Function does not expect PASID TLP Prefixes on any PRG Response Message. Function behavior is undefined if this bit is Clear and the Function receives a PRG Response Message with a PASID TLP Prefix. Function behavior is undefined if this bit is Set and the Function receives a PRG Response Message with no PASID TLP Prefix when the corresponding Page Requests had a PASID TLP Prefix. This bit is RsvdZ if the Function does not support the PASID TLP Prefix.	RO

10.5.2.4 *Outstanding Page Request Capacity (08h)*

This register contains the number of outstanding page request messages the associated Page Request Interface physically supports. This is the upper limit on the number of pages that can be usefully allocated to the Page Request Interface.

This register is Read Only.

10.5.2.5 *Outstanding Page Request Allocation (0Ch)*

This register contains the number of outstanding page request messages the associated Page Request Interface is allowed to issue (have outstanding at any given instance).

The number of PRGs a Page Request Interface has outstanding is less than or equal to the number of request messages it has issued. For example, if system software allocates 1000 messages to a Page Request Interface then a single PRG could use all 1000 of the possible requests. Conversely, at one request per PRG the Page Request Interface would run out of PRG indices (of which there are only 512) before it consumes all its page request credits. A Page Request Interface must pre-allocate its request availability for any given PRG, that is, all the requests required by a given PRG must be available before any of the requests may be issued.

This register is Read/Write. Behavior is undefined if this register is changed while the Enable flag is set. Behavior is undefined if this register is written with a value larger than Outstanding Page Request Capacity. Default value is 0.

When PASID TLP Prefix is supported, the Request Allocation remains associated with the Function and is shared across the Function as well as all PASIDs of the Function.

Stopping a PASID does not affect any allocation used by that PASID. The system should continue to respond with PRG Response Messages in order to return Page Request and PRG Index resources to the Function (see Section 10.4.2.1).

Stop Marker Messages consume buffering but are not included in this allocation (see Section 10.4.1.2.1). Systems should provide additional buffering for Stop Marker Messages and should limit the number of outstanding Stop Marker Messages to avoid overrunning this additional buffering.

This page intentionally left blank

A

A. Isochronous Applications

A.1. Introduction

The design goal of isochronous mechanisms in PCI Express is to ensure that isochronous traffic receives its allocated bandwidth over a relevant time period while also preventing starvation of other non-isochronous traffic.

Furthermore, there may exist data traffic that requires a level of service falling in between what is required for bulk data traffic and isochronous data traffic. This type of traffic can be supported through the use of Port arbitration within Switches, the use of TC labels [1:7], and optional additional VC resources. Policies for assignment of TC labels and VC resources that are not isochronous-focused are outside the scope of the PCI Express specification.

Two paradigms of PCI Express communication are supported by the PCI Express isochronous mechanisms: Endpoint-to-Root-Complex communication model and peer-to-peer (Endpoint-to-Endpoint) communication model. In the Endpoint-to-Root-Complex communication model, the primary isochronous traffic is memory read and write requests to the Root Complex and read completions from the Root Complex. Figure A-1 shows an example of a simple system with both communication models. In the figure, devices A, B, called Requesters, are PCI Express Endpoints capable of issuing isochronous request transactions, while device C and Root Complex, called Completers, are capable of being the targets of isochronous request transactions. An Endpoint-to-Root-Complex communication is established between device A and the Root Complex, and a peer-to-peer communication is established between device B and device C. In the rest of this section, Requester and Completer will be used to make reference to PCI Express elements involved in transactions. The specific aspects of each communication model will be called out explicitly.

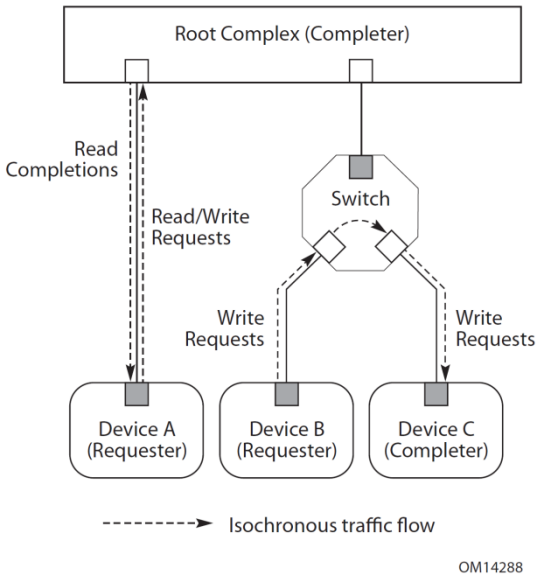


Figure A-1: An Example Showing Endpoint-to-Root-Complex and Peer-to-Peer Communication Models

Guaranteed bandwidth and deterministic latency require end-to-end configuration of fabric resources. If isochronous traffic is intermixed with non-isochronous traffic, it may not be possible to provide any guarantees/determinism as required by the application usage model. It is recommended that system software configure and assign fabric resources such that traffic intermix either does not occur or is such that the application usage model guarantees can be met. This can be accomplished by assigning dedicated VC resources and corresponding TC labels to the isochronous traffic flow(s) on a given path within the fabric.

Note that there may be one or more isochronous traffic flows per VC/TC label and it is up to system software to insure that the aggregation of these flows does not exceed the requisite bandwidth and latency requirements.

It is also possible for a fabric to support multiple isochronous traffic flows separated across multiple VC (a given flow cannot span multiple VC/TC labels).

In general, as long as the device can meet the isochronous bandwidth and latency requirements, there is nothing to preclude a single VC device from supporting isochronous traffic if multiple TC labels are supported to delineate such traffic from non-isochronous traffic within the fabric.

A.2. Isochronous Contract and Contract Parameters

In order to support isochronous data transfer with guaranteed bandwidth and deterministic latency, an isochronous contract must be established between a Requester/Completer pair and the PCI Express fabric. This contract must enforce both resource reservation and traffic regulation. Without such a contract, two basic problems, over-subscription and congestion, may occur as illustrated in Figure A-2. When interconnect bandwidth resources are over-subscribed, the increased latency may cause failure of isochronous service and starvation of non-isochronous services. Traffic congestion occurs when flow control credits are not returned possibly due to a higher than expected/provisioned packet injection rate. This may cause excessive service latencies for both isochronous traffic and non-isochronous traffic.

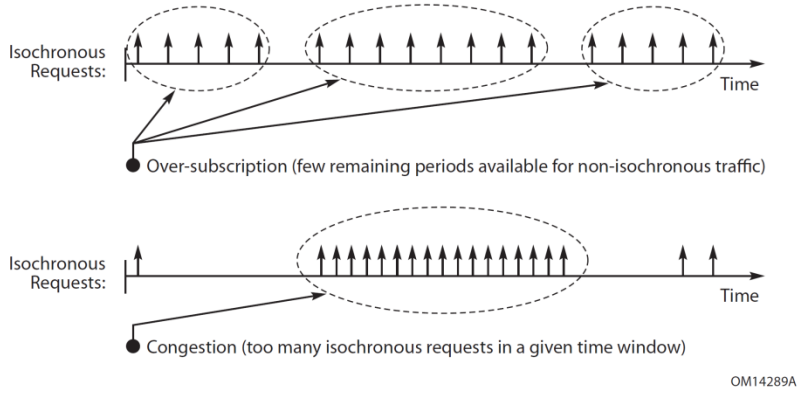


Figure A-2: Two Basic Bandwidth Resourcing Problems: Over-Subscription and Congestion

The isochronous transfer mechanism in this specification addresses these problems with traffic regulation including admission control and service discipline. Under a software managed admission control, a Requester must not issue isochronous transactions unless the required isochronous bandwidth and resource have been allocated. Specifically, the isochronous bandwidth is given by the following formula:

$$BW = \frac{N \cdot Y}{T}$$

The formula defines allocated bandwidth (BW) as a function of specified number (*N*) of transactions of a specified payload size (*Y*) within a specified time period (*T*). Another important parameter in the isochronous contract is latency. Based on the contract, isochronous transactions are completed within a specified latency (*L*). Once a Requester/Completer pair is admitted for isochronous communication, the bandwidth and latency are guaranteed to the Requester (a PCI Express Endpoint) by the Completer (Root Complex for Endpoint-to-Root-Complex communication and another PCI Express Endpoint for peer-to-peer communication) and by the PCI Express fabric components (Switches).

Specific service disciplines must be implemented by isochronous-capable PCI Express components. The service disciplines are imposed to PCI Express Switches and Completers in such a manner that

the service of isochronous requests is subject to a specific service interval (*t*). This mechanism is used to provide the method of controlling when an isochronous packet injected by a Requester is serviced. Consequently, isochronous traffic is policed in such manner that only packets that can be injected into the fabric in compliance with the isochronous contract are allowed to make immediate progress and start being serviced by the PCI Express fabric. A non-compliant Requester that tries to inject more isochronous transactions than what was being allowed by the contract is prevented from doing so by the flow-control mechanism thereby allowing compliant Requesters to correctly operate independent of non-compliant Requesters.

In the Endpoint-to-Root-Complex model, since the aggregated isochronous traffic is eventually limited by the host memory subsystem's bandwidth capabilities, isochronous read requests, and write requests (and Messages) are budgeted together. A Requester may divide the isochronous bandwidth between read requests and write requests as appropriate.

A.2.1. Isochronous Time Period and Isochronous Virtual Timeslot

The PCI Express isochronous time period (*T*) is uniformly divided into units of virtual timeslots (*t*). To provide precise isochronous bandwidth distribution only one isochronous request packet is allowed per virtual timeslot. The virtual timeslot supported by a PCI Express component is reported through the Reference Clock field in the Virtual Channel Capability structure or the Multi-Function Virtual Channel Capability structure. When Reference Clock = 00b, duration of a virtual timeslot *t* is 100 ns. Duration of isochronous time period *T* depends on the number of phases of the supported time-based WRR Port arbitration table size. When the time-based WRR Port Arbitration Table size equals to 128, there are 128 virtual timeslots (*t*) in an isochronous time period, i.e. T = 12.8 μs.

Note that isochronous period *T* as well as virtual timeslots *t* do not need to be aligned and synchronized among different PCI Express isochronous devices, i.e., the notion of {*T, t*} is local to each individual isochronous device.

A.2.2. Isochronous Payload Size

The payload size (Y) for isochronous transactions must not exceed Max_Payload_Size (see Section 7.8.4). After configuration, the Max_Payload_Size is set and fixed for each path that supports isochronous service with a value required to meet isochronous latency. The fixed Max_Payload_Size value is used for isochronous bandwidth budgeting regardless of the actual size of data payload associated with isochronous transactions. For isochronous bandwidth budgeting, we have

$$Y = \text{Max_Payload_Size}$$

A transaction with partial writes is treated as a normally accounted transaction. A Completer must account for partial writes as part of bandwidth assignment (for worst case servicing time).

A.2.3. Isochronous Bandwidth Allocation

Given T, t and Y, the maximum virtual timeslots within a time period is

$$N_{\max} = \frac{T}{t}$$

and the maximum specifiable isochronous bandwidth is

$$BW_{\max} = \frac{Y}{t}$$

The granularity with which isochronous bandwidth can be allocated is defined as:

$$BW_{granularity} = \frac{Y}{T}$$

Given T and t at 12.8 µs and 100 ns, respectively, N_{max} is 128. As shown in Table A-1, BW_{max} and $BW_{granularity}$ are functions of the isochronous payload size Y.

Table A-1: Isochronous Bandwidth Ranges and Granularities

Y (bytes)	128	256	512	1024
BW_{max} (MB/s)	1280	2560	5120	10240
$BW_{granularity}$ (MB/s)	10	20	40	80

Similar to bandwidth budgeting, isochronous service disciplines including arbitration schemes are based on counting requests (not the sizes of those requests. Therefore, assigning isochronous bandwidth BW_{link} to a PCI Express Link is equivalent to assigning N_{link} virtual timeslots per isochronous period, where N_{link} is given by

$$N_{link} = \frac{BW_{link}}{BW_{granularity}}$$

A Switch Port serving as an Egress Port (or an RCRB serving as a "virtual" Egress Port) for an isochronous traffic, the N_{max} virtual timeslots within T are represented by the time-based WRR Port Arbitration Table in the PCI Express Virtual Channel Capability structure detailed in Section 7.11.

The table consists of N_{max} entries. An entry in the table represents one virtual timeslot in the isochronous time period. When a table entry is given a value of PN, it means that the timeslot is assigned to an Ingress Port (in respect to the isochronous traffic targeting the Egress Port) designated by a Port Number of PN. Therefore, N_{link} virtual timeslots are assigned to the Ingress Port when there are N_{link} entries in the table with value of PN. The Egress Port may admit one isochronous request transaction from the Ingress Port for further service only when the table entry reached by the Egress Port's isochronous time ticker (that increments by 1 every *t* time and wraps around when reaching *T*) is set to PN. Even if there are outstanding isochronous requests ready in the Ingress Port, they will not be served until next round of time-based WRR arbitration. In this manner, the time-based Port Arbitration Table serves for both isochronous bandwidth assignment and isochronous traffic regulation.

For an Endpoint serving as a Requester or a Completer, isochronous bandwidth allocation is accomplished through negotiation between system software and device driver, which is outside of the scope of this specification.

A.2.4. Isochronous Transaction Latency

Transaction latency is composed of the latency through the PCI Express fabric and the latency contributed by the Completer. Isochronous transaction latency is defined for each transaction and measured in units of virtual timeslot *t*.

- The *read latency* is defined as the round-trip latency. This is the delay from the time when the device submits a memory read request packet to its Transaction Layer (Transmit side) to the time when the corresponding read completion arrives at the device's Transaction Layer (Receive side).
- The *write latency* is defined as the delay from the time when the Requester posts a memory write request to its PCI Express Transaction Layer (Transmit side) to the time when the data write becomes globally visible within the memory subsystem of the Completer. A write to memory reaches the point of global visibility when all agents accessing that memory address get the updated data.

When the upper bound and the lower bound of isochronous transaction latency are provided, the size of isochronous data buffers in a Requester can be determined. For most of common platforms, the minimum isochronous transaction latency is much smaller than the maximum. As a conservative measure, the minimum isochronous transaction latency is assumed to be zero; only guidelines on measuring the maximum isochronous transaction latency are provided here.

For a Requester, the maximum isochronous (read or write) transaction latency (L) can be accounted as the following:

$$L = L_{Fabric} + L_{Completer} ,$$

where L_{Fabric} is the maximum latency of the PCI Express fabric and $L_{Completer}$ is the maximum latency of the Completer.

L_{Fabric} which applies to both read and write transactions, depends on the topology, latency across each PCI Express Link, and the arbitration point in the path between the Requester to the Completer. The latency on a PCI Express Link depends on pipeline delays, width and operational

frequency of the Link, transmission of electrical signals across the medium, wake up latency from low power states, and delays caused by Data Link Layer Retry.

A restriction on the PCI Express topology may be imposed for each targeted platform in order to provide a practically meaningful guideline for L_{Fabric}. The values of L_{Fabric} should be reasonable and serve as practical upper limits under normal operating conditions.

The value of $L_{Completer}$ depends on the memory technology, memory configuration, and the arbitration policies in the Completer that comprehend PCI Express isochronous traffic. The target value for $L_{Completer}$ should provide enough headroom to allow for implementation tradeoffs.

Definitions of read and write transaction latencies for a Completer are different:

- Read transaction latency for the Completer is defined as the delay from the time a memory read transaction is available at the Receiver end of a PCI Express Port in the Completer to the time the corresponding read completion transaction is posted to the transmission end of the PCI Express Port.
- Write transaction latency is defined as the delay from the time a memory write transaction is available at the Receiver end of a PCI Express Port in the Completer to the time that the transmitted data is globally visible.

All of the isochronous transaction latencies defined above are based on the assumption that the Requester injects isochronous transactions uniformly. According to an isochronous contract of $\{N, T, t\}$, the uniform traffic injection is defined such that up to N transactions are evenly distributed over the isochronous period T based on a ticker granularity of virtual timeslot t. For a Requester with non-uniform isochronous transaction injection, the Requester is responsible of accounting for any additional delay due to the deviation of its injection pattern from a uniform injection pattern.

A.2.5. An Example Illustrating Isochronous Parameters

Figure A-3 illustrates the key isochronous parameters using a simplified example with $T = 20t$ and $L = 22t$. A Requester has reserved isochronous bandwidth of four transactions per T. The device shares the allocated isochronous bandwidths for both read requests and write requests. As shown, during one isochronous time period, the Requester issues two read requests and two write requests. All requests are completed within the designated transaction latency L. Also shown in the figure, there is no time dependency between the service time of write requests and the arrival time of read completions.

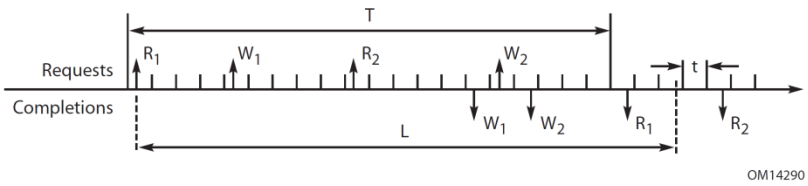


Figure A-3: A Simplified Example Illustrating PCI Express Isochronous Parameters

A.3. Isochronous Transaction Rules

Isochronous transactions follow the same rules as described in Chapter 2. In order to assist the Completer to meet latency requirements, the following additional rules further illustrate and clarify the proper behavior of isochronous transactions:

- The value in the Length field of requests must never exceed Max_Payload_Size.

A.4. Transaction Ordering

In general, isochronous transactions follow the ordering rules described in Section 2.4. The following ordering rule further illustrates and clarifies the proper behavior of isochronous transactions:

- There are no ordering guarantees between any isochronous and non-isochronous transactions because the traffic has been segregated into distinct VC resources.
- Isochronous write requests are serviced on any PCI Express Link in strictly the same order as isochronous write requests are posted.
- Switches must allow isochronous posted requests to pass isochronous read completions.

A.5. Isochronous Data Coherency

Cache coherency for isochronous transactions is an operating system software and Root Complex hardware issue. PCI Express provides the necessary mechanism to control Root Complex behavior in terms of enforcing hardware cache coherency on a per transaction basis.

For platforms where snoop latency in a Root Complex is either unbounded or can be excessively large, in order to meet tight maximum isochronous transaction latency $L_{Completer}$, or more precisely $L_{Root_Complex}$, all isochronous transactions should have the No Snoop Attribute bit set.

A Root Complex must report the Root Complex's capability to the system software by setting the Reject Snoop Transactions field in the VC Resource Capability register (for any VC resource capable of supporting isochronous traffic) in its RCRB. Based on whether or not a Root Complex is capable of providing hardware enforced cache coherency for isochronous traffic while still meeting isochronous latency target, system software can then inform the device driver of Endpoints to set or unset the No Snoop Attribute bit for isochronous transactions.

Note that cache coherency considerations for isochronous traffic do not apply to peer-to-peer communication.

A.6. Flow Control

Completers and PCI Express fabric components should implement proper sizing of buffers such that under normal operating conditions, no backpressure due to flow control should be applied to isochronous traffic injected uniformly by a Requester. For Requesters that are compliant to the isochronous contract, but have bursty injection behavior, Switches and Completers may apply flow control backpressure as long as the admitted isochronous traffic is uniform and compliant to the isochronous contract. Under abnormal conditions when isochronous traffic jitter becomes significant or when isochronous traffic is oversubscribed due to excessive Data Link Layer Retry, flow control provides a natural mechanism to ensure functional correctness.

A.7. Considerations for Bandwidth Allocation

A.7.1. Isochronous Bandwidth of PCI Express Links

Isochronous bandwidth budgeting for PCI Express Links can be derived based on Link parameters such as isochronous payload size and the speed and width of the Link.

Isochronous bandwidth allocation for a PCI Express Link should be limited to certain percentage of the maximum effective Link bandwidth in order to leave sufficient bandwidth for non-isochronous traffic and to account for temporary Link bandwidth reduction due to retries. Link utilization is counted based on the actual cycles consumed on the physical PCI Express Link. The maximum number of virtual slots allowed per Link (N_{link}) depends on the isochronous packet payload size and the speed and width of the Link.

As isochronous bandwidth allocation on a PCI Express Link is based on number of requests N_{link} per isochronous period. There is no distinction between read requests and write requests in budgeting isochronous bandwidth on a PCI Express Link.

A.7.2. Isochronous Bandwidth of Endpoints

For peer-to-peer communication, the device driver is responsible for reporting to system software if the device is capable of being a Completer for isochronous transactions. In addition, the driver must report the device's isochronous bandwidth capability. The specifics of the report mechanism are outside the scope of this specification.

A.7.3. Isochronous Bandwidth of Switches

Allocation of isochronous bandwidth for a Switch must consider the capacity and utilization of PCI Express Links associated with the Ingress Port and the Egress Port of the Switch that connect the Requester and the Completer, respectively. The lowest common denominator of the two determines if a requested isochronous bandwidth can be supported.

A.7.4. Isochronous Bandwidth of Root Complex

Isochronous bandwidth of Root Complex is reported to the software through its RCRB structure. Specifically, the Maximum Time Slots field of the VC Resource Capability register in VC Capability structure indicates the total isochronous bandwidth shared by the Root Ports associated with the RCRB. Details of the platform budgeting for available isochronous bandwidth within a Root Complex are outside of the scope of this specification.

A.8. Considerations for PCI Express Components

A.8.1. An Endpoint as a Requester

Before an Endpoint as a Requester can start issuing isochronous request transactions, the following configuration steps must be performed by software:

- Configuration of at least one VC resource capable of supporting isochronous communication and assignment of at least one TC label.
- Enablement of this VC resource.

When the Requester uniformly injects isochronous requests, the Receive Port, either a Switch Port or a Root Port, should issue Flow Control credits back promptly such that no backpressure should be applied to the associated VC. This type of Requester may size its buffer based on the PCI Express fabric latency L_{Fabric} plus the Completer's latency $L_{Completer}$.

When isochronous transactions are injected non-uniformly, either some transactions experience longer PCI Express fabric delay or the Requester gets back-pressured on the associated VC. This type of Requester must size its buffer to account for the deviation of its injection pattern from uniformity.

A.8.2. An Endpoint as a Completer

An Endpoint may serve as a Completer for isochronous peer-to-peer communication. Before an Endpoint starts serving isochronous transactions, system software must identify/configure a VC resource capable of supporting isochronous traffic and assigned a corresponding TC label.

An Endpoint Completer must observe the maximum isochronous transaction latency ($L_{Completer}$). An Endpoint Completer does not have to regulate isochronous request traffic if attached to a Switch since Switches implement traffic regulation. However, an Endpoint Completer must size its internal buffer such that no backpressure should be applied to the corresponding VC.

A.8.3. Switches

A Switch may have multiple ports capable of supporting isochronous transactions. Before a Switch starts serving isochronous transactions for a Port, the software must perform the following configuration steps:

- ❑ Configuration/enablement of at least one VC resource capable of supporting isochronous communication.
- ❑ Configuration of the Port as an Ingress Port:
  - Configuration (or reconfiguration if the associated VC of the Egress Port is already enabled) of the time-based WRR Port Arbitration Table of the targeting Egress Port to include N_{link} entries set to the Ingress Port's Port Number. Here N_{link} is the isochronous allocation for the Ingress Port.
  - Enabling the targeting Egress Port to load newly programmed Port Arbitration Table.
- ❑ Configuration of the Port as an Egress Port:
  - Configuration of each VC's Port Arbitration Table with number of entries set according to the assigned isochronous bandwidth for all Ingress Ports.
  - Select proper VC Arbitration, e.g., as strict-priority based VC Arbitration.
  - If required, configuration of the Port's VC Arbitration Table with large weights assigned accordingly to each associated VC.

Each VC associated with isochronous traffic may be served as the highest priority in arbitrating for the shared PCI Express Link resource at an Egress Port. This is comprehended by a Switch's internal arbitration scheme.

In addition, a Switch Port may use "just in time" scheduling mechanism to reduce VC arbitration latency. Instead of pipelining non-isochronous Transport Layer packets to the Data Link Layer of the Egress Port in a manner that Data Link Layer transmit buffer becomes saturated, the Switch Port may hold off scheduling of a new non-isochronous packet to the Data Link Layer as long as it is possible without incurring unnecessary Link idle time.

When a VC configured to support isochronous traffic is enabled for a Switch Port (ingress) that is connected to a Requester, the Switch must enforce proper traffic regulation to ensure that isochronous traffic from the Port conforms to this specification. With such enforcement, normal isochronous transactions from compliant Requesters will not be impacted by ill behavior of any non-compliant Requester.

The above isochronous traffic regulation mechanism only applies to request transactions but not to completion transactions. When Endpoint-to-Root-Complex and peer-to-peer communications co-exist in a Switch, an Egress Port may mix isochronous write requests and read completions in the same direction. In the case of contention, the Egress Port must allow write requests to pass read completions to ensure the Switch meets latency requirement for isochronous requests.

A.8.4. Root Complex

A Root Complex may have multiple Root Ports capable of supporting isochronous transactions. Before a Root Complex starts serving isochronous transactions for a Root Port, the Port must be configured by software to enable VC to support isochronous traffic using the following configuration steps:

- ❑ Configuration of at least one VC resource capable of supporting isochronous communication and assignment of at least one TC label.
- ❑ Configuration of the Root Port as an Ingress Port:
  - Configuration (or reconfiguration if the associated VC in RCRB is already enabled) of the time-based WRR Port Arbitration Table of the targeting RCRB to include N_{link} entries set to the Ingress Port's Port Number. Here N_{link} is the isochronous allocation for the Port.
  - Enabling the targeting RCRB to load newly programmed Port Arbitration Table.
- ❑ Configuration of the Root Port as an Egress Port:
  - If supported, configuration of the Root Port's VC Arbitration Table with large weights assigned to the associated VC.
  - If the Root Complex supports peer-to-peer traffic between Root Ports, configuration of the Root Port's Port Arbitration Table number of entries is set according to the assigned isochronous bandwidth for all Ingress Ports.

A Root Complex must observe the maximum isochronous transaction latency ($L_{Completer}$ or more precisely $L_{Root_Complex}$) that applies to all the Root Ports in the Root Complex. How a Root Complex schedules memory cycles for PCI Express isochronous transactions and other memory transactions is outside of the scope of this specification as long as $L_{Root_Complex}$ is met for PCI Express isochronous transactions.

When a VC is enabled to support isochronous traffic for a Root Port, the Root Complex must enforce proper traffic regulation to ensure that isochronous traffic from the Root Port conforms to this specification. With such enforcement, normal isochronous transactions from compliant Requesters will not be impacted by ill behavior of any non-compliant Requesters. Isochronous traffic regulation is implemented using the time-based Port Arbitration Table in RCRB.

Root Complex may perform the following operations for invalid isochronous transactions:

- ❑ Return partial completions for read requests with the value in the Length field exceeding Max_Payload_Size.

B

B. Symbol Encoding

Table B-1 shows the byte-to-Symbol encodings for data characters. Table B-2 shows the Symbol encodings for the Special Symbols used for TLP/DLLP Framing and for interface management. RD- and RD+ refer to the Running Disparity of the Symbol sequence on a per-Lane basis.

Table B-1: 8b/10b Data Symbol Codes

Data Byte Name	Data Byte Value (hex)	Bits HGF EDCBA (binary)	Current RD- abcdei fghj(binary)	Current RD+ abcdei fghj (binary)
D0.0	00	000 00000	100111 0100	011000 1011
D1.0	01	000 00001	011101 0100	100010 1011
D2.0	02	000 00010	101101 0100	010010 1011
D3.0	03	000 00011	110001 1011	110001 0100
D4.0	04	000 00100	110101 0100	001010 1011
D5.0	05	000 00101	101001 1011	101001 0100
D6.0	06	000 00110	011001 1011	011001 0100
D7.0	07	000 00111	111000 1011	000111 0100
D8.0	08	000 01000	111001 0100	000110 1011
D9.0	09	000 01001	100101 1011	100101 0100
D10.0	0A	000 01010	010101 1011	010101 0100
D11.0	0B	000 01011	110100 1011	110100 0100
D12.0	0C	000 01100	001101 1011	001101 0100
D13.0	0D	000 01101	101100 1011	101100 0100
D14.0	0E	000 01110	011100 1011	011100 0100
D15.0	0F	000 01111	010111 0100	101000 1011
D16.0	10	000 10000	011011 0100	100100 1011
D17.0	11	000 10001	100011 1011	100011 0100
D18.0	12	000 10010	010011 1011	010011 0100
D19.0	13	000 10011	110010 1011	110010 0100
D20.0	14	000 10100	001011 1011	001011 0100
D21.0	15	000 10101	101010 1011	101010 0100
D22.0	16	000 10110	011010 1011	011010 0100

Data Byte Name	Data Byte Value (hex)	Bits HGF EDCBA (binary)	Current RD- abcdei fghj(binary)	Current RD+ abcdei fghj (binary)
D23.0	17	000 10111	111010 0100	000101 1011
D24.0	18	000 11000	110011 0100	001100 1011
D25.0	19	000 11001	100110 1011	100110 0100
D26.0	1A	000 11010	010110 1011	010110 0100
D27.0	1B	000 11011	110110 0100	001001 1011
D28.0	1C	000 11100	001110 1011	001110 0100
D29.0	1D	000 11101	101110 0100	010001 1011
D30.0	1E	000 11110	011110 0100	100001 1011
D31.0	1F	000 11111	101011 0100	010100 1011
D0.1	20	001 00000	100111 1001	011000 1001
D1.1	21	001 00001	011101 1001	100010 1001
D2.1	22	001 00010	101101 1001	010010 1001
D3.1	23	001 00011	110001 1001	110001 1001
D4.1	24	001 00100	110101 1001	001010 1001
D5.1	25	001 00101	101001 1001	101001 1001
D6.1	26	001 00110	011001 1001	011001 1001
D7.1	27	001 00111	111000 1001	000111 1001
D8.1	28	001 01000	111001 1001	000110 1001
D9.1	29	001 01001	100101 1001	100101 1001
D10.1	2A	001 01010	010101 1001	010101 1001
D11.1	2B	001 01011	110100 1001	110100 1001
D12.1	2C	001 01100	001101 1001	001101 1001
D13.1	2D	001 01101	101100 1001	101100 1001
D14.1	2E	001 01110	011100 1001	011100 1001
D15.1	2F	001 01111	010111 1001	101000 1001
D16.1	30	001 10000	011011 1001	100100 1001
D17.1	31	001 10001	100011 1001	100011 1001
D18.1	32	001 10010	010011 1001	010011 1001
D19.1	33	001 10011	110010 1001	110010 1001
D20.1	34	001 10100	001011 1001	001011 1001
D21.1	35	001 10101	101010 1001	101010 1001
D22.1	36	001 10110	011010 1001	011010 1001
D23.1	37	001 10111	111010 1001	000101 1001

Data Byte Name	Data Byte Value (hex)	Bits HGF EDCBA (binary)	Current RD- abcdei fghj(binary)	Current RD+ abcdei fghj (binary)
D24.1	38	001 11000	110011 1001	001100 1001
D25.1	39	001 11001	100110 1001	100110 1001
D26.1	3A	001 11010	010110 1001	010110 1001
D27.1	3B	001 11011	110110 1001	001001 1001
D28.1	3C	001 11100	001110 1001	001110 1001
D29.1	3D	001 11101	101110 1001	010001 1001
D30.1	3E	001 11110	011110 1001	100001 1001
D31.1	3F	001 11111	101011 1001	010100 1001
D0.2	40	010 00000	100111 0101	011000 0101
D1.2	41	010 00001	011101 0101	100010 0101
D2.2	42	010 00010	101101 0101	010010 0101
D3.2	43	010 00011	110001 0101	110001 0101
D4.2	44	010 00100	110101 0101	001010 0101
D5.2	45	010 00101	101001 0101	101001 0101
D6.2	46	010 00110	011001 0101	011001 0101
D7.2	47	010 00111	111000 0101	000111 0101
D8.2	48	010 01000	111001 0101	000110 0101
D9.2	49	010 01001	100101 0101	100101 0101
D10.2	4A	010 01010	010101 0101	010101 0101
D11.2	4B	010 01011	110100 0101	110100 0101
D12.2	4C	010 01100	001101 0101	001101 0101
D13.2	4D	010 01101	101100 0101	101100 0101
D14.2	4E	010 01110	011100 0101	011100 0101
D15.2	4F	010 01111	010111 0101	101000 0101
D16.2	50	010 10000	011011 0101	100100 0101
D17.2	51	010 10001	100011 0101	100011 0101
D18.2	52	010 10010	010011 0101	010011 0101
D19.2	53	010 10011	110010 0101	110010 0101
D20.2	54	010 10100	001011 0101	001011 0101
D21.2	55	010 10101	101010 0101	101010 0101
D22.2	56	010 10110	011010 0101	011010 0101
D23.2	57	010 10111	111010 0101	000101 0101
D24.2	58	010 11000	110011 0101	001100 0101

Data Byte Name	Data Byte Value (hex)	Bits HGF EDCBA (binary)	Current RD- abcdei fghj(binary)	Current RD+ abcdei fghj (binary)
D25.2	59	010 11001	100110 0101	100110 0101
D26.2	5A	010 11010	010110 0101	010110 0101
D27.2	5B	010 11011	110110 0101	001001 0101
D28.2	5C	010 11100	001110 0101	001110 0101
D29.2	5D	010 11101	101110 0101	010001 0101
D30.2	5E	010 11110	011110 0101	100001 0101
D31.2	5F	010 11111	101011 0101	010100 0101
D0.3	60	011 00000	100111 0011	011000 1100
D1.3	61	011 00001	011101 0011	100010 1100
D2.3	62	011 00010	101101 0011	010010 1100
D3.3	63	011 00011	110001 1100	110001 0011
D4.3	64	011 00100	110101 0011	001010 1100
D5.3	65	011 00101	101001 1100	101001 0011
D6.3	66	011 00110	011001 1100	011001 0011
D7.3	67	011 00111	111000 1100	000111 0011
D8.3	68	011 01000	111001 0011	000110 1100
D9.3	69	011 01001	100101 1100	100101 0011
D10.3	6A	011 01010	010101 1100	010101 0011
D11.3	6B	011 01011	110100 1100	110100 0011
D12.3	6C	011 01100	001101 1100	001101 0011
D13.3	6D	011 01101	101100 1100	101100 0011
D14.3	6E	011 01110	011100 1100	011100 0011
D15.3	6F	011 01111	010111 0011	101000 1100
D16.3	70	011 10000	011011 0011	100100 1100
D17.3	71	011 10001	100011 1100	100011 0011
D18.3	72	011 10010	010011 1100	010011 0011
D19.3	73	011 10011	110010 1100	110010 0011
D20.3	74	011 10100	001011 1100	001011 0011
D21.3	75	011 10101	101010 1100	101010 0011
D22.3	76	011 10110	011010 1100	011010 0011
D23.3	77	011 10111	111010 0011	000101 1100
D24.3	78	011 11000	110011 0011	001100 1100
D25.3	79	011 11001	100110 1100	100110 0011

Data Byte Name	Data Byte Value (hex)	Bits HGF EDCBA (binary)	Current RD- abcdei fghj(binary)	Current RD+ abcdei fghj (binary)
D26.3	7A	011 11010	010110 1100	010110 0011
D27.3	7B	011 11011	110110 0011	001001 1100
D28.3	7C	011 11100	001110 1100	001110 0011
D29.3	7D	011 11101	101110 0011	010001 1100
D30.3	7E	011 11110	011110 0011	100001 1100
D31.3	7F	011 11111	101011 0011	010100 1100
D0.4	80	100 00000	100111 0010	011000 1101
D1.4	81	100 00001	011101 0010	100010 1101
D2.4	82	100 00010	101101 0010	010010 1101
D3.4	83	100 00011	110001 1101	110001 0010
D4.4	84	100 00100	110101 0010	001010 1101
D5.4	85	100 00101	101001 1101	101001 0010
D6.4	86	100 00110	011001 1101	011001 0010
D7.4	87	100 00111	111000 1101	000111 0010
D8.4	88	100 01000	111001 0010	000110 1101
D9.4	89	100 01001	100101 1101	100101 0010
D10.4	8A	100 01010	010101 1101	010101 0010
D11.4	8B	100 01011	110100 1101	110100 0010
D12.4	8C	100 01100	001101 1101	001101 0010
D13.4	8D	100 01101	101100 1101	101100 0010
D14.4	8E	100 01110	011100 1101	011100 0010
D15.4	8F	100 01111	010111 0010	101000 1101
D16.4	90	100 10000	011011 0010	100100 1101
D17.4	91	100 10001	100011 1101	100011 0010
D18.4	92	100 10010	010011 1101	010011 0010
D19.4	93	100 10011	110010 1101	110010 0010
D20.4	94	100 10100	001011 1101	001011 0010
D21.4	95	100 10101	101010 1101	101010 0010
D22.4	96	100 10110	011010 1101	011010 0010
D23.4	97	100 10111	111010 0010	000101 1101
D24.4	98	100 11000	110011 0010	001100 1101
D25.4	99	100 11001	100110 1101	100110 0010
D26.4	9A	100 11010	010110 1101	010110 0010

Data Byte Name	Data Byte Value (hex)	Bits HGF EDCBA (binary)	Current RD- abcdei fghj(binary)	Current RD+ abcdei fghj (binary)
D27.4	9B	100 11011	110110 0010	001001 1101
D28.4	9C	100 11100	001110 1101	001110 0010
D29.4	9D	100 11101	101110 0010	010001 1101
D30.4	9E	100 11110	011110 0010	100001 1101
D31.4	9F	100 11111	101011 0010	010100 1101
D0.5	A0	101 00000	100111 1010	011000 1010
D1.5	A1	101 00001	011101 1010	100010 1010
D2.5	A2	101 00010	101101 1010	010010 1010
D3.5	A3	101 00011	110001 1010	110001 1010
D4.5	A4	101 00100	110101 1010	001010 1010
D5.5	A5	101 00101	101001 1010	101001 1010
D6.5	A6	101 00110	011001 1010	011001 1010
D7.5	A7	101 00111	111000 1010	000111 1010
D8.5	A8	101 01000	111001 1010	000110 1010
D9.5	A9	101 01001	100101 1010	100101 1010
D10.5	AA	101 01010	010101 1010	010101 1010
D11.5	AB	101 01011	110100 1010	110100 1010
D12.5	AC	101 01100	001101 1010	001101 1010
D13.5	AD	101 01101	101100 1010	101100 1010
D14.5	AE	101 01110	011100 1010	011100 1010
D15.5	AF	101 01111	010111 1010	101000 1010
D16.5	B0	101 10000	011011 1010	100100 1010
D17.5	B1	101 10001	100011 1010	100011 1010
D18.5	B2	101 10010	010011 1010	010011 1010
D19.5	B3	101 10011	110010 1010	110010 1010
D20.5	B4	101 10100	001011 1010	001011 1010
D21.5	B5	101 10101	101010 1010	101010 1010
D22.5	B6	101 10110	011010 1010	011010 1010
D23.5	B7	101 10111	111010 1010	000101 1010
D24.5	B8	101 11000	110011 1010	001100 1010
D25.5	B9	101 11001	100110 1010	100110 1010
D26.5	BA	101 11010	010110 1010	010110 1010
D27.5	BB	101 11011	110110 1010	001001 1010

Data Byte Name	Data Byte Value (hex)	Bits HGF EDCBA (binary)	Current RD- abcdei fghj(binary)	Current RD+ abcdei fghj (binary)
D28.5	BC	101 11100	001110 1010	001110 1010
D29.5	BD	101 11101	101110 1010	010001 1010
D30.5	BE	101 11110	011110 1010	100001 1010
D31.5	BF	101 11111	101011 1010	010100 1010
D0.6	C0	110 00000	100111 0110	011000 0110
D1.6	C1	110 00001	011101 0110	100010 0110
D2.6	C2	110 00010	101101 0110	010010 0110
D3.6	C3	110 00011	110001 0110	110001 0110
D4.6	C4	110 00100	110101 0110	001010 0110
D5.6	C5	110 00101	101001 0110	101001 0110
D6.6	C6	110 00110	011001 0110	011001 0110
D7.6	C7	110 00111	111000 0110	000111 0110
D8.6	C8	110 01000	111001 0110	000110 0110
D9.6	C9	110 01001	100101 0110	100101 0110
D10.6	CA	110 01010	010101 0110	010101 0110
D11.6	CB	110 01011	110100 0110	110100 0110
D12.6	CC	110 01100	001101 0110	001101 0110
D13.6	CD	110 01101	101100 0110	101100 0110
D14.6	CE	110 01110	011100 0110	011100 0110
D15.6	CF	110 01111	010111 0110	101000 0110
D16.6	D0	110 10000	011011 0110	100100 0110
D17.6	D1	110 10001	100011 0110	100011 0110
D18.6	D2	110 10010	010011 0110	010011 0110
D19.6	D3	110 10011	110010 0110	110010 0110
D20.6	D4	110 10100	001011 0110	001011 0110
D21.6	D5	110 10101	101010 0110	101010 0110
D22.6	D6	110 10110	011010 0110	011010 0110
D23.6	D7	110 10111	111010 0110	000101 0110
D24.6	D8	110 11000	110011 0110	001100 0110
D25.6	D9	110 11001	100110 0110	100110 0110
D26.6	DA	110 11010	010110 0110	010110 0110
D27.6	DB	110 11011	110110 0110	001001 0110
D28.6	DC	110 11100	001110 0110	001110 0110

Data Byte Name	Data Byte Value (hex)	Bits HGF EDCBA (binary)	Current RD- abcdei fghj(binary)	Current RD+ abcdei fghj (binary)
D29.6	DD	110 11101	101110 0110	010001 0110
D30.6	DE	110 11110	011110 0110	100001 0110
D31.6	DF	110 11111	101011 0110	010100 0110
D0.7	E0	111 00000	100111 0001	011000 1110
D1.7	E1	111 00001	011101 0001	100010 1110
D2.7	E2	111 00010	101101 0001	010010 1110
D3.7	E3	111 00011	110001 1110	110001 0001
D4.7	E4	111 00100	110101 0001	001010 1110
D5.7	E5	111 00101	101001 1110	101001 0001
D6.7	E6	111 00110	011001 1110	011001 0001
D7.7	E7	111 00111	111000 1110	000111 0001
D8.7	E8	111 01000	111001 0001	000110 1110
D9.7	E9	111 01001	100101 1110	100101 0001
D10.7	EA	111 01010	010101 1110	010101 0001
D11.7	EB	111 01011	110100 1110	110100 1000
D12.7	EC	111 01100	001101 1110	001101 0001
D13.7	ED	111 01101	101100 1110	101100 1000
D14.7	EE	111 01110	011100 1110	011100 1000
D15.7	EF	111 01111	010111 0001	101000 1110
D16.7	F0	111 10000	011011 0001	100100 1110
D17.7	F1	111 10001	100011 0111	100011 0001
D18.7	F2	111 10010	010011 0111	010011 0001
D19.7	F3	111 10011	110010 1110	110010 0001
D20.7	F4	111 10100	001011 0111	001011 0001
D21.7	F5	111 10101	101010 1110	101010 0001
D22.7	F6	111 10110	011010 1110	011010 0001
D23.7	F7	111 10111	111010 0001	000101 1110
D24.7	F8	111 11000	110011 0001	001100 1110
D25.7	F9	111 11001	100110 1110	100110 0001
D26.7	FA	111 11010	010110 1110	010110 0001
D27.7	FB	111 11011	110110 0001	001001 1110
D28.7	FC	111 11100	001110 1110	001110 0001
D29.7	FD	111 11101	101110 0001	010001 1110

Data Byte Name	Data Byte Value (hex)	Bits HGF EDCBA (binary)	Current RD- abcdei fghj(binary)	Current RD+ abcdei fghj (binary)
D30.7	FE	111 11110	011110 0001	100001 1110
D31.7	FF	111 11111	101011 0001	010100 1110

Table B-2: 8b/10b Special Character Symbol Codes

Data Byte Name	Data Byte Value	Bits HGF EDCBA	Current RD - abcdei fghj	Current RD + abcdei fghj
K28.0	1C	000 11100	001111 0100	110000 1011
K28.1	3C	001 11100	001111 1001	110000 0110
K28.2	5C	010 11100	001111 0101	110000 1010
K28.3	7C	011 11100	001111 0011	110000 1100
K28.4	9C	100 11100	001111 0010	110000 1101
K28.5	BC	101 11100	001111 1010	110000 0101
K28.6	DC	110 11100	001111 0110	110000 1001
K28.7	FC	111 11100	001111 1000	110000 0111
K23.7	F7	111 10111	111010 1000	000101 0111
K27.7	FB	111 11011	110110 1000	001001 0111
K29.7	FD	111 11101	101110 1000	010001 0111
K30.7	FE	111 11110	011110 1000	100001 0111

C

C. Physical Layer Appendix

C.1. 8b/10b Data Scrambling Example

The following subroutines encode and decode an 8-bit value contained in "inbyte" with the LFSR. This is presented as one example only; there are many ways to obtain the proper output. This example demonstrates how to advance the LFSR eight times in one operation and how to XOR the data in one operation. Many other implementations are possible but they must all produce the same output as that shown here.

The following algorithm uses the "C" programming language conventions, where "<<" and ">>" represent the shift left and shift right operators, ">" is the compare greater than operator, and "^" is the exclusive or operator, and "&" is the logical "AND" operator.

```
/*
   this routine implements the serial descrambling algorithm in parallel form
   for the LSFR polynomial:  x^16+x^5+x^4+x^3+1
   this advances the LSFR 8 bits every time it is called
   this requires fewer than 25 xor gates to implement (with a static register)

   The XOR required to advance 8 bits/clock is:
bit   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15

      8   9  10  11  12  13  14  15   0   1   2   3   4   5   6   7
                  8   9  10  11  12  13  14  15
                      8   9  10  11  12  13  14  15
                          8   9  10  11  12  13  14  15

   The serial data is just the reverse of the upper byte:
bit   0   1   2   3   4   5   6   7
     15  14  13  12  11  10   9   8
*/

int scramble_byte(int inbyte)
{

static int scrambit[16];
static int bit[16];
```

```
static int bit_out[16];
static unsigned short lfsr = 0xffff; // 16 bit short for polynomial
int i, outbyte;

    if (inbyte == COMMA)   // if this is a comma
    {
        lfsr = 0xffff;     // reset the LFSR
        return (COMMA);    // and return the same data
    }


    if (inbyte == SKIP)    // don't advance or encode on skip
        return (SKIP);

    for (i=0; i<16;i++)    // convert LFSR to bit array for legibility
        bit[i] = (lfsr >> i) & 1;

    for (i=0; i<8; i++)    // convert byte to be scrambled for legibility
        scrambit[i] = (inbyte >> i) & 1;


    // apply the xor to the data
    if (! (inbyte & 0x100) &&    // if not a KCODE, scramble the data
        ! (TrainingSequence == TRUE))  // and if not in the middle of
    {                                  // a training sequence
        scrambit[0] ^= bit[15];
        scrambit[1] ^= bit[14];
        scrambit[2] ^= bit[13];
        scrambit[3] ^= bit[12];
        scrambit[4] ^= bit[11];
        scrambit[5] ^= bit[10];
        scrambit[6] ^= bit[9];
        scrambit[7] ^= bit[8];
    }

    // Now advance the LFSR 8 serial clocks
    bit_out[ 0] = bit[ 8];
    bit_out[ 1] = bit[ 9];
    bit_out[ 2] = bit[10];
    bit_out[ 3] = bit[11] ^ bit[ 8];
    bit_out[ 4] = bit[12] ^ bit[ 9] ^ bit[ 8];
    bit_out[ 5] = bit[13] ^ bit[10] ^ bit[ 9] ^ bit[ 8];
    bit_out[ 6] = bit[14] ^ bit[11] ^ bit[10] ^ bit[ 9];
```

```
    bit_out[ 7] = bit[15] ^ bit[12] ^ bit[11] ^ bit[10];
    bit_out[ 8] = bit[ 0] ^ bit[13] ^ bit[12] ^ bit[11];
    bit_out[ 9] = bit[ 1] ^ bit[14] ^ bit[13] ^ bit[12];
    bit_out[10] = bit[ 2] ^ bit[15] ^ bit[14] ^ bit[13];
    bit_out[11] = bit[ 3]           ^ bit[15] ^ bit[14];
    bit_out[12] = bit[ 4]                     ^ bit[15];
    bit_out[13] = bit[ 5];
    bit_out[14] = bit[ 6];
    bit_out[15] = bit[ 7];
    lfsr = 0;
    for (i=0; i <16; i++) // convert the LFSR back to an integer
        lfsr += (bit_out[i] << i);

    outbyte = 0;
    for (i= 0; i<8; i++) // convert data back to an integer
        outbyte += (scrambit[i] << i);

    return outbyte;
}

/*  NOTE THAT THE DESCRAMBLE ROUTINE IS IDENTICAL TO THE SCRAMBLE ROUTINE
    this routine implements the serial descrambling algorithm in parallel form
    this advances the lfsr 8 bits every time it is called
    this uses fewer than 25 xor gates to implement (with a static register)
    The XOR tree is the same as the scrambling routine
*/

int unscramble_byte(int inbyte)
{

    static int descrambit[8];
    static int bit[16];
    static int bit_out[16];
    static unsigned short lfsr = 0xffff;  // 16 bit short for polynomial
    int outbyte, i;
```

```
if (inbyte == COMMA)   // if this is a comma
{
   lfsr = 0xffff;      // reset the LFSR
   return (COMMA);     // and return the same data
}

if (inbyte == SKIP)   // don't advance or encode on skip
    return (SKIP);

for (i=0; i<16;i++)   // convert the LFSR to bit array for legibility
    bit[i] = (lfsr >> i) & 1;

for (i=0; i<8; i++)   // convert byte to be de-scrambled for legibility
    descrambit[i] = (inbyte >> i) & 1;

// apply the xor to the data
if (! (inbyte & 0x100) &&    // if not a KCODE, scramble the data
    ! (TrainingSequence == TRUE))  // and if not in the middle of
{                                  // a training sequence
    descrambit[0] ^= bit[15];
    descrambit[1] ^= bit[14];
    descrambit[2] ^= bit[13];
    descrambit[3] ^= bit[12];
    descrambit[4] ^= bit[11];
    descrambit[5] ^= bit[10];
    descrambit[6] ^= bit[9];
    descrambit[7] ^= bit[8];
}

// Now advance the LFSR 8 serial clocks
bit_out[ 0] = bit[ 8];
bit_out[ 1] = bit[ 9];
bit_out[ 2] = bit[10];
bit_out[ 3] = bit[11] ^ bit[ 8];
bit_out[ 4] = bit[12] ^ bit[ 9] ^ bit[ 8];
bit_out[ 5] = bit[13] ^ bit[10] ^ bit[ 9] ^ bit[ 8];
bit_out[ 6] = bit[14] ^ bit[11] ^ bit[10] ^ bit[ 9];
bit_out[ 7] = bit[15] ^ bit[12] ^ bit[11] ^ bit[10];
bit_out[ 8] = bit[ 0] ^ bit[13] ^ bit[12] ^ bit[11];
bit_out[ 9] = bit[ 1] ^ bit[14] ^ bit[13] ^ bit[12];
bit_out[10] = bit[ 2] ^ bit[15] ^ bit[14] ^ bit[13];
bit_out[11] = bit[ 3]           ^ bit[15] ^ bit[14];
```

```
    bit_out[12] = bit[ 4]                     ^ bit[15];
    bit_out[13] = bit[ 5];
    bit_out[14] = bit[ 6];
    bit_out[15] = bit[ 7];
    lfsr = 0;
    for (i=0; i <16; i++) // convert the LFSR back to an integer
        lfsr += (bit_out[i] << i);

    outbyte = 0;
    for (i= 0; i<8; i++) // convert data back to an integer
        outbyte += (descrambit[i] << i);

    return outbyte;
}
```

The initial 16-bit values of the LFSR for the first 128 LFSR advances following a reset are listed below:

	0, 8	**1, 9**	**2, A**	**3, B**	**4, C**	**5, D**	**6, E**	**7, F**
00	FFFF	E817	0328	284B	4DE8	E755	404F	4140
08	4E79	761E	1466	6574	7DBD	B6E5	FDA6	B165
10	7D09	02E5	E572	673D	34CF	CB54	4743	4DEF
18	E055	40E0	EE40	54BE	B334	2C7B	7D0C	07E5
20	E5AF	BA3D	248A	8DC4	D995	85A1	BD5D	4425
28	2BA4	A2A3	B8D2	CBF8	EB43	5763	6E7F	773E
30	345F	5B54	5853	5F18	14B7	B474	6CD4	DC4C
38	5C7C	70FC	F6F0	E6E6	F376	603B	3260	64C2
40	CB84	9743	5CBF	B3FC	E47B	6E04	0C3E	3F2C
48	29D7	D1D1	C069	7BC0	CB73	6043	4A60	6FFA
50	F207	1102	01A9	A939	2351	566B	6646	4FF6
58	F927	3081	85B0	AC5D	478C	82EF	F3F2	E43B
60	2E04	027E	7E72	79AE	A501	1A7D	7F2A	2197
68	9019	0610	1096	9590	8FCD	D0E7	F650	46E6
70	E8D6	C228	3AB2	B70A	129F	9CE2	FC3C	2B5C
78	5AA3	AF6A	70C7	CDF0	E3D5	C0AB	B9C0	D9C1

An 8-bit value of 0 repeatedly encoded with the LFSR after reset produces the following consecutive 8-bit values:

	00	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**0A**	**0B**	**0C**	**0D**	**0E**	**0F**
00	FF	17	C0	14	B2	E7	02	82	72	6E	28	A6	BE	6D	BF	8D
10	BE	40	A7	E6	2C	D3	E2	B2	07	02	77	2A	CD	34	BE	E0
20	A7	5D	24	B1	9B	A1	BD	22	D4	45	1D	D3	D7	EA	76	EE
30	2C	DA	1A	FA	28	2D	36	3B	3A	0E	6F	67	CF	06	4C	26
40	D3	E9	3A	CD	27	76	30	FC	94	8B	03	DE	D3	06	52	F6
50	4F	88	80	95	C4	6A	66	F2	9F	0C	A1	35	E2	41	CF	27
60	74	40	7E	9E	A5	58	FE	84	09	60	08	A9	F1	0B	6F	62
70	17	43	5C	ED	48	39	3F	D4	5A	F5	0E	B3	C7	03	9D	9B
80	8B	0D	8E	5C	33	98	77	AE	2D	AC	0B	3E	DA	0B	42	7A
90	7C	D1	CF	A8	1C	12	EE	41	C2	3F	38	7A	0D	69	F4	01
A0	DA	31	72	C5	A0	D7	93	0E	DC	AF	A4	55	E7	F0	72	16
B0	68	D5	38	84	DD	00	CD	18	9E	CA	30	59	4C	75	1B	77
C0	31	C5	ED	CF	91	64	6E	3D	FE	E8	29	04	CF	6C	FC	C4
D0	0B	5E	DA	62	BA	5B	AB	DF	59	B7	7D	37	5E	E3	1A	C6
E0	88	14	F5	4F	8B	C8	56	CB	D3	10	42	63	04	8A	B4	F7
F0	84	01	A0	01	83	49	67	EE	3E	2A	8B	A4	76	AF	14	D5
100	4F	AC	60	B6	79	D6	62	B7	43	E7	E5	2A	40	2C	6E	7A
110	56	61	63	20	6A	97	4A	38	05	E5	DD	68	0D	78	4C	53
120	8B	D6	86	57	B2	AA	1A	80	18	DC	BA	FC	03	A3	4B	30

At 2.5 GT/s, scrambling produces the power spectrum (in the 10-bit domain) shown in Figure C-1.

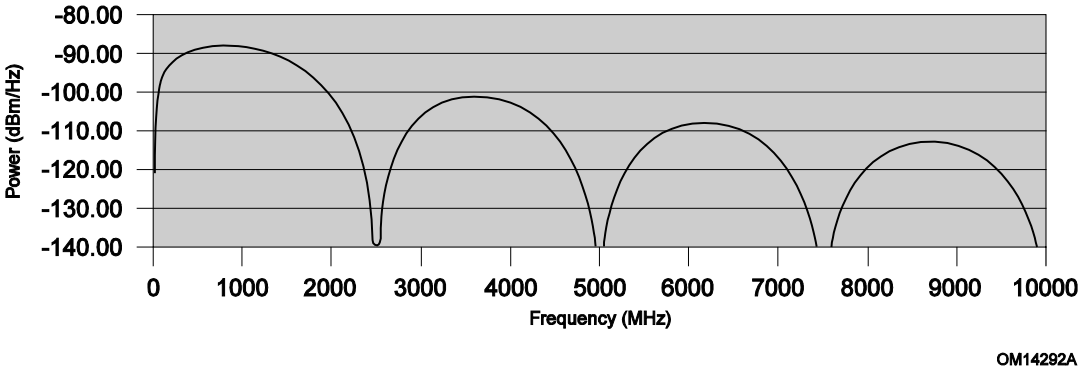


Figure C-1: Scrambling Spectrum at 2.5 GT/s for Data Value of 0

C.2. 128b/130b Data Scrambling Example

The following subroutines illustrate how calculate the next value of the 128b/130b scrambling LFSR and how to scramble (or descramble) an 8-bit value, both given the current value of the LFSR.

This is presented as one example only; there are many ways to obtain the proper output. This example demonstrates how to advance the LFSR eight times in one operation and how to XOR the data in one operation. Many other implementations are possible but they must all produce the same output as that shown here.

The following algorithm uses the "C" programming language conventions, where "<<" and ">>" represent the shift left and shift right operators, "^" is the exclusive or operator, and "|=" is the assignment by bitwise or operator.

```
#include <stdio.h>

// "Set Bit" - Sets bit number "bit" of "var" to the value "val".  Bit "bit" of "var" must start cleared.
#define SB(var,bit,val) var |= (val & 1) << bit

// "Get Bit" - Returns the value of bit number "bit" of "var".
#define GB(var,bit) ((var >> bit) & 1)

// Function to advance the LFSR value by 8 bits, given the current LFSR value
unsigned long int calc_next_lfsr(unsigned long int lfsr) {
  unsigned long int next_lfsr = 0;
  SB(next_lfsr,22, GB(lfsr,14) ^ GB(lfsr,16) ^ GB(lfsr,18) ^ GB(lfsr,20) ^ GB(lfsr,21) ^ GB(lfsr,22));
  SB(next_lfsr,21, GB(lfsr,13) ^ GB(lfsr,15) ^ GB(lfsr,17) ^ GB(lfsr,19) ^ GB(lfsr,20) ^ GB(lfsr,21));
  SB(next_lfsr,20, GB(lfsr,12) ^ GB(lfsr,19) ^ GB(lfsr,21));
  SB(next_lfsr,19, GB(lfsr,11) ^ GB(lfsr,18) ^ GB(lfsr,20) ^ GB(lfsr,22));
  SB(next_lfsr,18, GB(lfsr,10) ^ GB(lfsr,17) ^ GB(lfsr,19) ^ GB(lfsr,21));
  SB(next_lfsr,17, GB(lfsr, 9) ^ GB(lfsr,16) ^ GB(lfsr,18) ^ GB(lfsr,20) ^ GB(lfsr,22));
  SB(next_lfsr,16, GB(lfsr, 8) ^ GB(lfsr,15) ^ GB(lfsr,17) ^ GB(lfsr,19) ^ GB(lfsr,21) ^ GB(lfsr,22));
  SB(next_lfsr,15, GB(lfsr, 7) ^ GB(lfsr,22));
  SB(next_lfsr,14, GB(lfsr, 6) ^ GB(lfsr,21));
  SB(next_lfsr,13, GB(lfsr, 5) ^ GB(lfsr,20) ^ GB(lfsr,22));
  SB(next_lfsr,12, GB(lfsr, 4) ^ GB(lfsr,19) ^ GB(lfsr,21) ^ GB(lfsr,22));
  SB(next_lfsr,11, GB(lfsr, 3) ^ GB(lfsr,18) ^ GB(lfsr,20) ^ GB(lfsr,21) ^ GB(lfsr,22));
  SB(next_lfsr,10, GB(lfsr, 2) ^ GB(lfsr,17) ^ GB(lfsr,19) ^ GB(lfsr,20) ^ GB(lfsr,21) ^ GB(lfsr,22));
  SB(next_lfsr, 9, GB(lfsr, 1) ^ GB(lfsr,16) ^ GB(lfsr,18) ^ GB(lfsr,19) ^ GB(lfsr,20) ^ GB(lfsr,21));
  SB(next_lfsr, 8, GB(lfsr, 0) ^ GB(lfsr,15) ^ GB(lfsr,17) ^ GB(lfsr,18) ^ GB(lfsr,19) ^ GB(lfsr,20));
  SB(next_lfsr, 7, GB(lfsr,17) ^ GB(lfsr,19) ^ GB(lfsr,20) ^ GB(lfsr,21));
  SB(next_lfsr, 6, GB(lfsr,16) ^ GB(lfsr,18) ^ GB(lfsr,19) ^ GB(lfsr,20) ^ GB(lfsr,22));
  SB(next_lfsr, 5, GB(lfsr,15) ^ GB(lfsr,17) ^ GB(lfsr,18) ^ GB(lfsr,19) ^ GB(lfsr,21) ^ GB(lfsr,22));
  SB(next_lfsr, 4, GB(lfsr,17));
```

```
  SB(next_lfsr, 3, GB(lfsr,16));
  SB(next_lfsr, 2, GB(lfsr,15) ^ GB(lfsr,22));
  SB(next_lfsr, 1, GB(lfsr,16) ^ GB(lfsr,18) ^ GB(lfsr,20) ^ GB(lfsr,22));
  SB(next_lfsr, 0, GB(lfsr,15) ^ GB(lfsr,17) ^ GB(lfsr,19) ^ GB(lfsr,21) ^ GB(lfsr,22));
  return(next_lfsr);
}

// Function to scramble a byte, given the current LFSR value
unsigned char scramble_data(unsigned long lfsr, unsigned char data_in) {
  unsigned char data_out = 0;
  SB(data_out, 7, GB(data_in,7) ^ GB(lfsr,15) ^ GB(lfsr,17) ^ GB(lfsr,19) ^ GB(lfsr,21) ^ GB(lfsr,22));
  SB(data_out, 6, GB(data_in,6) ^ GB(lfsr,16) ^ GB(lfsr,18) ^ GB(lfsr,20) ^ GB(lfsr,22));
  SB(data_out, 5, GB(data_in,5) ^ GB(lfsr,17) ^ GB(lfsr,19) ^ GB(lfsr,21));
  SB(data_out, 4, GB(data_in,4) ^ GB(lfsr,18) ^ GB(lfsr,20) ^ GB(lfsr,22));
  SB(data_out, 3, GB(data_in,3) ^ GB(lfsr,19) ^ GB(lfsr,21));
  SB(data_out, 2, GB(data_in,2) ^ GB(lfsr,20) ^ GB(lfsr,22));
  SB(data_out, 1, GB(data_in,1) ^ GB(lfsr,21));
  SB(data_out, 0, GB(data_in,0) ^ GB(lfsr,22));
  return(data_out);
}


// Example of LFSR and scrambled data “0” sequence for Lane 0
main() {
  unsigned long lfsr = 0x1DBFBC; // Lane 0 reset LFSR value
  unsigned char unscrambled_data = 0x00;

  int i;
  printf("Iteration LFSR   Next LFSR Scrambled Data\n");
  printf("--------- ------ --------- --------------\n");
  for (i = 0; i < 128; i++) {
    unsigned char scrambled_data = scramble_data(lfsr,unscrambled_data);
    unsigned long next_lfsr = calc_next_lfsr(lfsr);
    printf("%3d       %06X    %06X             %02X\n",
           i, lfsr, next_lfsr, scrambled_data);
    lfsr = next_lfsr;
  }
}
```

The initial 23-bit values of the LFSR for the first 128 LFSR advances for an 8-bit quantity following a reset for Lane 0 are listed below (ordered left to right, top to bottom):

	0, 8	**1, 9**	**2, A**	**3, B**	**4, C**	**5, D**	**6, E**	**7, F**
00	1DBFBC	498C2E	1186E9	0FC5AD	7CB75D	3D8DA2	0ECC8F	379717
08	046BDF	21D462	423FCE	5177D6	1447EF	67CBA0	794F7A	6CA2AF
10	425060	3ED9D6	3DBF74	3C1A8F	7AE2E0	073E71	137D99	70B139
18	44F521	559720	1FBBA8	689D9F	376B02	597FFA	297C0E	7E450A
20	4BDE36	669B58	6BB530	78C3F9	0322C0	45C7FB	254F6A	323D89
28	07FDD2	71DFBC	68726B	799E27	1CFE8A	4AB864	42CB12	04AAF3
30	41F947	51F808	3A98CA	36A816	796895	4B4DAF	54037D	68E5C7
38	4F3302	60A51F	3AFCE3	528116	248131	1F65E6	17D2BA	34987E
40	6C0524	44DA45	7AF320	16FE71	5A5134	60B5F5	2A16E3	73AFF1
48	3D3ADA	18B5AA	4B9306	2BAB58	2D179E	5FDE68	46E1F8	644B91
50	3F78A4	7FCE1B	23CC59	7F017F	4DA97C	7EDF8B	705E13	0ADE04
58	6F1775	318CBE	70CC0C	39C021	0944ED	33F8AB	21DCBD	4AE0CE
60	1A6112	1B2F92	17ADD8	4BFA7E	42D358	1CE0F3	54C164	0BFDE2
68	0EF33F	082717	1200E1	4FCB73	39D53A	1C5FED	4ADE41	24EE12
70	7046E6	122B04	642E73	5A9AA4	0A24D0	34C250	362B24	5B5BB0
78	2833BF	73F640	648BDA	5E3281	490B97	373ECC	0CB1FA	6FE7A6

An 8-bit value of 0x00 repeatedly encoded with the LFSR after reset for Lane 0 produces the following consecutive 8-bit values (ordered left to right, top to bottom):

	00	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**0A**	**0B**	**0C**	**0D**	**0E**	**0F**
00	6C	BD	94	98	53	C6	D8	CE	50	6A	75	C1	04	4F	C3	07
10	75	26	C6	06	A3	B0	B4	AB	05	11	CC	57	4E	69	42	73
20	1D	0F	B7	03	E0	45	BA	5E	30	EB	D7	43	2C	5D	F5	D0
30	15	41	76	8E	C3	9D	D1	57	CD	FF	76	A1	7A	4C	64	2E
40	87	05	A3	24	89	FF	A2	4B	46	7C	1D	62	12	19	A5	2F
50	E6	B3	CA	33	ED	F3	2B	88	67	3E	AB	96	E8	9E	6A	5D
60	5C	1C	64	1D	F5	2C	51	C8	D8	A8	F4	4D	96	AC	5D	7A
70	2B	F4	2F	09	08	2E	0E	C9	02	4B	AF	D9	3D	4E	78	E7

D. Request Dependencies

The *PCI Express Base Specification* does not specify the rules governing the creation of resource dependencies between TLPs using different Traffic Classes. Dependencies between packets in different Traffic Classes can create potential deadlock if devices make different assumptions about what is allowed and what is not. Dependencies can be created when a packet is forwarded (transmitted verbatim) or translated (transmitted with modification) from an input Port to an output Port.

Resource dependencies are created when received packets are forwarded/translated on the same or a different Link. Due to the fact that the forwarding/translating device has finite buffer resources this behavior creates a dependency between the ability to receive a packet and the ability to transmit a packet (in potentially a different VC or sub-channel).

The following notation is used to create a framework to enumerate the possibilities:

X(m) -> Y(n)

This means:

a request in sub-channel X(TC=m) is forwarded/translated in sub-channel Y(TC=n).

n and m are between 0-7.

X and Y are either P (Posted Request), N (Non-Posted Request), or C (Completion).

The list of possible dependencies is:

P(m) -> P(n)

P(m) -> N(n)

P(m) -> C(n)

N(m) -> P(n)

N(m) -> N(n)

N(m) -> C(n)

C(m) -> P(n)

C(m) -> N(n)

C(m) -> C(n)

For a given system, each of these dependencies needs to be classified as legal or illegal for each of the following cases:

- ❑ Root Port forwarding to own Link.
- ❑ Root Port forwarding to different Root Port's Link.
- ❑ Endpoint or Bridge forwarding to own Link.

A Switch is not allowed to modify the TLPs that flow through it, but must ensure complete independence of resources assigned to separate VCs. System software must comprehend the system dependency rules when configuring TC/VC mapping throughout the system.

One possible legal mapping is:

	RC (Same Port)	RC (Different Port)	Endpoint
P(m) -> P(n)	m <= n	m <= n	m < n
P(m) -> N(n)	m < n	m < n	m < n
P(m) -> C(n)	illegal	illegal	illegal
N(m) -> P(n)	m < n	m < n	m < n
N(m) -> N(n)	m <= n	m <= n	m < n
N(m) -> C(n)	m = n	m = n	m = n
C(m) -> P(n)	illegal	illegal	illegal
C(m) -> N(n)	illegal	illegal	illegal
C(m) -> C(n)	m >= n	m >= n	m > n

Note that this discussion only deals with avoiding the deadlock caused by the creation of resource dependencies. It does not deal with the additional livelock issues (or lack of forward progress) caused by the system's Virtual Channel arbitration policies.

Some of these potential dependencies are illegal or unreachable:

- ❑ P(m) -> P(n), N(m) -> N(n)
  - m = n – This case is illegal and will lead to deadlock, except when a Request is being forwarded from one Port to another of a Switch or Root Complex.
  - m! = n – See discussion below.
- ❑ P(m) -> N(n)
  - m = n – This case is illegal and will lead to deadlock.
  - m! = n – See discussion below.
- ❑ N(m) -> P(n) – See discussion below.
- ❑ P(m) -> C(n) – This case is illegal and will lead to deadlock.

- N(m) -> C(n)
  - m = n – This case occurs during the normal servicing of a non-posted request by either a root complex or an endpoint.
  - m! = n – This case is unreachable and should never happen. Completions always use the same TC as the corresponding request.
- C(m) -> P(n), C(m) -> N(n) – These cases are unreachable and should never happen due to the fact that completion buffers must be preallocated.
- C(m) -> C(n)
  - m = n – This case is illegal and will lead to deadlock, except when a Completion is being forwarded from one Port to another of a Switch or Root Complex.
  - m! = n – This case will occur if N(n)->N(m) dependencies are allowed.

Other potential dependencies may be legal when comprehended as part of a specific usage model. For example, these cases:

P(m) ->P(n), N(m) -> N(n), P(m) -> N(n), N(m) -> P(n) where m! = n

might exist where a device services incoming requests by issuing modified versions of those requests using a different Traffic Class (for example, remapping the first requests address and generating the new request with the resulting address). In these cases, suitable rules must be applied to prevent circular dependencies that would lead to deadlock or livelock.

Examples of devices that may find the above mappings useful:

- Bridges to complex protocols that require state to be save/restored to/from host memory, i.e., PCI Express to Infiniband bridges.
- Messaging engines that must do address translation based upon page tables stored in host memory.
- UMA graphics devices that store their frame buffer in host memory.

E. ID-Based Ordering Usage

E.1. Introduction

ID-Based Ordering (IDO) is a mechanism that permits certain ordering restrictions to be relaxed as a means to improve performance. IDO permits certain TLPs to pass other TLPs in cases where otherwise such passing would be forbidden. The passing permitted by IDO is not required for proper operation (e.g., deadlock avoidance); it is only a means of improving performance.

For discussing IDO, it's useful to introduce the concept of a "TLP stream", which is a set of TLPs that all have the same originator.[110] For several important cases where TLP passing is normally forbidden, IDO permits such passing to occur if the TLPs belong to different TLP streams.

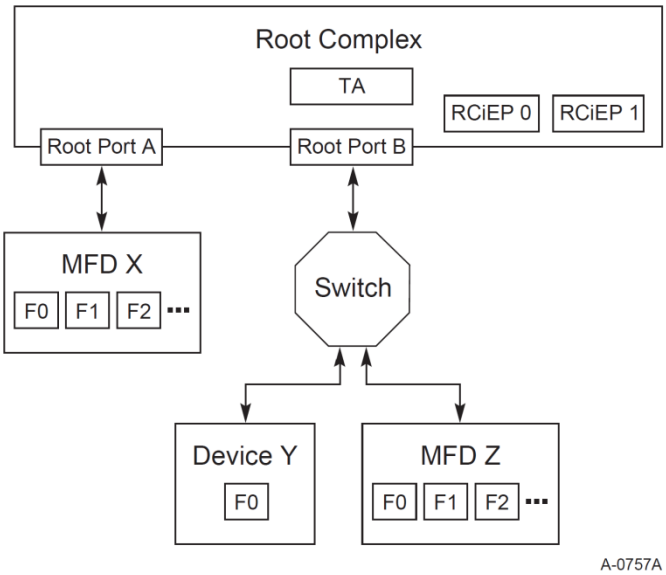


Figure E-1: Reference Topology for IDO Use

Figure E-1 shows a reference topology. The reference topology is not intended to discourage the use of IDO with other topologies, but rather to provide specific examples for discussion.

Devices X and Z are multi-Function devices (MFDs); device Y is a single-Function device. The RCiEPs are Root Complex Integrated Endpoint Functions, and might or might not be part of the

[110] That is, the Requester IDs of Requests and the Completer IDs of Completions are all the same.

same Device. We will assume that one or more Functions are using the Translation Agent (TA) in the Root Complex (RC).

Referring to the ordering table and descriptions in Section 2.4.1, having the IDO bit set in a Posted Request, Non-Posted Request, or Completion TLP permits that TLP to pass a Posted Request TLP if the two TLPs belong to different TLP streams. In the following examples, DMAR and DMAW stand for Direct Memory Access Read and Write; PIOR and PIOW stand for Programmed I/O Read and Write.

E.2. Potential Benefits with IDO Use

Here are some example potential benefits that are envisioned with IDO use. Generally, IDO provides the most benefit when multiple TLP streams share a common Link and that Link becomes congested, either due to high utilization or due to temporary lack of Flow Control (FC) credit.

E.2.1. Benefits for MFD/RP Direct Connect

Here are some examples in the context of traffic between MFD X and the RC in Figure E-1.

- Posted Request passing another Posted Request: when a DMAW from F0 is stalled due to an TA miss, if IDO is set for a DMAW from F1, it is permitted within the RC for this DMAW to pass the stalled DMAW from F0.
- Non-Posted Request passing a Posted Request: when a DMAW from F0 is stalled due to an TA miss, if IDO is set for a DMAR Request from F1, it is permitted within the RC for this DMAR Request to pass the stalled DMAW from F0.
- Completion passing a Posted Request: when a DMAW from F0 is stalled due to an TA miss, if IDO is set for a PIOR Completion from F1, it is permitted within the RC for this PIOR Completion to pass the stalled DMAW from F0.

E.2.2. Benefits for Switched Environments

Here are some examples in the context of traffic within the Switch in Figure E-1.

- Non-Posted Request passing a Posted Request: when a DMAW from Device Y is stalled within the Switch due to a lack of FC credit from Root Port B, if IDO is set for a DMAR Request from MFD Z, it is permitted within the Switch for this DMAR Request to pass the stalled DMAW from Device Y. The same also holds for a DMAR Request from one Function in MFD Z passing a stalled DMAW from a different Function in MFD Z.
- Completion passing a Posted Request: when a DMAW from Device Y is stalled within the Switch due to a lack of FC credit from Root Port B, if IDO is set for a PIOR Completion from MFD Z, it is permitted within the Switch for this PIOR Completion to pass the stalled DMAW from Device Y. The same also holds for a PIOR Completion from one Function in MFD Z passing a stalled DMAW from a different Function in MFD Z.
- Posted Request passing another Posted Request: within a Switch, there is little or no envisioned benefit from having a DMAW from one TLP stream passing a DMAW from a different TLP

stream. However, it is not prohibited for Switches to implement such passing as permitted by IDO.

E.2.3. Benefits for Integrated Endpoints

Here are some examples for the Root Complex Integrated Endpoints (RCiEPs) in Figure E-1. The benefits are basically the same as for the MFD/RP Direct Connect case.

- Posted Request passing another Posted Request: when a DMAW from RCiEP 0 is stalled due to an TA miss, if IDO is set for a DMAW from RCiEP 1, it is permitted for this DMAW to pass the stalled DMAW from RCiEP 0.
- Non-Posted Request passing a Posted Request: when a DMAW from RCiEP 0 is stalled due to an TA miss, if IDO is set for a DMAR Request from RCiEP 1, it is permitted for this DMAR Request to pass the stalled DMAW from RCiEP 0.
- Completion passing a Posted Request: when a DMAW from RCiEP 0 is stalled due to an TA miss, if IDO is set for a PIOR Completion from RCiEP 1, it is permitted for this PIOR Completion to pass the stalled DMAW from RCiEP 0.

E.2.4. IDO Use in Conjunction with RO

IDO and RO[111] are orthogonal. Certain instances of passing; for example, a Posted Request passing another Posted Request, might be permitted by IDO, RO, or both at the same time. While IDO and RO have significant overlap for some cases, it is highly recommended that both be used whenever safely possible. RO permits certain TLP passing within the same TLP stream, which is never permitted by IDO. For traffic in different TLP streams, IDO permits control traffic to pass any other traffic, and generally it is not safe to Set RO with control traffic.

E.3. When to Use IDO

With Endpoint Functions[112], it is safe to Set IDO in all applicable TLPs originated by the Endpoint when the Endpoint is directly communicating with only one other entity, most commonly the RC. For the RC case, "directly communicating" specifically includes DMA traffic, PIO traffic, and interrupt traffic; communicating with RCiEPs or communicating using P2P Root Port traffic constitutes communicating with multiple entities.

With a Root Port, there are no envisioned high-benefit use models where it is safe to Set IDO in all applicable TLPs that it originates. Use models where a Root Port Sets IDO in a subset of the applicable TLPs it originates are outside the scope of this specification.

---

[111] In this Appendix, "RO" is an abbreviation for the Relaxed Ordering Attribute field.

[112] Endpoint Functions include PCI Express Endpoints, Legacy PCI Express Endpoints, and Root Complex Integrated Endpoints.

E.4. When Not to Use IDO

E.4.1. When Not to Use IDO with Endpoints

With Endpoint Functions, it is not always safe to Set IDO in applicable TLPs it originates if the Endpoint directly communicates with multiple entities. It may be safe to Set IDO in some TLPs and not others, but such use models are outside the scope of this specification.

For example, in Figure E-1 if Device Y and MFD Z are communicating with P2P traffic and also communicating via host memory, it is not always safe for them to Set IDO in the TLPs they originate. As an example failure case, let's assume that Device Y does a DMAW (to host memory) followed by a P2P Write to MFD Z. Upon observing the P2P Write, let's assume that MFD Z then does a DMAW to the same location earlier targeted by the DMAW from Device Y. Normal ordering rules would guarantee that the DMAW from Device Y would be observed by host memory before the DMAW from MFD Z. However, if IDO is set in the DMAW from MFD Z, the RC would be permitted to have the second DMAW pass the first, causing a different end result in host memory contents.

Synchronization techniques like performing zero-length Reads might be used to avoid such communication failures when IDO is used, but specific use models are outside the scope of this specification.

E.4.2. When Not to Use IDO with Root Ports

With Root Ports, it is not always safe to Set IDO in applicable TLPs it originates if Endpoint Functions in the hierarchy do any P2P traffic. It may be safe to Set IDO in some TLPs and not others, but such use models are outside the scope of this specification.

As an example, in Figure E-1 if Device Y and MFD Z are communicating with P2P traffic and also communicating with host software, it is not always safe for Root Port B to Set IDO in the TLPs it originates. For example, let's assume that Device Y does a P2P Write to MFD Z followed by a DMAW (to host memory). Upon observing the DMAW, let's assume that the host does a PIOW to MFD Z. Normal ordering rules would guarantee that the P2P Write from Device Y would be observed by MFD Z before the PIOW from the host. However, if IDO is set in the PIOW from the host, the Switch would be permitted to have the PIOW pass the P2P Write, ultimately having the two Writes arrive at MFD Z out of order.

IMPLEMENTATION NOTE

Requester and Completer IDs for RC-Originated TLPs

With RC implementations where the Requester ID in a PIO Request does not match the Completer ID in a DMAR Completion, this enables another potential communication failure case if IDO is Set in the Completion. For this case, if a PIOW is followed by a DMAR Completion with IDO Set, a Switch below the Root Port could permit the DMAR Completion to pass the PIOW, violating the normal ordering rule that a non-RO Read Completion must not pass Posted Requests. The PIOW and DMAR Completion would appear to belong to different TLP streams, though logically they belong to the same TLP stream. Special caution is advised in setting IDO with TLPs originating from such RCs.

E.5. Software Control of IDO Use

E.5.1. Software Control of Endpoint IDO Use

By default, Endpoints are not enabled to Set IDO in any TLPs they originate.

IMPLEMENTATION NOTE

The "Simple" Policy for IDO Use

It is envisioned that Endpoints designed primarily to communicate directly with only one other entity (e.g., the RC) may find a "simple" policy for setting IDO to be adequate. Here's the envisioned "simple" policy. If the IDO Request Enable bit is Set, the Endpoint Sets IDO in all applicable Request TLPs that it originates. If the IDO Completion Enable bit is Set, the Endpoint Sets IDO in all Completion TLPs that it originates.

It is envisioned that a software driver associated with each Endpoint will determine when it is safe for the Endpoint to set IDO in applicable TLPs it originates. A driver should be able to determine if the Endpoint is communicating with multiple other entities, and should know the Endpoint's capabilities as far as setting IDO with all applicable TLPs when enabled, versus setting IDO selectively. If a driver determines that it is safe to enable the setting of IDO, the driver can set the IDO Request Enable and/or IDO Completion Enable bits either indirectly via OS services or directly, subject to OS policy.

If an Endpoint is designed for communication models where it is not safe to utilize the "simple" policy for IDO use, the Endpoint can implement more complex policies for determining when the Endpoint sets the IDO bit. Such implementations might utilize device-specific controls that are managed by the device driver. Such policies and device-specific control mechanisms are outside the scope of this specification.

E.5.2. Software Control of Root Port IDO Use

Since there are no envisioned high-benefit "simple" use models for Root Ports setting the IDO bit with TLPs they originate, and there are known communication failure cases if Root Ports set the IDO bit with all applicable TLPs they originate, it is anticipated that Root Ports will rarely be enabled to set IDO in TLPs they originate. Such use models and policies for Root Ports setting IDO are outside the scope of this specification.

F. Message Code Usage

Table F-1 contains a list of currently defined PCI Express Message Codes. Message codes are defined in this specification and in other specifications. This table will be updated as Messages are defined in other specifications but due to document release schedules, this table might not contain recently defined Messages.

Table F-1: Message Code Usage

Message Code	Routing r[2:0]	Type	Description
0000 0000	011	Msg	Unlock, see Section 2.2.8.4
0000 0001	010	MsgD	Invalidate Request Message, see Section 10.3.1.
0000 0010	010	Msg	Invalidate Completion Message, see Section 10.3.2.
0000 0100	000	Msg	Page Request Message, see Section 10.4.1.
0000 0101	010	Msg	PRG Response Message, see Section 10.4.2.
0001 0000	100	Msg	Latency Tolerance Reporting (LTR) Message, see Section 2.2.8.8
0001 0010	100	Msg	Optimized Buffer Flush/Fill (OBFF) Message, see Section 2.2.8.9
0001 0100	100	Msg	PM_Active_State_Nak, see Section 2.2.8.2
0001 1000	000	Msg	PM_PME, see Section 2.2.8.2
0001 1001	011	Msg	PME_Turn_Off, see Section 2.2.8.2
0001 1011	101	Msg	PME_TO_Ack, see Section 2.2.8.2
0010 0000	100	Msg	Assert_INTA, see Section 2.2.8.1
0010 0001	100	Msg	Assert_INTB, see Section 2.2.8.1
0010 0010	100	Msg	Assert_INTC, see Section 2.2.8.1
0010 0011	100	Msg	Assert_INTD, see Section 2.2.8.1
0010 0100	100	Msg	Deassert_INTA, see Section 2.2.8.1
0010 0101	100	Msg	Deassert_INTB, see Section 2.2.8.1
0010 0110	100	Msg	Deassert_INTC, see Section 2.2.8.1
0010 0111	100	Msg	Deassert_INTD, see Section 2.2.8.1
0011 0000	000	Msg	ERR_COR, see Section 2.2.8.3

Message Code	Routing r[2:0]	Type	Description
0011 0001	000	Msg	ERR_NONFATAL, see Section 2.2.8.3
0011 0011	000	Msg	ERR_FATAL, see Section 2.2.8.3
0100 0000	100	Msg	Ignored Message, see Section 2.2.8.7
0100 0001	100	Msg	Ignored Message, see Section 2.2.8.7
0100 0011	100	Msg	Ignored Message, see Section 2.2.8.7
0100 0100	100	Msg	Ignored Message, see Section 2.2.8.7
0100 0101	100	Msg	Ignored Message, see Section 2.2.8.7
0100 0111	100	Msg	Ignored Message, see Section 2.2.8.7
0100 1000	100	Msg	Ignored Message, see Section 2.2.8.7
0101 0000	100	MsgD	Set_Slot_Power_Limit, see Section 2.2.8.5
0101 0010	100	Msg	PTM Request, see Section 2.2.8.10
0101 0011	100	Msg/MsgD	PTM Response/PTM ResponseD, see Section 2.2.8.10
0111 1110	000, 010, 011, or 100	Msg/MsgD	Vendor_Defined Type 0, see Section 2.2.8.6
0111 1111	000, 010, 011, or 100	Msg/MsgD	Vendor_Defined Type 1, see Section 2.2.8.6

Table F-2 contains a list of currently defined Subtype codes for PCI-SIG-Defined VDMs (see Section 2.2.8.6.1). Subtype codes are defined in this specification and in other specifications. This table will be updated as Subtype codes are defined in other specifications but due to document release schedules, this table might not contain recently defined Subtypes.

Table F-2: PCI-SIG-Defined VDM Subtype Usage

Subtype	Routing r[2:0]	Type	Description
0000 0000	010 or 011	MsgD	LN Message, see Section 2.2.8.6.1.1
0000 0001	011	MsgD	Hierarchy ID Message, See Section 2.2.8.6.5 and Section 6.26
0000 1000	100	Msg	Device Readiness Status, see Section 2.2.8.6.3
0000 1001	000	Msg	Function Readiness Status, see Section 2.2.8.6.4

G

G. Protocol Multiplexing

The Protocol Multiplexing mechanism provides a standard mechanism to transport non-PCI Express protocols across a PCI Express Link. The mechanism supports the multiplexing of PMUX Packets and TLPs onto a single PCI Express Link.

An example system topology using Protocol Multiplexing is shown in Figure G-1. In this example, the Link may operate in two modes:

- PCI Express Link. Protocol Multiplexing is disabled.
- PMUX Link. Protocol Multiplexing is enabled. Both TLPs and PMUX Packets are used in a coordinated fashion. PMUX Packets may be used to support additional protocols efficiently.

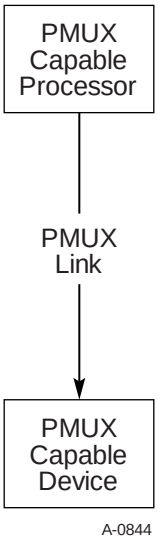


Figure G-1: Device and Processor Connected Using a PMUX Link

A PMUX Link is shown in Figure G-2. Arbitration and encapsulation occurs between the transmit queues and the Link. Demultiplexing and decapsulation occurs between the Link and the various receive queues. Packets are sent from transmit queues to the corresponding receive queues. Packets are identified as either PMUX Packets or TLPs.

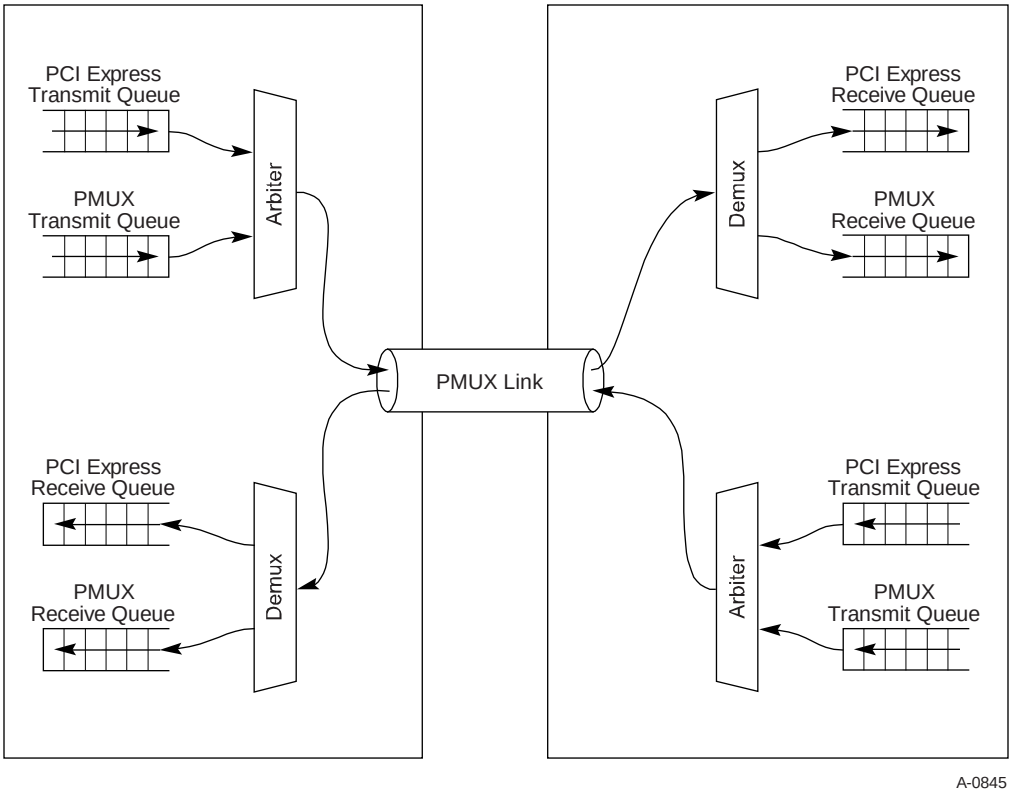


Figure G-2: PMUX Link

Important attributes of the Protocol Multiplexing mechanism are:

- Protocol Multiplexing support is optional normative.
- Protocol Multiplexing has no impact on PCI Express components that do not support it.
- Protocol Multiplexing has no impact on PCI Express TLPs and DLLPs, even when it is enabled.
- A Link may be used for both TLPs and PMUX Packets at the same time.
- Protocol Multiplexing does not consume or interfere with PCI Express resources (sequence numbers, credits, etc.). PMUX Packets use distinct resources associated with the specific multiplexed protocol.
- Protocol Multiplexing is disabled by default and is enabled by software. PMUX Packets must not be sent until enabled by software. PMUX Packets received at Ports that support Protocol Multiplexing are ignored until Protocol Multiplexing is enabled by software.
- Protocol Multiplexing is selectable on a per-Link basis. Protocol Multiplexing may be used on any collection of Links in a system.
- A PMUX Link may support up to four simultaneously active PMUX Channels. Software configures the protocol used on each PMUX Channel.

- PMUX Packets contain an LCRC. This is used to provide data resiliency in a similar fashion as PCI Express TLPs.
- PMUX Packets do not use the ACK/NAK mechanism of PCI Express. Multiplexed protocol specific acknowledgement mechanisms can be used to provide reliable delivery when needed.
- PMUX Packets do not contain a TLP Sequence Number. Instead, they contain a 12 bit PMUX Packet Metadata field that is available for multiplexed protocol specific use.
- PMUX Packet transmitters must contain some arbitration/QoS mechanism for scheduling sending of PMUX Packets, TLPs and DLLPs; however, the mechanism used is outside the scope of this specification.
- The Protocol Multiplexing mechanism does not define any addressing or routing mechanism for PMUX Packets.

PMUX Packets are similar to PCI Express TLPs. The PMUX Packet Flow Through is shown in Figure G-3. The PCI Express Packet Flow Through is shown in Figure 1-5. Changes from PCI Express Packet Flow Through are:

- PMUX Packets use a protocol specific PMUX Protocol Layer instead of the PCI Express Transaction Layer.
- PMUX Packets use a simplified Data Link Layer. The packet integrity portion of the Data Link Layer is mostly unchanged (LCRC computation uses a different seed value). The reliability and flow control aspects of the Data Link Layer are removed (the TLP Sequence Number field is repurposed as PMUX Packet Metadata).
- The Physical Layer is slightly modified to provide a mechanism to identify PMUX Packets.

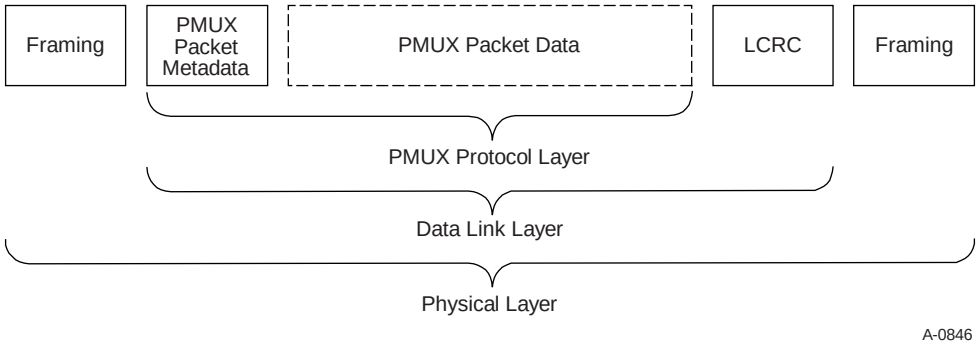


Figure G-3: PMUX Packet Flow Through the Layers

G.1. Protocol Multiplexing Interactions with PCI Express

Table G-1 and Table G-2 describe interactions between Protocol Multiplexing and PCI Express. Table G-1 describes how PCI Express attributes affect Protocol Multiplexing. Table G-2 describes how Protocol Multiplexing features affect PCI Express attributes.

Table G-1: PCI Express Attribute Impact on Protocol Multiplexing

PCI Express Attribute	Impact on Protocol Multiplexing
Link Speed	All PMUX Channels are disabled when the Current Link Speed corresponds to a speed that is not supported by Protocol Multiplexing (see Section G.5.4). A PMUX Channel may be disabled when the Current Link Speed corresponds to a speed that is not supported by the associated protocol.[113] Link speed can change either explicitly due to a change in the Target Link Speed field or automatically due to an autonomous Link speed change (see *PCI Express Base Specification* Section 6.11). The PMUX Protocol Layer is permitted to influence the mechanism used by a component to determine when it requests an autonomous Link speed change. In addition, setting Hardware Autonomous Speed Disable at each end of the Link will prevent certain autonomous Link speed changes (see Section 7.8.19). The PMUX Protocol Layer may be notified of the change.
Link Width	A PMUX Channel may be disabled when the Link Width corresponds to a width that is not supported by the associated protocol.[114] The PMUX Protocol Layer is permitted to influence the mechanism used by a component to determine when it requests a Link Width change. The PMUX Protocol Layer may be notified of the change.
FLR initiated	All PMUX Channels on a Link are disabled if an FLR is directed to the Upstream Port's Function 0. No PMUX Channels are affected if an FLR is directed to any other Function.
DL_Down	All PMUX Channels on a Link are disabled.
Hot Reset	All PMUX Channels on a Link are disabled.
PERST#	All PMUX Channels on a Link are disabled.
TLP Replay	No effect on Protocol Multiplexing.
DLLP Lost	No effect on Protocol Multiplexing.
TLP Prefix	No effect on Protocol Multiplexing.

[113] The mechanism software uses to determine what Link Speeds are supported by a protocol is outside the scope of this specification.

[114] The mechanism software uses to determine what Link Widths are supported by a protocol is outside the scope of this specification.

PCI Express Attribute	Impact on Protocol Multiplexing
Locked Transactions	No effect on Protocol Multiplexing (including no effect on any protocol specific forwarding of PMUX Packets).
AtomicOp Transactions	No effect on Protocol Multiplexing.
Multicast Transactions	No effect on Protocol Multiplexing.
Access Control Services (ACS)	No effect on Protocol Multiplexing.
Alternative Routing ID-Interpretation (ARI)	No effect on Protocol Multiplexing.
TLP Processing Hints (TPH)	No effect on Protocol Multiplexing.
Virtual Channels	No effect on Protocol Multiplexing. PCI Express Links remain capable of supporting a full complement of VCs.
Internal Error	Corrected or Uncorrectable Internal Errors in the PMUX Protocol Layer may be reported as PCI Express Internal Errors.
L0s Link Power State	Protocol Multiplexing tracks the Link state. The PMUX Protocol Layer may request the Link transition back to L0.
L1 Link Power State	Protocol Multiplexing tracks the Link state. The PMUX Protocol Layer may request the Link transition back to L0.
Disabled LTSSM State	Disabling a Link also disables all PMUX Channels on the Link.
Loopback LTSSM State	Entering Loopback state disables all PMUX Channels on the Link.
Recovery LTSSM State	No effect on Protocol Multiplexing. The PMUX Protocol Layer may be notified.
Receiver or Framing Error	The error is reported to the PMUX Protocol Layer to indicate that data might have been lost. This can be used to initiate protocol specific error recovery mechanisms. The Error is reported to software using PCI Express Mechanisms.
Lane Reversal	No effect on Protocol Multiplexing. Support for Lane Reversal remains optional.
Polarity Inversion	No effect on Protocol Multiplexing.
Crosslink	No effect on Protocol Multiplexing. Support for Crosslink remains optional. If supported, the PMUX Protocol Layer may be notified of the outcome of the Crosslink Upstream / Downstream negotiation.
Lane assignment rules	Placement and frequency rules for STP Symbols and STP Tokens are not changed (see Section 4.2.1.2 and Section 4.2.2.3.2). These rules apply identically to PCI Express TLPs and PMUX Packets.
PCI Power Management Power State	All PMUX Channels on a Link are disabled if the Upstream Port's Function 0 is sent to non-D0 state.
Dynamic Power Allocation (DPA)	No effect on Protocol Multiplexing. The PMUX Protocol Layer is notified of the change and may participate in the power reduction. PCI Express power management includes any power used by the PMUX Protocol Layer.

PCI Express Attribute	Impact on Protocol Multiplexing
PCI Power Management Power Consumed / Power Dissipated / Aux_Current	Power required by the PMUX Protocol Layer is included in the PCI structures.
Power Budgeting	Power required by the PMUX Protocol Layer is included in the PCI Express structures.
Slot Power Limit	Slot Power Limit includes power available to PMUX Protocol Layer.
ASPM L0s Entry Condition	The definition of Idle is extended to include: ❑ No pending PMUX Packets to transmit over the Link. ❑ For PMUX Channels that use protocol specific Flow Control, no credits are available to send PMUX Packets in that PMUX Channel.
ASPM L1 Entry Condition	A Link may not enter L1 if PMUX Packets are pending or scheduled to be transmitted.
ASPM L0s/L1 Exit Conditions	A Link may be directed to exit L0s or L1 if a component needs to transmit a PMUX Packet. Routing of PMUX Packets through routing elements is outside the scope of this specification; the associated L0s/L1 exit rules are also unspecified.
Bus Renumbering	No effect on Protocol Multiplexing.
Hot Plug	No direct effect on Protocol Multiplexing. Note Hot Plug events indirectly affect Data Link State which, in turn affects Protocol Multiplexing.
TLP Sequence Number	No effect on Protocol Multiplexing. PMUX Packets do not consume TLP Sequence Numbers.
PCI Express Flow Control	No effect on Protocol Multiplexing. PMUX Packets do not consume PCI Express Flow Control credits. Flow Control Update DLLPs must be sent as required by PCI Express.
Error Reporting	No direct effect on Protocol Multiplexing. The PMUX Protocol Layer may be notified when an error is signaled or when an error message is received.
LCRC Errors in TLPs	No effect on Protocol Multiplexing.
Nullified TLPs	No effect on Protocol Multiplexing.
VC Arbitration	No effect on Protocol Multiplexing. Arbitration within PCI Express is unaffected by Protocol Multiplexing.
Port Arbitration	No effect on Protocol Multiplexing. Arbitration within PCI Express is unaffected by Protocol Multiplexing.
Electrical Idle Inference	PMUX Packets count as TLPs for the purpose of inferring Electrical Idle.
MR-IOV	Protocol Multiplexing may co-exist with MR-IOV. PMUX Packets are not part of any MR-IOV Virtual Hierarchy. Protocol Multiplexing is controlled using configuration space in the Management VH(s).

Table G-2: PMUX Attribute Impact on PCI Express

PMUX Attribute	Impact on PCI Express
PMUX Protocol Error	No effect on PCI Express.
LCRC Errors in PMUX Packets	No effect on PCI Express. PMUX Packets with LCRC errors are discarded without triggering PCI Express replay. This error is reported to the PMUX Protocol Layer and can be used to initiate protocol specific error recovery and/or error reporting mechanisms.
Link Unreliability	The PMUX Protocol Layer is permitted to influence the mechanism used by a component to determine if it requests an autonomous link speed change.
Nullified PMUX Packets	No effect on PCI Express. It is protocol specific whether PMUX Packets within a specific PMUX Channel may be nullified. If supported, PMUX Packets are nullified in the same manner as TLPs (e.g., inverting the LCRC and signaling nullification at the Physical Layer). Receiving a nullified PMUX Packet may be reported to the PMUX Protocol Layer.
Electrical Idle Inference	PMUX Packets count as TLPs for the purpose of inferring Electrical Idle.
PMUX Protocol Layer directs LTSSM to enter Recovery	Both PCI Express and the PMUX Protocol Layer are permitted to direct a transition from L0 to Recovery.
PMUX Channel Enabled / Disabled	No effect on PCI Express
PMUX Packet Receiver Buffer Overflow	No effect on PCI Express. This is a protocol problem within the PMUX Channel. The PMUX Transport Layer must continue to accept such packets dispose of them using protocol specific mechanisms.
Received PMUX Packet larger or smaller than supported by the associated protocol	No effect on PCI Express. These are protocol problems within the PMUX Channel. The PMUX Transport Layer must accept such packets and dispose of them using protocol specific mechanisms.
Received PMUX Packet that contains more than 125 DWORDs of PMUX Packet Data	No effect on PCI Express. This is an invalid PMUX Packet. The PMUX Transport Layer must accept such a packet and dispose of it. A protocol specific mechanism may be used to report the error. Note: This situation only exists for a packet encoded using 8b/10b. The TLP Length field of a packet encoded using 128b/130b cannot contain values that cause this situation.
PMUX Packet Received on disabled PMUX Channel	No effect on PCI Express. No effect on any other PMUX Channel. Receivers must silently ignore such packets regardless of packet length and regardless of whether or not the packet is nullified. PMUX Packets arriving on a disabled PMUX Channel may occur normally when software is in the process of initializing Protocol Multiplexing.

PMUX Attribute	Impact on PCI Express
PMUX Packet Received at component that does not support Protocol Multiplexing	Software should not enable PMUX Packets unless both ends of a Link support Protocol Multiplexing. In the 128b/130b encoding, receiving a PMUX Packet by a component that does not support Protocol Multiplexing is a Framing Error (see Section 4.2.2.3.1). In the 8b/10b encoding, the PMUX Packet LCRC is computed differently that the TLP LCRC. Receivers that do not support Protocol Multiplexing will interpret PMUX Packets as TLPs with LCRC errors and will not process them.
Large PMUX Packets when PCI Express Max_Payload_Size is small	Under certain conditions, it is possible for a large PMUX Packet to trigger a premature PCI Express replay. For example, this can occur when the time needed to transmit a PMUX Packet is larger than the REPLAY_TIMER (see Section 3.5.2.1). To avoid this issue, implementations are permitted to not advance (hold) their REPLAY_TIMER during the reception of PMUX Packets. Note: The PCI Express REPLAY_TIMER mechanism has adequate headroom for most cases. This issue exists when (1) Max_Payload_Size is 000b, (2) PMUX Packets are larger than about 80 DWORDs, and (3) the REPLAY_TIMER is at the low end of the -0%/+100% tolerance.

G.2. PMUX Packets

A PMUX Packet contains the information shown in Figure G-4.

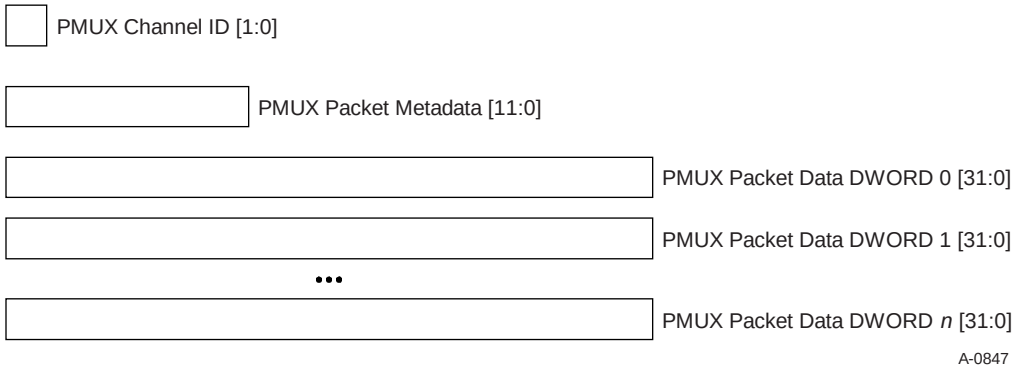


Figure G-4: PMUX Packet

PMUX Channel ID is a 2 bit field that identifies which protocol is associated with a PMUX Packet. PMUX Channel ID values are between 0 and 3 (inclusive).

PMUX Packet Metadata is a 12 bit field that provides information about the PMUX Packet. Definition of this field is protocol specific and is outside the scope of this specification.

A PMUX Packet consists of between 0 and 125 DWORDs of PMUX Packet Data. Layout and usage of these DWORDs is protocol specific and is outside the scope of this specification. A PMUX Packet need not have any PMUX Packet Data and may consist only of PMUX Channel ID and PMUX Packet Metadata.

G.3. PMUX Packet Layout

There are two layouts defined for PMUX Packets. One layout is used for 2.5 and 5.0 GT/s data rates and another layout is used for 8.0 GT/s and higher data rates. These layouts are discussed in the following sections.

G.3.1. PMUX Packet Layout for 8b/10b Encoding

Figure G-5 and Table G-3 show the layout of PMUX Packets when using 8b/10b encoding. For reference, the 8b/10b encoding of a TLP is also shown (see Section 4.2.1.2 for the official definition). In Table G-3, items shown in *italics* are identical in PMUX Packets and TLPs.

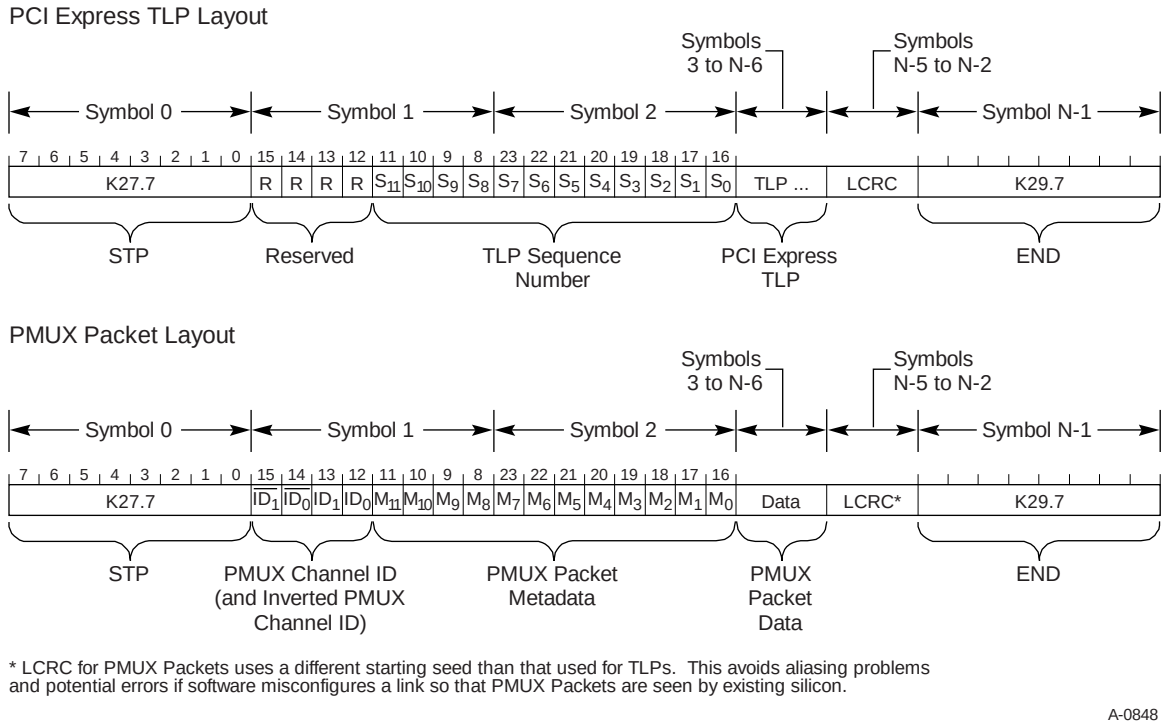


Figure G-5: TLP and PMUX Packet Framing (8b/10b Encoding)

Table G-3: PMUX Packet Layout (8b/10b Encoding)

Symbol	Field	Bit Position(s)	PMUX Packet Usage	TLP Usage
0	*Start TLP Indicator*	*7:0*	*K27.7*	
1	Inverted PMUX Channel ID [1:0]	7:6	Inverted (1s complement) of Symbol 1 bits 6:4	Reserved
	PMUX Channel ID[1:0]	5:4	PMUX Channel ID	Reserved
	PMUX Packet Metadata[11:8]	3:0	PMUX Packet Metadata[11:8]	TLP Sequence Number[11:8]
2	PMUX Packet Metadata[7:0]	7:0	PMUX Packet Metadata[7:0]	TLP Sequence Number[7:0]
3 to N-6	Packet	7:0	PMUX Packet	TLP
N-5 to N-2	LCRC	7:0	PMUX LCRC	PCI Express LCRC
N-1	*END*	*7:0*	*K29.7*	

For PMUX Packets, symbols 1 and 2 contain PMUX Packet Metadata in the same bit positions that TLPs use for TLP Sequence Number.

The PMUX LCRC algorithm is identical to the TLP LCRC algorithm as described in Section 3.5.2 with the following modifications:

- The seed value is FB3E E248h (TLP LCRC uses FFFF FFFFh).
- The PMUX Channel ID field in Symbol 1 bits 7:4 is included in the PMUX LCRC in the same manner as the 4 reserved bits in the TLP LCRC.
- The PMUX Packet Metadata field is included in the PMUX LCRC in the same manner as the TLP Sequence Number field is included in the TLP LCRC.

IMPLEMENTATION NOTE

PMUX Packets at Receivers that do not Support Protocol Multiplexing

The bits used for PMUX Channel ID are reserved unless Protocol Multiplexing is supported. As such, Receivers that do not support Protocol Multiplexing must ignore the PMUX Channel ID bits. If software misconfigures Protocol Multiplexing, a component that does not support Protocol Multiplexing could receive a PMUX Packet. To prevent that component from misinterpreting such a PMUX Packet as a valid TLP, the LCRC computation is changed for PMUX Packets. The result is that a valid PMUX Packet will never be misinterpreted as a valid TLP. These LCRC "errors" may trigger PCI Express replay and may result in REPLAY_NUM Rollover correctable errors being reported.

IMPLEMENTATION NOTE

PMUX Packet LCRC

The PMUX Channel ID field is covered by the LCRC. As such, when using 8b/10b encoding, receivers must wait until the LCRC is checked to make firm decisions based on the PMUX Channel ID value. The Inverted PMUX Channel ID can be compared against the PMUX Channel ID to make tentative decisions.

Note: The value of the LCRC associated with a given PMUX Packet is independent of the encoding used to transmit the packet.

G.3.2. PMUX Packet Layout at 128b/130b Encoding

Figure G-6 and Table G-4 show the layout of PMUX Packets when using 128b/130b encoding. For reference, the 128b/130b encoding of a TLP is also shown (see Section 4.2.2.2 for the official definition). In Table G-4, items shown in *italics* are identical in PMUX Packets and TLPs.

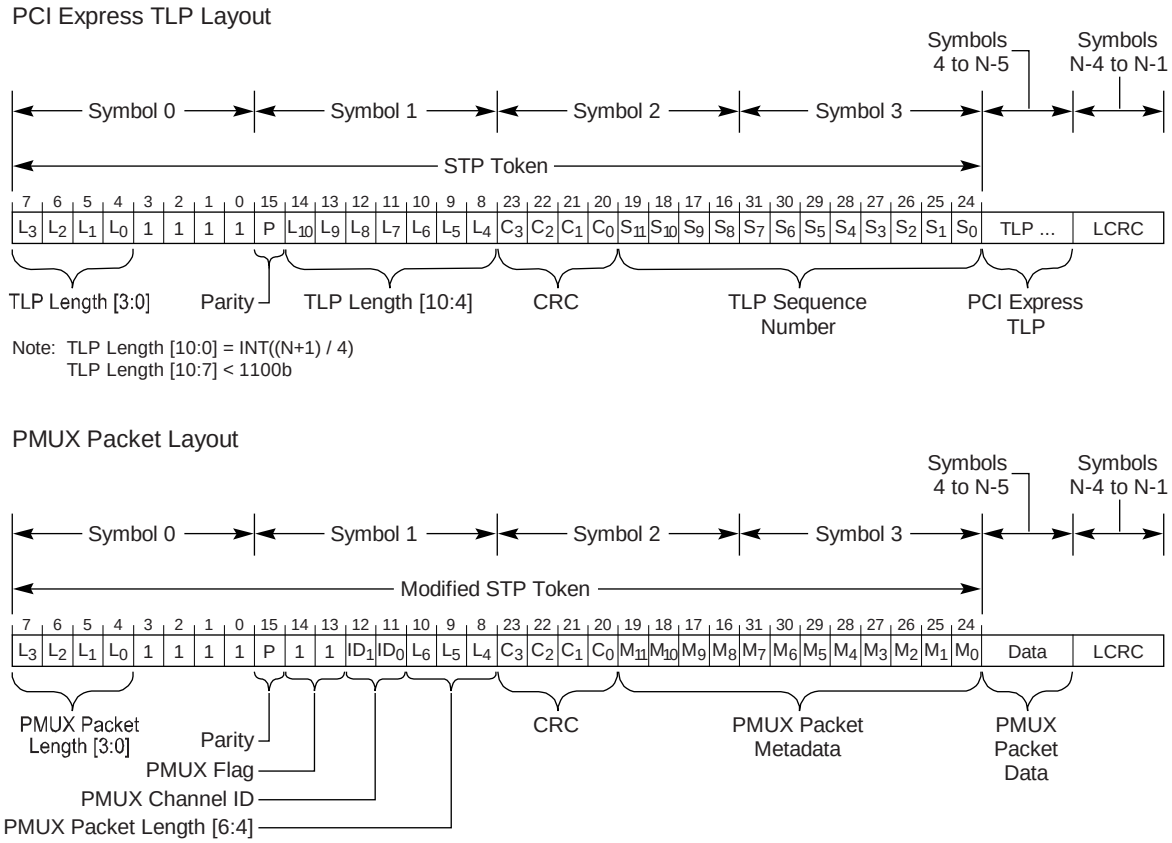


Figure G-6: TLP and PMUX Packet Framing (128b/130b Encoding)

Table G-4: PMUX Packet Layout (128b/130b Encoding)

Symbol	Field	Bit Position(s)	PMUX Packet Usage	TLP Usage
0	*Start TLP Indicator*	*3:0*	*Value of 1111b*	
	PMUX Packet Length[3:0]	7:4	Bits [3:0] of the PMUX Packet Length. Bit 0 is the least significant PMUX Packet Length bit.	Bits [3:0] of the TLP Length field. Bit 0 is the least significant TLP Length bit.
1	*Frame Parity (P)*	*7*	*Even parity of Symbol 0 bits [7:4], Symbol 1 bits [6:0] and Symbol 2 bits [7:4]*	
	PMUX Packet Indicator	6:5	Value of 11b	Bits [10:4] of the TLP Length field. Bit 10 is the most significant TLP Length bit.
	PMUX Channel ID[1:0]	4:3	PMUX Channel ID	
	PMUX Packet Length[6:4]	2:0	Bits [6:4] of PMUX Packet Length. Bit 6 is the most significant PMUX Packet Length bit.	
2	PMUX Packet Metadata[11:8]	3:0	PMUX Packet Metadata[11:8]	TLP Sequence Number[11:8]
	Frame CRC (C[3:0])	*7:4*	*CRC of Symbol 0, bits [7:4] and Symbol 1 bits [6:0]*	
3	PMUX Packet Metadata[7:0]	7:0	PMUX Packet Metadata[7:0]	TLP Sequence Number[7:0]
4 to N-5	Packet	7:0	PMUX Packet	TLP
N-4 to N-1	*LCRC*	*7:0*	*LCRC*	

Table G-5 describes the encodings of Symbol 1 bits [6:3] in more detail. If these bits contain a value less than 1001b, the packet is a TLP and is processed as described in Section 4.2.2.[115] If these bits contain 1001b, 1010b, or 1011b, the encoding is reserved for future standardization and is processed as described in Section 4.2.2.3.3. If these bits contain a value greater than or equal to 1100b, the packet is a PMUX Packet is defined as specified in this appendix.[116]

[115] The value 1001b supports a maximum TLP Length [10:0] value of 1151 DWORDs (decimal).This will accommodate a TLP consisting of 4096 bytes of payload, 16 bytes of TLP Header, 4 bytes of TLP digest, and 480 bytes of TLP Prefix.

[116] The value 1100b was chosen to simplify distinguishing PMUX Packets from TLPs and from the reserved encodings.

Table G-5: Symbol 1 Bits [6:3]

Symbol 1 bits [6:3]	Meaning
0xxxb or 1000b	Packet is a TLP. Bits [6:3] are TLP Length [10:7].
1001b, 1010b, or 1011b	Encoding reserved for future standardization. Receivers detecting these encodings shall process them as described in Section 4.2.2.3.3.
1100b	Packet is a PMUX Packet. PMUX Channel ID is 0.
1101b	Packet is a PMUX Packet. PMUX Channel ID is 1.
1110b	Packet is a PMUX Packet. PMUX Channel ID is 2.
1111b	Packet is a PMUX Packet. PMUX Channel ID is 3.

For PMUX Packets, the packet length in DWORDs is contained in PMUX Packet Length [6:0]. Other than being a smaller field, PMUX Packet Length is interpreted in the same manner as TLP Length. Specifically, PMUX Packet Length also includes the framing and PMUX LCRC DWORDs (see Section 4.2.2.2).

For PMUX Packets, symbols 2 and 3 contain PMUX Packet Metadata in the same bit positions that TLPs use for TLP Sequence Number.

The PMUX LCRC algorithm is identical to the TLP LCRC algorithm as described in Section 3.5.2 with the following modifications:

- The seed value is FB3E E248h (TLP LCRC uses FFFF FFFFh).
- The PMUX Channel ID field in Symbol 1 bits 4:3 is used to compute a 4 bit value that is included in the PMUX LCRC in the same manner as the 4 reserved bits in the TLP LCRC. This 4 bit value contains the value that would be used, by the 8b/10b encoding, for Symbol 1 bits 7:4. Specifically, the lower 2 bits of this 4 bit value contain the PMUX Channel ID and the upper 2 bits contain the inverse (1s complement) of the PMUX Channel ID.
- The PMUX Packet Metadata field is included in the PMUX LCRC in the same manner as the TLP Sequence Number field is included in the TLP LCRC.

The Frame CRC and Frame Parity fields are computed as shown below. This is the same algorithm computed over the same bit positions as defined in Section 4.2.2.2.

C[0] = 1b ^ PMUX_Channel_ID[0] ^ L[6] ^ L[4] ^ L[2] ^ L[1] ^ L[0]

C[1] = 1b ^ 1b ^ PMUX_Channel_ID[0] ^ L[5] ^ L[4] ^ L[3] ^ L[2]

C[2] = 1b ^ PMUX_Channel_ID[1] ^ L[6] ^ L[4] ^ L[3] ^ L[2] ^ L[1]

C[3] = PMUX_Channel_ID[1] ^ PMUX_Channel_ID[0] ^ L[5] ^ L[3] ^ L[2] ^ L[1] ^ L[0]

P = 1b ^ 1b ^ PMUX_Channel_ID[1] ^ PMUX_Channel_ID[0] ^ L[6] ^ L[5] ^ L[4] ^ L[3] ^ L[2] ^ L[1] ^ L[0] ^ C[3] ^ C[2] ^ C[1] ^ C[0]d

IMPLEMENTATION NOTE

PMUX Channel ID and Frame CRC

When using 128b/130b encoding, the PMUX Channel ID field is covered by the Frame CRC and Frame Parity fields. As such, receivers may make decisions based on the PMUX Channel ID value as soon as the Frame CRC and Frame Parity is checked and need not wait until the PMUX LCRC is checked.

Note: The PMUX Channel ID is also covered by the LCRC. The value of the LCRC associated with a given PMUX Packet is independent of the encoding used to transmit the packet.

G.4. PMUX Control

Protocol Multiplexing is disabled by default. Each PMUX Channel must be explicitly enabled by software at each end of the associated Link. Protocol Multiplexing is disabled whenever the link drops (Data Link Layer indicates DL_Down).

A component that supports Protocol Multiplexing indicates such by the presence of the PMUX Extended Capability.

The following rules apply to components that support Protocol Multiplexing:

- PMUX Packets received in a PMUX Channel that is not enabled are silently ignored.
- PMUX Packets may not be transmitted unless the associated PMUX Channel is enabled. A PMUX Channel may also require additional, protocol specific, initialization mechanisms before PMUX Packets may be transmitted.

G.5. PMUX Extended Capability

Figure G-7 shows the PMUX Extended Capability structure. The presence of this capability indicates that the Port supports the optional Protocol Multiplexing mechanism. This capability is optional and may be present in any Downstream Port and in Function 0 of any Upstream Port. It must not be present in non-zero Functions of Upstream Ports or in RCRBs.

The length of the PMUX Extended Capability is determined by the PMUX Protocol Array Size field (see Section G.5.2).

This capability contains a list of the protocols supported by the Link (the PMUX Protocol Array). It also contains the mechanism software uses to enable and configure PMUX Channels. This capability must be present in both the Upstream and Downstream Ports of a Link in order for Protocol Multiplexing to be successfully enabled.

Software may enable the Upstream and Downstream Ports of a Link in either order. Software may enable multiple PMUX Channels using a single write to the PMUX Control Register.

Behavior is undefined if software enables Protocol Multiplexing in one Port and the other Port of the Link does not support Protocol Multiplexing. Behavior is also undefined if software configures a PMUX Channel inconsistently (the same PMUX Channel in the Ports on each end of a Link configured with incompatible protocols).

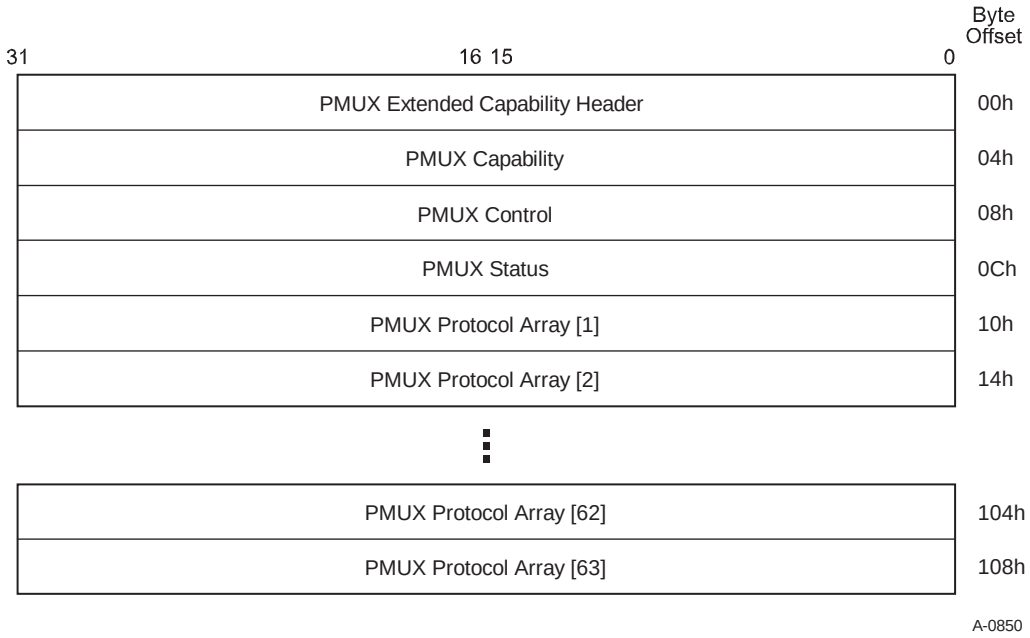


Figure G-7: PMUX Extended Capability

G.5.1. PCI Express Extended Header (Offset 00h)

Figure G-8 details the allocation of fields in the PMUX Extended Capability header; Table G-6 provides the respective bit definitions.

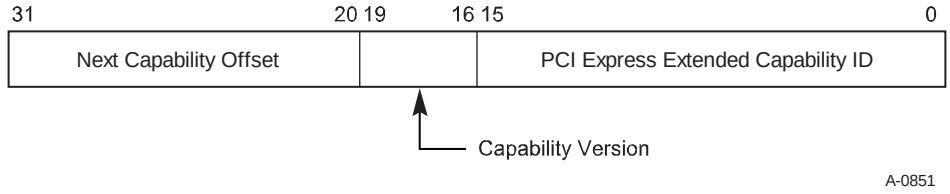


Figure G-8: PMUX Extended Capability Header

Table G-6: PMUX Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. The Extended Capability ID for the PMUX Extended Capability is 001Ah.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.	RO

Bit Location	Register Description	Attributes
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of capabilities. This offset is relative to the beginning of PCI compatible Configuration Space and thus must always be either 000h (for terminating the list of Capabilities) or greater than 0FFh.	RO

G.5.2. PMUX Capability Register (Offset 04h)

Figure G-9 details the allocation of fields in the PMUX Capability register. Table G-7 provides the respective bit definitions.

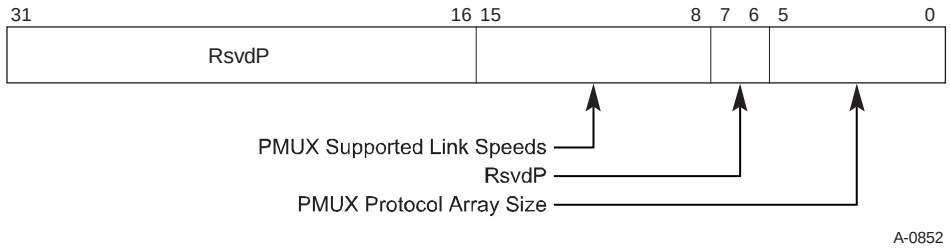


Figure G-9: PMUX Capability Register

Table G-7: PMUX Capability Register

Bit Location	Register Description	Attributes
5:0	**PMUX Protocol Array Size** – Indicates the size of this Function's PMUX Protocol Array. This field may be 0 to indicate that even though no protocols are supported, the Port will ignore all received PMUX Packets.	RO
7:6	**Reserved**	RsvdP

Bit Location	Register Description	Attributes
15:8	**PMUX Supported Link Speeds** – This field indicates the Link speed(s) where Protocol Multiplexing is supported. Each bit corresponds to a Link speed. If a bit is Set, Protocol Multiplexing is supported at that Link speed. If a bit is Clear, Protocol Multiplexing is not supported at that Link speed. Bit definitions are: Bit 8 2.5 GT/s Bit 9 5.0 GT/s Bit 10 8.0 GT/s Bit 11 16.0 GT/s Bits 15:12 RsvdP At least one Link speed must be supported (i.e., the field must be non-zero). A Port may support any combination of Link speeds. For example, this field could contain the value 0000 0100b indicating that Protocol Multiplexing is only supported at 8.0 GT/s. This field must not indicate support for Link speeds that are not supported by the Link (see Section 7.8.18). Note that this field indicates the Link speeds supported by Protocol Multiplexing for the Link. The Link speeds that a particular protocol supports and the mechanism used to report that information are protocol specific.	RO / RsvdP
31:16	**Reserved**	RsvdP

G.5.3. PMUX Control Register (Offset 08h)

Figure G-10 details the allocation of fields in the PMUX Control register. Table G-8 provides the respective bit definitions.

Channel *n* is enabled and available for use by the PMUX Protocol Layer when all of the following are true:

- The Channel *n* Assignment field is non-zero.
- The Channel *n* Assignment field is less than or equal to PMUX Protocol Array Size.
- The Channel *n* Assignment field indicates an implemented entry in the PMUX Protocol Array (see Section G.5.5).
- All of the PMUX Channel *n* Disabled bits are Clear (see Section G.5.4).

Otherwise, Channel *n* is disabled.

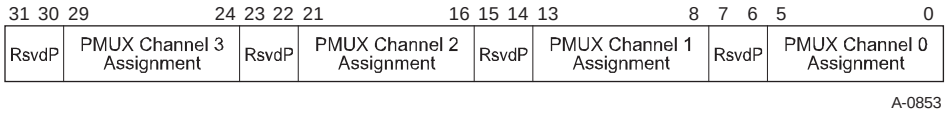


Figure G-10: PMUX Control Register

Table G-8: PMUX Control Register

Bit Location	Register Description	Attributes
5:0	**PMUX Channel 0 Assignment** – This field indicates the protocol assigned to PMUX Channel 0. If the field is 0h, no protocol is assigned. If the field is non-zero, it is the index in the PMUX Protocol Array of the protocol assigned to PMUX Channel 0. If PMUX Protocol Array Size is less than 63 (see Section G.5.2), unused upper bits of this field may be hardwired to 0b. If PMUX Protocol Array Size is 0, this entire field may be hardwired to 0. This field defaults to 0h.	RW
7:6	**Reserved**	RsvdP
13:8	**PMUX Channel 1 Assignment** – This field indicates the protocol assigned to PMUX Channel 1. If the field is 0h, no protocol is assigned. If the field is non-zero, it is the index in the PMUX Protocol Array of the protocol assigned to PMUX Channel 1. If PMUX Protocol Array Size is less than 63 (see Section G.5.2), unused upper bits of this field may be hardwired to 0b. If PMUX Protocol Array Size is 0, this entire field may be hardwired to 0. This field defaults to 0h.	RW
15:14	**Reserved**	RsvdP
21:16	**PMUX Channel 2 Assignment** – This field indicates the protocol assigned to PMUX Channel 2. If the field is 0h, no protocol is assigned. If the field is non-zero, it is the index in the PMUX Protocol Array of the protocol assigned to PMUX Channel 2. If PMUX Protocol Array Size is less than 63 (see Section G.5.2), unused upper bits of this field may be hardwired to 0b. If PMUX Protocol Array Size is 0, this entire field may be hardwired to 0. This field defaults to 0h.	RW
23:22	**Reserved**	RsvdP

Bit Location	Register Description	Attributes
29:24	**PMUX Channel 3 Assignment** – This field indicates the protocol assigned to PMUX Channel 3. If the field is 0h, no protocol is assigned. If the field is non-zero, it is the index in the PMUX Protocol Array of the protocol assigned to PMUX Channel 3. If PMUX Protocol Array Size is less than 63 (see Section G.5.2), unused upper bits of this field may be hardwired to 0b. If PMUX Protocol Array Size is 0, this entire field may be hardwired to 0. This field defaults to 0h.	RW
31:30	**Reserved**	RsvdP

G.5.4. PMUX Status Register (Offset 0Ch)

Figure G-11 details the allocation of fields in the PMUX Status register. Table G-9 provides the respective bit definitions.

Each channel has a set of Disabled bits. When Channel *n* Assignment field is non-zero, the Channel *n* Disabled bits reflect the error status of the channel. The following Disabled bits are defined:

- PMUX Channel *n* Disabled: Link Speed
- PMUX Channel *n* Disabled: Link Width
- PMUX Channel *n* Disabled: Protocol Specific

When there are multiple reasons for disabling a channel, an implementation may choose which reason(s) to report. For example, if a protocol needs bandwidth equivalent to x1 8.0 GT/s, when there is inadequate bandwidth (e.g., the Link is operating at x1 5.0 GT/s, x1 2.5 GT/s, or x2 2.5 GT/s), it could disable the PMUX Channel by indicating any or all of Disabled: Link Width, Disabled: Link Speed, or Disabled: Protocol Specific.

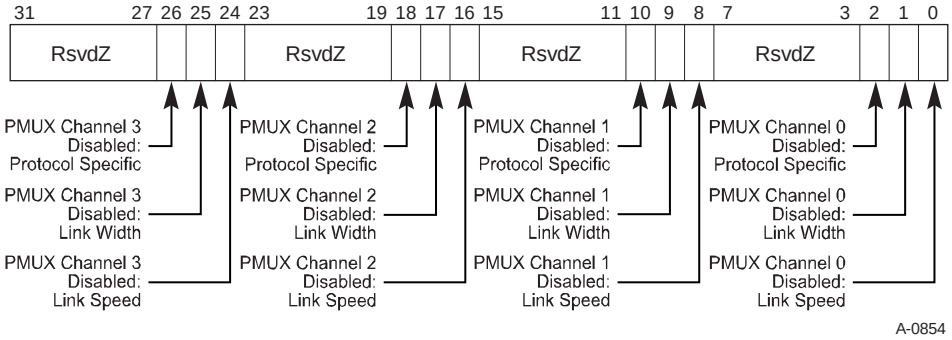


Figure G-11: PMUX Status Register

Table G-9: PMUX Status Register

Bit Location	Register Description	Attributes
0	**PMUX Channel 0 Disabled: Link Speed** – If Set, Channel 0 is disabled because the Current Link Speed (Section 7.8.8) is not supported by Protocol Multiplexing or by the protocol assigned to Channel 0. This bit is 0 when no protocol is assigned to Channel 0 (i.e., Channel 0 Control field is 0h).	RO
1	**PMUX Channel 0 Disabled: Link Width** – If Set, Channel 0 is disabled because the current Link Width is not supported by the protocol assigned to Channel 0. This bit is 0 when no protocol is assigned to Channel 0 (i.e., Channel 0 Assignment field is 0h).	RO
2	**PMUX Channel 0 Disabled: Protocol Specific** – If Set, Channel 0 is disabled for protocol specific reasons. This bit is 0 when no protocol is assigned to Channel 0 (i.e., Channel 0 Assignment field is 0h).	RO
7:3	**Reserved**	RsvdZ
8	**PMUX Channel 1 Disabled: Link Speed** – If Set, Channel 1 is disabled because the Current Link Speed (Section 7.8.8) is not supported by Protocol Multiplexing or by the protocol assigned to Channel 1. This bit is 0 when no protocol is assigned to Channel 1 (i.e., Channel 1 Assignment field is 0h).	RO
9	**PMUX Channel 1 Disabled: Link Width** – If Set, Channel 1 is disabled because the current Link Width is not supported by the protocol assigned to Channel 1. This bit is 0 when no protocol is assigned to Channel 1 (i.e. Channel 1 Assignment field is 0h).	RO
10	**PMUX Channel 1 Disabled: Protocol Specific** – If Set, Channel 1 is disabled for protocol specific reasons. This bit is 0 when no protocol is assigned to Channel 1 (i.e., Channel 1 Assignment field is 0h).	RO
15:11	**Reserved**	RsvdZ
16	**PMUX Channel 2 Disabled: Link Speed** – If Set, Channel 2 is disabled because the Current Link Speed (Section 7.8.8) is not supported by Protocol Multiplexing or by the assigned protocol. This bit is 0 when no protocol is assigned to Channel 2 (i.e., Channel 2 Assignment field is 0h).	RO

Bit Location	Register Description	Attributes
17	**PMUX Channel 2 Disabled: Link Width** – If Set, Channel 2 is disabled because the current Link Width is not supported by the assigned protocol. This bit is 0 when no protocol is assigned to Channel 2 (i.e., Channel 2 Assignment field is 0h).	RO
18	**PMUX Channel 2 Disabled: Protocol Specific** – If Set, Channel 2 is disabled for protocol specific reasons. This bit is 0 when no protocol is assigned to Channel 2 (i.e., Channel 2 Assignment field is 0h).	RO
23:19	**Reserved**	RsvdZ
24	**PMUX Channel 3 Disabled: Link Speed** – If Set, Channel 3 is disabled because the Current Link Speed (Section 7.8.8) is not supported by Protocol Multiplexing or by the assigned protocol. This bit is 0 when no protocol is assigned to Channel 3 (i.e., Channel 3 Assignment field is 0h).	RO
25	**PMUX Channel 3 Disabled: Link Width** – If Set, Channel 3 is disabled because the current Link Width is not supported by the assigned protocol. This bit is 0 when no protocol is assigned to Channel 3 (i.e., Channel 3 Assignment field is 0h).	RO
26	**PMUX Channel 3 Disabled: Protocol Specific** – If Set, Channel 3 is disabled for protocol specific reasons. This bit is 0 when no protocol is assigned to Channel 3 (i.e., Channel 3 Assignment field is 0h).	RO
32:27	**Reserved**	RsvdZ

G.5.5. PMUX Protocol Array (Offsets 10h Through 48h)

The PMUX Protocol Array consists of up to 63 entries. The size of the PMUX Protocol Array is indicated by the PMUX Protocol Array Size field (see Section G.5.2).

Figure G-12 details the allocation of fields in each PMUX Protocol Array entry. Table G-10 provides the respective bit definitions.

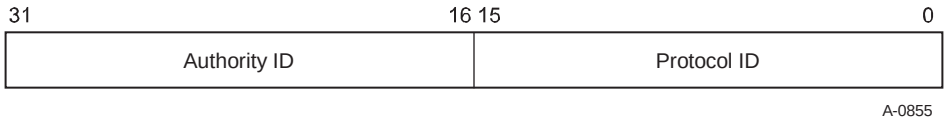


Figure G-12: PMUX Protocol Array Entry

Table G-10: PMUX Protocol Array Entry

Bit Location	Register Description	Attributes
15:0	**Protocol ID** – In conjunction with Authority ID designates a specific protocol and the mechanism by which that protocol is mapped onto Protocol Multiplexing.	RO
31:16	**Authority ID** – Designates the authority controlling the values used in the Protocol ID field. The Authority ID field contains a Vendor ID as assigned by the PCI-SIG.	RO

The value 0000 0000h indicates an unimplemented PMUX Protocol Array entry. The PMUX Protocol Array is indexed starting at 1.

PMUX Channel *n* is enabled and configured to support the protocol associated with PMUX Protocol Array entry at index *m* when the PMUX Channel *n* Assignment field contains the value *m* (see Section G.5.3).

Entries in the PMUX Protocol Array with the Authority ID value 1 (0001h) represent protocols that are defined by the PCI-SIG.

Duplicate Entries in the PMUX Protocol Array may be used to represent multiple instances of a particular protocol. This permits software control of the mapping between PMUX Channel ID and a specific instance of a protocol.

IMPLEMENTATION NOTE

Multiple Protocol Instances

A Link may have a single PMUX Protocol assigned to multiple PMUX Channels. Each PMUX Channel is assigned to a different instance of the protocol. Each instance of a protocol corresponds to an entry in the PMUX Protocol Array.

Consider a Port that supports two instances of protocol X. Two entries in the PMUX Protocol Array would indicate protocol X (indexes A and B for example). To assign instance A to PMUX Channel 0 and instance B to PMUX Channel 2, place the value A in the PMUX Channel 0 Assignment field and the value B in the PMUX Channel 2 Assignment field.

H

H. Flow Control Update Latency and ACK Update Latency Calculations

This appendix is informational only and should not be considered normative. Earlier revisions of this specification outlined the method used to calculate the Flow Control Update Latency and Ack Update Latency. This appendix describes the method used to derive the values and preserves the UpdateFactor and AckFactor values.

H.1. Flow Control Update Latency

Recommended Flow Control Update Latency is described in the Implementation Note in Section 2.6.1.2 entitled, FC Information Tracked by Receiver.

Table 2-45, Table 2-46 and Table 2-47 were simplified in the 4.0 revision of this specification and are reproduced in full below as Table H-1, Table H-2, and Table H-3 as they appeared in earlier revisions. Table 2-48 is not reproduced as the values are identical to Table 2-47.

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. The values are calculated as a function of the largest TLP payload size and Link width using the formula:

$$\frac{(Max_Payload_Size + TLPOverhead) * UpdateFactor}{LinkWidth} + InternalDelay$$

where:

Max_Payload_Size	The value in the Max_Payload_Size field of the Device Control register. For a multi-Function device, it is recommended that the smallest Max_Payload_Size setting across all Functions[117] is used.
TLPOverhead	Represents the additional TLP components which consume Link bandwidth (TLP Prefix, header, LCRC, framing Symbols) and is treated here as a constant value of 28 Symbols.
UpdateFactor	Represents the number of maximum size TLPs sent during the time between UpdateFC receptions, and is used to balance Link bandwidth efficiency and receive buffer sizes – the value varies according to Max_Payload_Size and Link width. Table H-1, Table H-2, and Table

[117] For ARI Devices, the Max_Payload_Size is determined solely by the setting in Function 0, and thus the settings in the other Functions should be ignored.

H-3 below include the UpdateFactor(UF).

LinkWidth — The operating width of the Link

InternalDelay — Represents the internal processing delays for received TLPs and transmitted DLLPs, and is treated here as a constant value of 19 Symbol Times for 2.5 GT/s mode operation, 70 Symbol Times for 5.0 GT/s mode operation, and 115 Symbol Times for 8.0 GT/s and 16.0 GT/s modes of operation.

Table H-1: Maximum UpdateFC Transmission Latency Guidelines for 2.5 GT/s Mode Operation by Link Width and Max Payload (Symbol Times)

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

Table H-2: Maximum UpdateFC Transmission Latency Guidelines for 5.0 GT/s Mode Operation by Link Width and Max Payload (Symbol Times)

		Link Operating Width						
		x1	**x2**	**x4**	**x8**	**x12**	**x16**	**x32**
Max_Payload_Size (bytes)	**128**	288 UF = 1.4	179 UF = 1.4	124 UF = 1.4	118 UF = 2.5	109 UF = 3.0	99 UF = 3.0	84 UF = 3.0
	256	467 UF = 1.4	268 UF = 1.4	169 UF = 1.4	158 UF = 2.5	141 UF = 3.0	123 UF = 3.0	96 UF = 3.0
	512	610 UF = 1.0	340 UF = 1.0	205 UF = 1.0	137 UF = 1.0	160 UF = 2.0	137 UF = 2.0	103 UF = 2.0
	1024	1122 UF = 1.0	596 UF = 1.0	333 UF = 1.0	201 UF = 1.0	245 UF = 2.0	201 UF = 2.0	135 UF = 2.0
	2048	2146 UF = 1.0	1108 UF = 1.0	589 UF = 1.0	329 UF = 1.0	416 UF = 2.0	329 UF = 2.0	199 UF = 2.0

	4096	4194 UF = 1.0	2132 UF = 1.0	1101 UF = 1.0	585 UF = 1.0	757 UF = 2.0	585 UF = 2.0	327 UF = 2.0

Table H-3: Maximum UpdateFC Transmission Latency Guidelines for 8.0 GT/s Operation by Link Width and Max Payload (Symbol Times)

		Link Operating Width						
		x1	**x2**	**x4**	**x8**	**x12**	**x16**	**x32**
Max_Payload_Size (bytes)	**128**	333 UF = 1.4	224 UF = 1.4	169 UF = 1.4	163 UF = 2.5	154 UF = 3.0	144 UF = 3.0	129 UF = 3.0
	256	512 UF = 1.4	313 UF = 1.4	214 UF = 1.4	203 UF = 2.5	186 UF = 3.0	168 UF = 3.0	141 UF = 3.0
	512	655 UF = 1.0	385 UF = 1.0	250 UF = 1.0	182 UF = 1.0	205 UF = 2.0	182 UF = 2.0	148 UF = 2.0
	1024	1167 UF = 1.0	641 UF = 1.0	378 UF = 1.0	246 UF = 1.0	290 UF = 2.0	246 UF = 2.0	180 UF = 2.0
	2048	2191 UF = 1.0	1153 UF = 1.0	634 UF = 1.0	374 UF = 1.0	461 UF = 2.0	374 UF = 2.0	244 UF = 2.0
	4096	4239 UF = 1.0	2177 UF = 1.0	1146 UF = 1.0	630 UF = 1.0	802 UF = 2.0	630 UF = 2.0	372 UF = 2.0

H.2. Ack Latency

The Maximum Ack Latency values are listed in Table 3-7, Table 3-8, Table 3-9 and Table 3-10 in Section 3.6.3.1 entitled, LCRC and Sequence Number Rules (TLP Receiver). Table 3-7, Table 3-8 and Table 3-9 were simplified in the 4.0 revision of this specification and are reproduced below as they appeared in earlier revisions including the AckFactor(AF) as Table H-4, Table H-5, and Table H-6. Table 3-10 is not reproduced as the values are identical to Table 3-9.

The Maximum Ack Latency limits are calculated using the formula:

$$\frac{(Max_Payload_Size + TLPOverhead) * AckFactor}{LinkWidth} + InternalDelay$$

where:

Max_Payload_Size is the value in the Max_Payload_Size field of the Device Control register. For ARI Devices, the Max_Payload_Size is determined solely by the setting in Function 0. For a non-ARI multi-Function device whose Max_Payload_Size settings are identical across all Functions, the common Max_Payload_Size setting must be used. For a non-ARI multi-Function device 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 smallest Max_Payload_Size setting across all Functions.

TLPOverhead — represents the additional TLP components which consume Link bandwidth (TLP Prefix, header, LCRC, framing Symbols) and is treated here as a constant value of 28 Symbols.

AckFactor — represents the number of maximum size TLPs which can be received before an Ack is sent, and is used to balance Link bandwidth efficiency and retry buffer size – the value varies according to Max_Payload_Size and Link width. Table H-4, Table H-5, and Table H-6 below include the AckFactor(AF).

LinkWidth — is the operating width of the Link.

InternalDelay — represents the internal processing delays for received TLPs and transmitted DLLPs, and is treated here as a constant value of 19 Symbol Times for 2.5 GT/s mode operation, 70 Symbol Times for 5.0 GT/s operation, and 115 Symbol Times for 8.0 GT/s and 16.0 GT/s operation.

Table H-4: Maximum Ack Latency Limit and AckFactor for 2.5 GT/s (Symbol Times)

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

Table H-5: Maximum Ack Transmission Latency Limit and AckFactor for 5.0 GT/s (Symbol Times)

		Link Operating Width						
		x1	**x2**	**x4**	**x8**	**x12**	**x16**	**x32**
Max_Payload_Size (bytes)	**128**	288 AF = 1.4	179 AF = 1.4	124 AF = 1.4	118 AF = 2.5	109 AF = 3.0	99 AF = 3.0	84 AF = 3.0
	256	467 AF = 1.4	268 AF = 1.4	169 AF = 1.4	158 AF = 2.5	141 AF = 3.0	123 AF = 3.0	96 AF = 3.0
	512	610 AF = 1.0	340 AF = 1.0	205 AF = 1.0	137 AF = 1.0	160 AF = 2.0	137 AF = 2.0	103 AF = 2.0
	1024	1122 AF = 1.0	596 AF = 1.0	333 AF = 1.0	201 AF = 1.0	245 AF = 2.0	201 AF = 2.0	135 AF = 2.0
	2048	2146 AF = 1.0	1108 AF = 1.0	589 AF = 1.0	329 AF = 1.0	416 AF = 2.0	329 AF = 2.0	199 AF = 2.0
	4096	4194 AF = 1.0	2132 AF = 1.0	1101 AF = 1.0	585 AF = 1.0	757 AF = 2.0	585 AF = 2.0	327 AF = 2.0

Table H-6: Maximum Ack Transmission Latency Limit and AckFactor for 8.0 GT/s (Symbol Times)

		Link Operating Width						
		x1	**x2**	**x4**	**x8**	**x12**	**x16**	**x32**
Max_Payload_Size (bytes)	**128**	333 AF = 1.4	224 AF = 1.4	169 AF = 1.4	163 AF = 2.5	154 AF = 3.0	144 AF = 3.0	129 AF = 3.0
	256	512 AF = 1.4	313 AF = 1.4	214 AF = 1.4	203 AF = 2.5	186 AF = 3.0	168 AF = 3.0	141 AF = 3.0
	512	655 AF = 1.0	385 AF = 1.0	250 AF = 1.0	182 AF = 1.0	205 AF = 2.0	182 AF = 2.0	148 AF = 2.0
	1024	1167 AF = 1.0	641 AF = 1.0	378 AF = 1.0	246 AF = 1.0	290 AF = 2.0	246 AF = 2.0	180 AF = 2.0
	2048	2191 AF = 1.0	1153 AF = 1.0	634 AF = 1.0	374 AF = 1.0	461 AF = 2.0	374 AF = 2.0	244 AF = 2.0
	4096	4239 AF = 1.0	2177 AF = 1.0	1146 AF = 1.0	630 AF = 1.0	802 AF = 2.0	630 AF = 2.0	372 AF = 2.0

Acknowledgements

The following persons were instrumental in the development of the PCI Express Base Specification:[118]

Chamath Abhayagunawardhana	Intel Corporation
Ryan Abraham	Oracle Corporation
Shiva Aditham	Intel Corporation
Hong Ahn	Xilinx
Shay Aisman	Mellanox Technologies, Ltd
Jasmin Ajanovic	Intel Corporation
Katsutoshi Akagi	NEC Corporation
Joe Allen	Tektronix Inc.
Michael W. Altmann	Intel Corporation
Taha Amiralli	Advanced Micro Devices, Inc.
Boon Ang	VMware Corporation
Sujith Arramreddy	ServerWorks, Inc.
Antonio Asaro	Advanced Micro Devices, Inc.
Yuval Avnon	Marvell Semiconductor, Inc.
Jay Avula	ServerWorks, Inc.
Brian Baldwin	Synopsys, Inc.
Jasper Balraj	Intel Corporation
Nat Barbiero	Advanced Micro Devices, Inc.
Phillip Barnes	Hewlett-Packard Company
Suparna Behera	LSI Logic Corporation
Joseph A. Bennett	Intel Corporation
Richard Bell	Advanced Micro Devices, Inc.
Stuart Berke	Hewlett-Packard Company
Harish Bharadwaj	LSI Logic Corporation
Ajay V. Bhatt	Intel Corporation
Gaurav Bhide	Advanced Micro Devices, Inc.
Harmeet Bhugra	IDT Corporation
Christiaan Bil	Intel Corporation
Bill Bissonette	Intel Corporation
Cass Blodget	Intel Corporation
Jeffrey D. Bloom	Intel Corporation
Mahesh Bohra	IBM Corporation

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

Naveen Bohra	Intel Corporation
Karl Bois	Hewlett Packard Enterprise
Austin Bolen	Dell Inc.
Jerry Bolen	Intel Corporation
Michele Bologna	Synopsys, Inc.
Wil de Bont	National Instruments
David Bouse	Intel Corporation
Hicham Bouzekri	ST-Ericsson
Gavin Bowlby	QLogic Corporation
Suri Brahmaroutu	Dell Inc.
Dave Brown	Integrated Device Technology
Tory Brown	Intel Corporation
Mike Brownell	Intel Corporation
Bala Cadambi	Intel Corporation
John Calvin	VTM
John Calvin	Tektronix, Inc.
Jun Cao	Synopsys, Inc.
Gord Caruk	Advanced Micro Devices, Inc.
Paul Cassidy	Synopsys, Inc.
James Chapple	Intel Corporation
Kabitha Chaturvedula	LSI Logic Corporation
Yogesh Chaudhary	Mentor Graphics
Santanu Chaudhuri	Intel Corporation
Rajinder Cheema	LSI Logic Corporation
Albert Chen	LSI Logic Corporation
Chih-Cheh Chen	Intel Corporation
Jian Chen	Parade
Qunwei Chen	Advanced Micro Devices, Inc.
Tony Chen	QLogic Corporation
Jonathan Cheung	Advanced Micro Devices, Inc.
Yorick Cho	Advanced Micro Devices, Inc.
Dickson Chow	Advanced Micro Devices, Inc.
Gene Chui	IDT Corporation
Shawn Clayton	Emulex Corporation
Mark Clements	IBM Corporation
Debra Cohen	Intel Corporation
Eric Combs	LSI Logic Corporation
Brad Congdon	Intel Corporation

Mike Converse	IDT Corporation
Justin Coppin	Hewlett-Packard Company
Joe Cowan	Hewlett-Packard Enterprise
Carrie Cox	IBM Corporation
H. Clay Cranford	IBM Corporation
Kenneth C. Creta	Intel Corporation
Pamela Cristo	Emulex Corporation
Sanjay Dabral	Intel Corporation
Eric Dahlen	Intel Corporation
Tugrul Daim	Intel Corporation
Ron Dammann	Intel Corporation
Sumit Das	Texas Instruments Incorporated
Debendra Das Sharma	Intel Corporation
Nicole Daugherty	Intel Corporation
Glen Dearth	Advanced Micro Devices, Inc.
Eric DeHaemer	Intel Corporation
Fei Deng	Intel Corporation
Dan DeSetto	Intel Corporation
Karishma Dhruv	LSI Logic Corporation
Gary Dick	Cadence Design Systems, Inc
Robert Dickson	Oracle Corporation
Bob Divivier	IDT Corporation
Hormoz Djahanshahi	Microsemi Corporation
Daniel Dreps	IBM Corporation
Ken Drottar	Intel Corporation
Dave Dunning	Intel Corporation
Rick Eads	Keysight
Rick Eads	Agilent Technologies, Inc.
Gregory L. Ebert	Intel Corporation
Yaron Elboim	Intel Corporation
Bassam Elkhoury	Intel Corporation
Salem Emara	Advanced Micro Devices, Inc.
Mike Engbretson	Tektronix, Inc.
Ohad Falik	Intel Corporation
David Fair	Intel Corporation
Blaise Fanning	Intel Corporation
John Feehrer	Oracle Corporation
Wes Ficken	GLOBALFOUNDRIES Inc.

Joshua Filliater	LSI Logic Corporation
Michael Fleischer-Reumann	Agilent Technologies, Inc.
Pelle Fornberg	Intel Corporation
Jeff Fose	Emulex Corporation
Jim Foster	IBM Corporation
Mike Foxcroft	Advanced Micro Devices, Inc.
Douglas Freimuth	IBM Corporation
Daniel S. Froelich	Intel Corporation
SivaPrasad Gadey	Intel Corporation
Yoni Galezer	Mellanox Technologies, Ltd
Eric Geisler	Intel Corporation
Gordon Getty	Agilent Technologies, Inc.
Gregory Geyfman	Intel Corporation
Stefano Giaconi	Intel Corporation
Steve Glaser	NVIDIA Corporation
Thorsten Goetzelmann	Keysight
Marc A. Goldschmidt	Intel Corporation
Dean Gonzales	Advanced Micro Devices, Inc.
Alan Goodrum	Hewlett-Packard Company
Robert Gough	Intel Corporation
Lucian Gozu	Neterion Corporation
Ilya Granovsky	IBM Corporation
Brien Gray	Intel Corporation
Richard Greene	Intel Corporation
Stephen Greenwood	ATI Technologies Inc.
Michael J. Greger	Intel Corporation
Buck Gremel	Intel Corporation
Andrew Gruber	Advanced Micro Devices, Inc.
George Gruber	Qualcomm
Vijay Gudur	LSI Logic Corporation
Dale Gulick	Advanced Micro Devices, Inc.
Liping Guo	Marvell Semiconductor, Inc.
Mickey Gutman	Intel Corporation
Clifford D. Hall	Intel Corporation
Stephen H. Hall	Intel Corporation
Joseph Hamel	IBM Corporation
Ken Haren	Intel Corporation
David Harriman	Intel Corporation

Hiromitsu Hashimoto	NEC Corporation
George R. Hayek	Intel Corporation
Wenmu He	Texas Instruments Incorporated
Matt Hendrick	Intel Corporation
Hamish Hendry	Cadence Design Systems, Inc
Michael Herz	Blackberry
Bent Hessen-Schmidt	Tektronix, Inc.
Hanh Hoang	Intel Corporation
Joe Hock	Intel Corporation
Michael Hopgood	NVIDIA Corporation
Dorcas Hsia	NVIDIA Corporation
James Huang	Advanced Micro Devices, Inc.
Robert Huang	NVIDIA Corporation
Yifan Huang	Amphenol-FCI
Kamlesh Hujwant	Advanced Micro Devices, Inc.
Mark Hummel	Advanced Micro Devices, Inc.
Mikal Hunsaker	Intel Corporation
Joaquin Ibanez	Synopsys, Inc.
Yasser Ibrahim	Microsoft Corporation
Franko Itay	Intel Corporation
Carl Jackson	Hewlett-Packard Company
David R. Jackson	Intel Corporation
Praveen Jain	NVIDIA Corporation
Martin James	Cadence Design Systems, Inc
Duane January	Intel Corporation
James E. Jaussi	Intel Corporation
Mike Jenkins	LSI Logic Corporation
Peter Jenkins	GLOBALFOUNDRIES Inc.
Hong Jiang	Intel Corporation
Viek Joshi	Intel Corporation
Dave Kaffine	Oracle Corporation
David Kahmayhew	Oracle Corporation
Ravi Kammaje	LSI Logic Corporation
Girish Karanam	LSI Logic Corporation
Kapil Karkra	Intel Corporation
Chad Kendall	Broadcom Corporation
Mukund Kharti	Dell Computer Corporation
David Kimble	Texas Instruments Incorporated

Lavi Koch	Mellanox Technologies, Ltd
Mohammad Kolbehdari	Intel Corporation
Abhimanyu Kolla	Intel Corporation
Ganesh Kondapuram	Intel Corporation
Kwok Kong	IDT Corporation
Michael Krause	Hewlett-Packard Enterprise
Gopi Krishnamurthy	Cadence Design Systems
Sreenivas Krishnan	NVIDIA Corporation
Kumaran Krishnasamy	Broadcom Corporation
Steve Krooswyk	Samtec
Steven K. Krooswyk	Intel Corporation
Manjari Kulkarni	Intel Corporation
Akhilesh Kumar	Intel Corporation
Mohan J. Kumar	Intel Corporation
Richard Kunze	Intel Corporation
Hugh Kurth	Oracle Corporation
Seh Kwa	Intel Corporation
Ricky Lai	Hewlett Packard Enterprise
Sunny Lam	Intel Corporation
Dave Landsman	Sandisk
Brian Langendorf	NVIDIA Corporation
Raymond R. Law	Intel Corporation
Dror Lazar	Intel Corporation
Brian L'Ecuyer	Agilent Technologies, Inc.
Beomtek Lee	Intel Corporation
Clifford D. Lee	Intel Corporation
David M. Lee	Intel Corporation
Edward Lee	Advanced Micro Devices, Inc.
Moshe Leibowitz	IBM Corporation
Tony L. Lewis	Intel Corporation
Mike Li	Wavecrest Corporation
Paul Li	Pericom Semiconductor Corporation
Stephen Li	Texas Instruments Incorporated
Tao Liang	Intel Corporation
Andrew K. Lillie	Intel Corporation
Wendy Liu	Advanced Micro Devices, Inc.
Jeff Lukanc	IDT Corporation
Jeffrey Lu	IBM Corporation

Betty Luk	Advanced Micro Devices, Inc.
Ngoc Luu	Advanced Micro Devices, Inc.
Kenneth Ma	Broadcom Corporation
Stephen Ma	Advanced Micro Devices, Inc.
Zorik Machulsky	IBM Corporation
Mallik Mahalingam	VMware, Inc.
Kevin Main	Texas Instruments Incorporated
Steven Makow	IBM Corporation
Tony Mangefeste	Microsoft Corporation
Nilange Manisha	Intel Corporation
Steve Manning	Advanced Micro Devices, Inc.
Gold Mao	VIA Technologies, Inc.
Jarek Marczewski	Advanced Micro Devices, Inc.
Mark Marlett	LSI Logic Corporation
Alberto Martinez	Intel Corporation
Andrew Martwick	Intel Corporation
Paul Mattos	GLOBALFOUNDRIES Inc.
Robert A. Mayer	Intel Corporation
David Mayhew	(Stargen, Inc.) Advanced Micro Devices, Inc.
Mohiuddin Mazumder	Intel Corporation
Mehdi Mechaik	Cadence Design Systems
Mehdi M. Mechaik	NVIDIA Corporation
Pranav H. Mehta	Intel Corporation
Vishal Mehta	NVIDIA Corporation
Richard Mellitz	Intel Corporation
Cindy Merkin	Dell Computer Corporation
Slobodan Milijevic	Microsemi Corporation
Dennis Miller	Intel Corporation
Jason Miller	Oracle Corporation
Robert J. Miller	Intel Corporation
Michael Mirmak	Intel Corporation
Suneel Mitbander	Intel Corporation
Mohammad Mobin	NVIDIA Corporation
Daniel Moertl	IBM Corporation
Lee Mohrmann	National Instruments Corporation
Puga Nathal Moises	Intel Corporation
Richard Moore	QLogic Corporation
Wayne Moore	Intel Corporation

Douglas R. Moran	Intel Corporation
Terry Morris	Hewlett-Packard Enterprise
Jeff C. Morriss	Intel Corporation
Linna Mu	Avago Technologies
Sridhar Muthrasanallur	Intel Corporation
Suresh Babu M. V.	LSI Logic Corporation
Gautam V. Naik	LSI Logic Corporation
Mohan K. Nair	Intel Corporation
Mukund Narasimhan	Intel Corporation
Alon Naveh	Intel Corporation
Ramin Neshati	Intel Corporation
Surena Neshvad	Intel Corporation
Andy Ng	IDT Corporation
Manisha Nilange	Intel Corporation
Hajime Nozaki	NEC Corporation
Kugao Ohuchi	NEC Corporation
Vitor Oliveira	Synopsys, Inc.
Olufemi Oluwafemi	Intel Corporation
Takuya Omura	Synopsys, Inc.
Peter Onufryk	Integrated Device Technology
Mike Osborn	Advanced Micro Devices, Inc.
Jake Oshins	Microsoft Corporation
Randy Ott	Intel Corporation
Jonathan Owen	Advanced Micro Devices, Inc.
Ali Oztaskin	Intel Corporation
David Pabisz	Oracle Corporation
Shreeram Palghat	Intel Corporation
Jim Panian	Qualcomm
Marc Pegolotti	ServerWorks, Inc.
Henry Peng	Intel Corporation
Akshay Pethe	Intel Corporation
Chris Pettey	NextIO, Inc.
Tien Pham	Hewlett Packard Enterprise
Lu-vong Phan	Intel Corporation
Tony Pierce	Microsoft Corporation
Edmund Poh	Molex, Inc.
Harshit Poladia	Intel Corporation
Jim Prijic	Intel Corporation

Dave Puffer	Intel Corporation
Duane Quiet	Intel Corporation
Jeffrey D. Rabe	Intel Corporation
Andy Raffman	Microsoft Corporation
Kianoush Rahbar	Intel Corporation
Guru Rajamani	Intel Corporation
Ramesh Raman	LSI Logic Corporation
Adee Ran	Intel Corporation
Todd Rasmus	IBM Corporation
Renato Recio	IBM Corporation
Ramakrishna Reddy	LSI Logic Corporation
Jack Regula	PLX Technology, Inc.
Dick Reohr	Intel Corporation
Curtis Ridgeway	LSI Logic Corporation
Dwight Riley	Hewlett-Packard Enterprise
Yoav Rozenbert	Mellanox Technologies, Ltd
Chris Runhaar	NVIDIA Corporation
Rajanataraj S.	LSI Logic Corporation
Devang Sachdev	NVIDIA Corporation
Gene Saghi	Broadcom Limited
Rajesh Sankaran	Intel Corporation
Bill Sauber	Dell Computer Corporation
Joseph Schachner	Teledyne LeCroy
Mike Schaecher	Oracle Corporation
Joe Schaefer	Intel Corporation
Michael Scheid	Broadcom Corporation
Daren Schmidt	Intel Corporation
Mark Schmisseur	Intel Corporation
Richard Schober	NVIDIA Corporation
Zale Schoenborn	Intel Corporation
Rick Schuckle	Dell Computer Corporation
Richard Schumacher	Hewlett-Packard Enterprise
Jeremiah Schwartz	Intel Corporation
Tudor Secasiu	Intel Corporation
Kazunori Seki	Synopsys, Inc.
Oren Sela	Mellanox Technologies, Ltd
Kevin Senohrabek	Advanced Micro Devices, Inc.
Kalev Sepp	Tektronix, Inc.

Prashant Sethi	Intel Corporation
Ankur Shah	Intel Corporation
Vasudevan Shanmugasundaram	Intel Corporation
Wesley Shao	Oracle Corporation
Prateek Sharma	LSI Logic Corporation
Charlie Shaver	Hewlett-Packard Company
Robert Sheldon	Oracle Corporation
John Sheplock	IBM Corporation
Milton Shih	Oracle Corporation
Mark Shillingburg	Agilent Technologies
Oren Shirak	Mellanox Technologies, Ltd
Sarvesh Shrivastava	Qualcomm
Bill Simms	NVIDIA Corporation
Vinita Singhal	NVIDIA Corporation
Dan Slocombe	Cadence Design Systems, Inc
Brian Small	Northwest Logic, Inc.
George Smith	Microsoft Corporation
Shamnad SN	LSI Logic Corporation
Rick Sodke	PMC-Sierra
Gary Solomon	Intel Corporation
Richard Solomon	Synopsys, Inc.
Gao Song	IDT Corporation
Brad Sonksen	Cavium Inc.
Walter Soto	Broadcom Corporation
Fulvio Spagna	Intel Corporation
Jason Squire	Molex, Inc.
Patrick Stabile	Oracle Corporation
Sean O. Stalley	Intel Corporation
Stan Stanski	IBM Corporation
Hermann Stehling	Bitifeye Digital Test Solutions
John Stonick	Synopsys, Inc.
Ron Swartz	Intel Corporation
Tim Symons	PMC-Sierra
Miki Takahashi	NEC Corporation
Gerry Talbot	Advanced Micro Devices, Inc.
Anthony Tam	Advanced Micro Devices, Inc.
Wanru Tao	Oracle Corporation
Mark Taylor	NVIDIA Corporation

Matthew Tedone	LSI Logic Corporation
Grigori Temkine	Advanced Micro Devices, Inc.
Peter Teng	NEC Corporation
Bruce A. Tennant	Intel Corporation
Larry Tesdall	Cavium Inc.
Andrew Thornton	Microsoft Corporation
Steven Thurber	IBM Corporation
Mike Tobin	Intel Corporation
Ben Toby	Hewlett Packard Enterprise
Duke Tran	Broadcom
Tan V. Tran	Intel Corporation
Alok Tripathi	Intel Corporation
William Tsu	NVIDIA Corporation
Alexander Umansky	Huawei Technologies Co.
Chris Van Beek	Intel Corporation
Arie van der Hoeven	Microsoft Corporation
Andrew Vargas	Intel Corporation
Robertson Velez	ATI Technologies Inc.
Archana Vasudevan	LSI Logic Corporation
Kiran Velicheti	Intel Corporation
Balaji Vembu	Intel Corporation
Gary Verdun	Dell Computer Corporation
Sushil Verghese	Broadcom Corporation
Divya Vijayaraghavan	Altera Corporation
Ravindra Viswanath	LSI Logic Corporation
Pete D. Vogt	Intel Corporation
Andrew Volk	Intel Corporation
Mahesh Wagh	Intel Corporation
Clint Walker	Intel Corporation
Davis Walker	Microsoft Corporation
Hui Wang	IDT Corporation
Kai A. Wang	Intel Corporation
Leo Warmuth	NXP Semiconductors
Dan Wartski	Intel Corporation
Neil Webb	Cadence Design Systems, Inc.
Eric Wehage	Intel Corporation
Dong Wei	Hewlett-Packard Company
Ron Weimer	QLogic Corporation

Amir Wiener	Intel Corporation
Marc Wells	Intel Corporation
Rob Wessel	Hewlett-Packard Company
Stephen F. Whalley	Intel Corporation
Bryan White	Intel Corporation
Paul Whittemore	Oracle Corporation
Amir Wiener	Intel Corporation
Timothy Wig	Intel Corporation
Craig Wiley	Parade
Jim Williams	Emulex Corporation
Theodore L. Willke	Intel Corporation
Dawn Wood	Intel Corporation
John Wood	Emulex Corporation
David Woodral	QLogic Corporation
David Wooten	Microsoft Corporation
Zhi Wong	Altera Corporation
David Wormus	LSI Logic Corporation
David Wu	Broadcom Corporation
William Wu	Broadcom Corporation
Zuoguo Wu	Intel Corporation
Liu Xin	IDT Corporation
Dan Yaklin	Texas Instruments Incorporated
Howard Yan	Intel Corporation
Jane Yan	Oracle Corporation
Al Yanes	IBM Corporation
Gin Yee	Advanced Micro Devices, Inc.
Ahmed Younis	Xilinx, Inc.
Chi-Ho Yue	NVIDIA Corporation
Wayne Yun	Advanced Micro Devices, Inc.
Dave Zenz	Dell Computer Corporation
Shubing Zhai	IDT Corporation
Wei Zhang	Broadcom
Bo Zhang	Intel Corporation
Guoqing Zhang	Cadence Design Systems, Inc.
Kevin Ziegler	Tektronix Inc.