PCI Express
Base Specification
Revision 1.0

April 29, 2002

REVISION	REVISION HISTORY	DATE
1.0	Initial release.	4/29/02

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

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

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

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

Technical Support
techsupp@pcisig.com

DISCLAIMER

This draft Specification is being provided to you for review purposes pursuant to Article 15.2 of the Bylaws of PCI-SIG. This draft Specification is subject to amendment until it is officially adopted by the Board of Directors of PCI-SIG. The Board of Directors may, at its discretion, initiate additional review periods, in which case you will be notified of the same. Pursuant to Article 14 of the Bylaws, this draft Specification is to be considered PCI-SIG Confidential until adopted by the Board of Directors.

All product names are trademarks, registered trademarks, or servicemarks of their respective owners.

Copyright © 2002 PCI-SIG

Contents

PREFACE **17**

OBJECTIVE OF THE SPECIFICATION 18
DOCUMENT ORGANIZATION 18
DOCUMENTATION CONVENTIONS 19
TERMS AND ABBREVIATIONS 20
REFERENCE DOCUMENTS 25

1. INTRODUCTION **27**

1.1. A THIRD GENERATION I/O INTERCONNECT 27
1.2. PCI EXPRESS LINK 29
1.3. PCI EXPRESS FABRIC TOPOLOGY 30
1.3.1. Root Complex *31*
1.3.2. Endpoints *32*
1.3.3. Switch *33*
1.3.4. PCI Express-PCI Bridge *34*
1.4. PCI EXPRESS FABRIC TOPOLOGY CONFIGURATION 34
1.5. PCI EXPRESS LAYERING OVERVIEW 35
1.5.1. Transaction Layer *36*
1.5.2. Data Link Layer *36*
1.5.3. Physical Layer *37*
1.5.4. Layer Functions and Services *37*
1.6. ADVANCED PEER-TO-PEER COMMUNICATION OVERVIEW 41

2. TRANSACTION LAYER SPECIFICATION **43**

2.1. TRANSACTION LAYER OVERVIEW 43
2.2. ADDRESS SPACES, TRANSACTION TYPES, AND USAGE 44
2.2.1. Memory Transactions *44*
2.2.2. I/O Transactions *45*
2.2.3. Configuration Transactions *45*
2.2.4. Message Transactions *45*
2.3. PACKET FORMAT OVERVIEW 47
2.4. TRANSACTION DESCRIPTOR 48
2.4.1. Overview *48*
2.4.2. Transaction Descriptor –Transaction ID Field *48*
2.4.3. Transaction Descriptor – Attributes Field *50*
2.4.4. Transaction Descriptor – Traffic Class Field *51*
2.5. TRANSACTION ORDERING 52
2.6. VIRTUAL CHANNEL (VC) MECHANISM 56
2.6.1. Virtual Channel Identification (VC ID) *58*
2.6.2. VC Support Options *58*
2.6.3. TC to VC Mapping *59*

2.6.4. VC and TC Rules ... 60
2.7. TRANSACTION LAYER PROTOCOL - PACKET DEFINITION AND HANDLING ... 61
2.7.1. Transaction Layer Packet Definition Rules ... 61
2.7.2. TLP Digest Rules ... 65
2.7.3. TLPs with Data Payloads - Rules ... 66
2.7.4. Requests ... 67
2.7.5. Completions ... 76
2.7.6. Handling of Received TLPs ... 78
2.8. MESSAGES ... 88
2.8.1. Baseline Messages ... 88
2.8.2. Advanced Switching Support Message Group ... 98
2.9. ORDERING AND RECEIVE BUFFER FLOW CONTROL ... 99
2.9.1. Overview and Definitions ... 99
2.9.2. Flow Control Rules ... 100
2.10. DATA INTEGRITY ... 109
2.10.1. Introduction ... 109
2.10.2. ECRC Rules ... 109
2.11. ERROR FORWARDING ... 113
2.11.1. Error Forwarding Usage Model ... 113
2.11.2. Rules For Use of Data Poisoning ... 114
2.12. COMPLETION TIMEOUT MECHANISM ... 114
2.13. TRANSACTION LAYER BEHAVIOR IN DL_DOWN STATUS ... 115
2.14. TRANSACTION LAYER BEHAVIOR IN DL_UP STATUS ... 116

3. DATA LINK LAYER SPECIFICATION ... 117

3.1. DATA LINK LAYER OVERVIEW ... 117
3.2. DATA LINK CONTROL AND MANAGEMENT STATE MACHINE ... 119
3.2.1. Data Link Control and Management State Machine Rules ... 120
3.3. FLOW CONTROL INITIALIZATION PROTOCOL ... 121
3.3.1. Flow Control Initialization State Machine Rules ... 123
3.4. DATA LINK LAYER PACKETS (DLLPS) ... 125
3.4.1. Data Link Layer Packet Rules ... 125
3.5. DATA INTEGRITY ... 130
3.5.1. Introduction ... 130
3.5.2. LCRC, Sequence Number, and Retry Management (TLP Transmitter) ... 130
3.5.3. LCRC and Sequence Number (TLP Receiver) ... 142

4. PHYSICAL LAYER SPECIFICATION ... 149

4.1. INTRODUCTION ... 149
4.2. LOGICAL SUB-BLOCK ... 149
4.2.1. Symbol Encoding ... 150
4.2.2. Framing and Application of Symbols to Lanes ... 153
4.2.3. Data Scrambling ... 156
4.2.4. Link Initialization and Training ... 157
4.2.5. Link Training and Status State Machine (LTSSM) ... 180
4.2.6. Link Training and Status State Descriptions ... 183
4.2.7. Clock Tolerance Compensation ... 195

4.2.8. Compliance Pattern ... 197
4.3. ELECTRICAL SUB-BLOCK ... 198
4.3.1. Electrical Sub-Block Requirements ... 198
4.3.2. Electrical Signal Specifications ... 201
4.3.3. Differential Transmitter (Tx) Output Specifications ... 206
4.3.4. Differential Receiver (Rx) Input Specifications ... 211

5. SOFTWARE INITIALIZATION AND CONFIGURATION ... 215

5.1. CONFIGURATION TOPOLOGY ... 215
5.2. PCI EXPRESS CONFIGURATION MECHANISMS ... 216
5.2.1. PCI 2.3 Compatible Configuration Mechanism ... 217
5.2.2. PCI Express Enhanced Configuration Mechanism ... 218
5.2.3. Root Complex Register Block ... 218
5.3. CONFIGURATION TRANSACTION RULES ... 219
5.3.1. Device Number ... 219
5.3.2. Configuration Transaction Addressing ... 219
5.3.3. Configuration Request Routing Rules ... 220
5.3.4. Generating PCI Special Cycles using PCI Configuration Mechanism #1 221
5.4. CONFIGURATION REGISTER TYPES ... 221
5.5. PCI-COMPATIBLE CONFIGURATION REGISTERS ... 222
5.5.1. Type 0/1 Common Configuration Space ... 223
5.5.2. Type 0 Configuration Space Header ... 228
5.5.3. Type 1 Configuration Space Header ... 229
5.6. PCI POWER MANAGEMENT CAPABILITY STRUCTURE ... 232
5.7. MSI CAPABILITY STRUCTURE ... 234
5.8. PCI EXPRESS CAPABILITY STRUCTURE ... 234
5.8.1. PCI Express Capability List Register (Offset 00h) ... 235
5.8.2. PCI Express Capabilities Register (Offset 02h) ... 235
5.8.3. Device Capabilities Register (Offset 04h) ... 237
5.8.4. Device Control Register (Offset 08h) ... 241
5.8.5. Device Status Register (Offset 0Ah) ... 244
5.8.6. Link Capabilities Register (Offset 0Ch) ... 246
5.8.7. Link Control Register (Offset 10h) ... 248
5.8.8. Link Status Register (Offset 12h) ... 250
5.8.9. Slot Capabilities Register (Offset 14h) ... 251
5.8.10. Slot Control Register (Offset 18h) ... 253
5.8.11. Slot Status Register (Offset 1Ah) ... 255
5.8.12. Root Control Register (Offset 1Ch) ... 256
5.8.13. Root Status Register (Offset 20h) ... 257
5.9. PCI EXPRESS EXTENDED CAPABILITIES ... 258
5.9.1. Extended Capabilities in Configuration Space ... 259
5.9.2. Extended Capabilities in the Root Complex Register Block ... 259
5.9.3. PCI Express Enhanced Capability Header ... 259
5.10. ADVANCED ERROR REPORTING CAPABILITY ... 260
5.10.1. Advanced Error Reporting Enhanced Capability Header (Offset 00h) ... 261
5.10.2. Uncorrectable Error Status Register (Offset 04h) ... 262

5.10.3. Uncorrectable Error Mask Register (Offset 08h) 263
5.10.4. Uncorrectable Error Severity Register (Offset 0Ch) 264
5.10.5. Correctable Error Status Register (Offset 10h) 265
5.10.6. Correctable Error Mask (Offset 14h) 265
5.10.7. Advanced Error Capabilities and Control Register (Offset 18h) 266
5.10.8. Header Log Register (Offset 1Ch) 267
5.10.9. Root Error Command Register (Offset 2Ch) 268
5.10.10. Root Error Status Register (Offset 30h) 269
5.10.11. Error Source Identification Register (Offset 34h) 270
5.11. VIRTUAL CHANNEL CAPABILITY 271
5.11.1. Virtual Channel Enhanced Capability Header 272
5.11.2. Port VC Capability Register 1 273
5.11.3. Port VC Capability Register 2 275
5.11.4. Port VC Control Register 276
5.11.5. Port VC Status Register 277
5.11.6. VC Resource Capability Register 277
5.11.7. VC Resource Control Register 279
5.11.8. VC Resource Status Register 281
5.11.9. VC Arbitration Table 282
5.11.10. Port Arbitration Table 283
5.12. DEVICE SERIAL NUMBER CAPABILITY 285
5.12.1. Device Serial Number Enhanced Capability Header (Offset 00h) 285
5.12.2. Serial Number Register (Offset 04h) 286
5.13. POWER BUDGETING CAPABILITY 287
5.13.1. Power Budgeting Enhanced Capability Header (Offset 00h) 287
5.13.2. Data Select Register (Offset 04h) 288
5.13.3. Data Register (Offset 08h) 289
5.13.4. Power Budget Capability Register (Offset 0Ch) 291

6. POWER MANAGEMENT 293

6.1. OVERVIEW 293
6.1.1. Statement of Requirements 294
6.2. LINK STATE POWER MANAGEMENT 294
6.3. PCI-PM SOFTWARE COMPATIBLE MECHANISMS 299
6.3.1. Device Power Management States (D-States) of a Function 299
6.3.2. PM Software Control of the Link Power Management State 302
6.3.3. Power Management Event Mechanisms 307
6.4. NATIVE PCI EXPRESS POWER MANAGEMENT MECHANISMS 316
6.4.1. Active-State Power Management 316
6.5. AUXILIARY POWER SUPPORT 332
6.5.1. Auxiliary Power Enabling 332
6.6. POWER MANAGEMENT SYSTEM MESSAGES AND DLLPS 333
6.6.1. Power Management System Messages 333
6.6.2. Power Management DLLPs 334

7. PCI EXPRESS SYSTEM ARCHITECTURE 335

7.1. INTERRUPT SUPPORT 335

7.1.1. Rationale for PCI Express Interrupt Model ... 335
7.1.2. PCI Compatible INTx Emulation ... 336
7.1.3. INTx Emulation Software Model ... 336
7.1.4. Message Signaled Interrupt (MSI) Support ... 336
7.1.5. MSI Software Model ... 337
7.1.6. PME Support ... 337
7.1.7. PME Software Model ... 338
7.1.8. PME Routing Between PCI Express and PCI Hierarchies ... 338
7.2. ERROR SIGNALING AND LOGGING ... 338
7.2.1. Scope ... 338
7.2.2. Error Classification ... 339
7.2.3. Error Signaling ... 340
7.2.4. Error Logging ... 343
7.2.5. Error Listing and Rules ... 344
7.2.6. Real and Virtual PCI Bridge Error Handling ... 346
7.3. VIRTUAL CHANNEL SUPPORT ... 347
7.3.1. Introduction and Scope ... 347
7.3.2. Supported TC/VC Configurations ... 348
7.3.3. VC Arbitration ... 350
7.3.4. Isochronous Support ... 356
7.4. DEVICE SYNCHRONIZATION STOP MECHANISM ... 359
7.5. LOCKED TRANSACTIONS ... 360
7.5.1. Introduction ... 360
7.5.2. Initiation and Propagation of Locked Transactions - Rules ... 360
7.5.3. Switches and Lock - Rules ... 361
7.5.4. PCI Express/PCI Bridges and Lock - Rules ... 362
7.5.5. Root Complex and Lock - Rules ... 362
7.5.6. Legacy Endpoints ... 363
7.5.7. PCI Express Endpoints ... 363
7.6. PCI EXPRESS RESET -RULES ... 363
7.7. PCI EXPRESS NATIVE HOT PLUG SUPPORT ... 366
7.7.1. PCI Express Hot Plug Usage Model ... 366
7.7.2. Event Behavior ... 371
7.7.3. Registers Grouped by Device Association ... 371
7.7.4. Messages ... 376
7.7.5. PCI Express Hot Plug Interrupt/Wake Signal Logic ... 377
7.7.6. The Operating System Hot Plug Method ... 379
7.8. POWER BUDGETING CAPABILITY ... 380
7.8.1. System Power Budgeting Process Recommendations ... 380
7.9. SLOT POWER LIMIT CONTROL ... 381

A. ISOCHRONOUS APPLICATIONS AND SUPPORT ... 383

A.1. INTRODUCTION ... 383
A.2. ISOCHRONOUS CONTRACT AND CONTRACT PARAMETERS ... 385
A.2.1. Isochronous Time Period and Isochronous Virtual Timeslot ... 386
A.2.2. Isochronous Payload Size ... 386
A.2.3. Isochronous Bandwidth Allocation ... 387

A.2.4. Isochronous Transaction Latency *388*
A.2.5. An Example Illustrating Isochronous Parameters *389*
A.3. ISOCHRONOUS TRANSACTION RULES 390
A.4. TRANSACTION ORDERING 390
A.5. ISOCHRONOUS DATA COHERENCY 390
A.6. FLOW CONTROL 391
A.7. TOPOLOGY RESTRICTIONS 391
A.8. TRANSFER RELIABILITY 392
A.9. CONSIDERATIONS FOR BANDWIDTH ALLOCATION 393
A.9.1. Isochronous Bandwidth of PCI Express Links *393*
A.9.2. Isochronous Bandwidth of Endpoint Devices *394*
A.9.3. Isochronous Bandwidth of Switches *394*
A.9.4. Isochronous Bandwidth of Root Complex *394*
A.10. CONSIDERATIONS FOR PCI EXPRESS COMPONENTS 394
A.10.1. A PCI Express Endpoint Device as a Requester *394*
A.10.2. A PCI Express Endpoint Device as a Completer *395*
A.10.3. Switches *396*
A.10.4. Root Complex *397*

B. SYMBOL ENCODING 399

C. PHYSICAL LAYER APPENDIX 409

C.1. DATA SCRAMBLING 409

Figures

FIGURE 1-1: PCI EXPRESS LINK 29
FIGURE 1-2: EXAMPLE TOPOLOGY 31
FIGURE 1-3: LOGICAL BLOCK DIAGRAM OF A SWITCH 33
FIGURE 1-4: HIGH-LEVEL LAYERING DIAGRAM 35
FIGURE 1-5: PACKET FLOW THROUGH THE LAYERS 36
FIGURE 1-6: ADVANCED PEER-TO-PEER COMMUNICATION 41
FIGURE 2-1: LAYERING DIAGRAM HIGHLIGHTING THE TRANSACTION LAYER 43
FIGURE 2-2: GENERIC TRANSACTION LAYER PACKET FORMAT 47
FIGURE 2-3: TRANSACTION DESCRIPTOR 48
FIGURE 2-4: TRANSACTION ID 48
FIGURE 2-5: ATTRIBUTES FIELD OF TRANSACTION DESCRIPTOR 50
FIGURE 2-6: VIRTUAL CHANNEL CONCEPT – AN ILLUSTRATION 57
FIGURE 2-7: VIRTUAL CHANNEL CONCEPT – SWITCH INTERNALS (UPSTREAM FLOW) 57
FIGURE 2-8: AN EXAMPLE OF TC/VC CONFIGURATIONS 60
FIGURE 2-9: REQUEST HEADER FORMAT FOR 32B ADDRESSING OF MEMORY 68
FIGURE 2-10: REQUEST HEADER FORMAT FOR 64B ADDRESSING OF MEMORY 68
FIGURE 2-11: REQUEST HEADER FORMAT FOR I/O TRANSACTIONS 68
FIGURE 2-12: REQUEST HEADER FORMAT FOR CONFIGURATION TRANSACTIONS 68
FIGURE 2-13: REQUEST HEADER FORMAT FOR MSG REQUEST 69
FIGURE 2-14: REQUEST HEADER FORMAT FOR MSGD REQUEST 69
FIGURE 2-15: REQUEST HEADER FORMAT FOR MSGAS REQUEST 69
FIGURE 2-16: REQUEST HEADER FORMAT FOR MSGASD REQUEST 69
FIGURE 2-17: COMPLETION HEADER FORMAT 76
FIGURE 2-18: COMPLETER ID 77
FIGURE 2-19: FLOWCHART FOR HANDLING OF RECEIVED TLPS 79
FIGURE 2-20: FLOWCHART FOR SWITCH HANDLING OF TLPS 80
FIGURE 2-21: FLOWCHART FOR HANDLING OF RECEIVED REQUEST 82
FIGURE 2-22: INTX COLLAPSING IN A DUAL-HEADED BRIDGE 92
FIGURE 2-23: PAYLOAD_DEFINED MESSAGE 96
FIGURE 2-24: RELATIONSHIP BETWEEN REQUESTER AND ULTIMATE COMPLETER 99
FIGURE 2-25: CALCULATION OF 32B ECRC FOR TLP END TO END DATA INTEGRITY PROTECTION 112
FIGURE 3-1: LAYERING DIAGRAM HIGHLIGHTING THE DATA LINK LAYER 117
FIGURE 3-2: DATA LINK CONTROL AND MANAGEMENT STATE MACHINE 119
FIGURE 3-3: FLOWCHART DIAGRAM OF FLOW CONTROL INITIALIZATION PROTOCOL 122
FIGURE 3-4: DLLP TYPE AND CRC FIELDS 126
FIGURE 3-5: DATA LINK LAYER PACKET FORMAT FOR ACK AND NAK 127
FIGURE 3-6: DATA LINK LAYER PACKET FORMAT FOR INITFC1 127
FIGURE 3-7: DATA LINK LAYER PACKET FORMAT FOR INITFC2 127
FIGURE 3-8: DATA LINK LAYER PACKET FORMAT FOR UPDATEFC 127
FIGURE 3-9: PM DATA LINK LAYER PACKET FORMAT 127
FIGURE 3-10: VENDOR SPECIFIC DATA LINK LAYER PACKET FORMAT 128
FIGURE 3-11: DIAGRAM OF CRC CALCULATION FOR DLLPS 129

FIGURE 3-12: TLP WITH LCRC AND SEQUENCE NUMBER APPLIED 130
FIGURE 3-13: TLP FOLLOWING APPLICATION OF SEQUENCE NUMBER AND RESERVED BITS 132
FIGURE 3-14: CALCULATION OF LCRC 134
FIGURE 3-15: RECEIVED DLLP ERROR CHECK FLOWCHART 139
FIGURE 3-16: ACK/NAK DLLP PROCESSING FLOWCHART 140
FIGURE 3-17: RECEIVE DATA LINK LAYER HANDLING OF TLPS 145
FIGURE 4-1: HIGH LEVEL LAYERING DIAGRAM HIGHLIGHTING PHYSICAL LAYER 149
FIGURE 4-2: CHARACTER TO SYMBOL MAPPING 150
FIGURE 4-3: BIT TRANSMISSION ORDER ON PHYSICAL LANES - X1 EXAMPLE 151
FIGURE 4-4: BIT TRANSMISSION ORDER ON PHYSICAL LANES - X4 EXAMPLE 151
FIGURE 4-5: TLP WITH FRAMING SYMBOLS APPLIED 154
FIGURE 4-6: DLLP WITH FRAMING SYMBOLS APPLIED 155
FIGURE 4-7: FRAMED TLP ON A X1 LINK 155
FIGURE 4-8: FRAMED TLP ON A X2 LINK 156
FIGURE 4-9: FRAMED TLP ON A X4 LINK 156
FIGURE 4-10: LFSR WITH SCRAMBLING POLYNOMIAL 157
FIGURE 4-11: WIDTH NEGOTIATION, SIMPLIFIED STATE MACHINE, DOWNSTREAM COMPONENT (PART 1) 170
FIGURE 4-12: WIDTH NEGOTIATION, SIMPLIFIED STATE MACHINE, DOWNSTREAM COMPONENT (PART 2) 171
FIGURE 4-13: WIDTH NEGOTIATION, SIMPLIFIED STATE MACHINE, UPSTREAM COMPONENT (PART 1) 172
FIGURE 4-14: WIDTH NEGOTIATION, SIMPLIFIED STATE MACHINE, UPSTREAM COMPONENT (PART 2) 173
FIGURE 4-15: WIDTH NEGOTIATION EXAMPLE 174
FIGURE 4-16: LINK WIDTH NEGOTIATION; STEPS 1,2 176
FIGURE 4-17: LINK WIDTH NEGOTIATION; STEPS 3, 4 177
FIGURE 4-18: LINK WIDTH NEGOTIATION; STEPS 5, 6 179
FIGURE 4-19: MAIN STATE DIAGRAM FOR LINK TRAINING AND STATUS STATE MACHINE 183
FIGURE 4-20: DETECT SUB-STATE MACHINE 184
FIGURE 4-21: POLLING SUB-STATE MACHINE 186
FIGURE 4-22: CONFIGURATION SUB-STATE MACHINE 188
FIGURE 4-23: RECOVERY SUB-STATE MACHINE 189
FIGURE 4-24: L0S SUB-STATE MACHINE 191
FIGURE 4-25: L1 SUB-STATE MACHINE 192
FIGURE 4-26: L2 SUB-STATE MACHINE 193
FIGURE 4-27: LOOPBACK STATE MACHINE 195
FIGURE 4-28: SAMPLE DIFFERENTIAL SIGNAL 202
FIGURE 4-29: SAMPLE TRANSMITTED WAVEFORM SHOWING -3.5 DB DE-EMPHASIS AROUND A 0.5 V COMMON MODE 203
FIGURE 4-30: A 30 KHZ BEACON SIGNALING THROUGH A 75 NF CAPACITOR 205
FIGURE 4-31: BEACON, WHICH INCLUDES A 2 NS PULSE THROUGH A 75 NF CAPACITOR 205

FIGURE 4-32: MINIMUM TRANSMITTER TIMING AND VOLTAGE OUTPUT COMPLIANCE SPECIFICATION 209
FIGURE 4-33: COMPLIANCE TEST/MEASUREMENT LOAD 210
FIGURE 4-34: MINIMUM RECEIVER EYE TIMING AND VOLTAGE COMPLIANCE SPECIFICATION 214
FIGURE 5-1: PCI EXPRESS ROOT COMPLEX DEVICE MAPPING 216
FIGURE 5-2: PCI EXPRESS SWITCH DEVICE MAPPING 216
FIGURE 5-3: PCI EXPRESS CONFIGURATION SPACE LAYOUT 217
FIGURE 5-4: COMMON CONFIGURATION SPACE HEADER 223
FIGURE 5-5: TYPE 0 CONFIGURATION SPACE HEADER 228
FIGURE 5-6: TYPE 1 CONFIGURATION SPACE HEADER 229
FIGURE 5-7: PCI POWER MANAGEMENT CAPABILITY STRUCTURE 232
FIGURE 5-8: POWER MANAGEMENT CAPABILITIES 232
FIGURE 5-9: POWER MANAGEMENT STATUS/CONTROL 233
FIGURE 5-10: PCI EXPRESS CAPABILITY STRUCTURE 234
FIGURE 5-11: PCI EXPRESS CAPABILITY LIST REGISTER 235
FIGURE 5-12: PCI EXPRESS CAPABILITIES REGISTER 235
FIGURE 5-13: DEVICE CAPABILITIES REGISTER 237
FIGURE 5-14: DEVICE CONTROL REGISTER 241
FIGURE 5-15: DEVICE STATUS REGISTER 244
FIGURE 5-16: LINK CAPABILITIES REGISTER 246
FIGURE 5-17: LINK CONTROL REGISTER 248
FIGURE 5-18: LINK STATUS REGISTER 250
FIGURE 5-19: SLOT CAPABILITIES REGISTER 251
FIGURE 5-20: SLOT CONTROL REGISTER 253
FIGURE 5-21: SLOT STATUS REGISTER 255
FIGURE 5-22: ROOT CONTROL REGISTER 256
FIGURE 5-23: ROOT STATUS REGISTER 257
FIGURE 5-24: PCI EXPRESS EXTENDED CONFIGURATION SPACE LAYOUT 258
FIGURE 5-25: PCI EXPRESS ENHANCED CAPABILITY HEADER 259
FIGURE 5-26: PCI EXPRESS ADVANCED ERROR REPORTING EXTENDED CAPABILITY STRUCTURE 260
FIGURE 5-27: ADVANCED ERROR REPORTING ENHANCED CAPABILITY HEADER 261
FIGURE 5-28: UNCORRECTABLE ERROR STATUS REGISTER 262
FIGURE 5-29: UNCORRECTABLE ERROR MASK REGISTER 263
FIGURE 5-30: UNCORRECTABLE ERROR SEVERITY REGISTER 264
FIGURE 5-31: CORRECTABLE ERROR STATUS REGISTER 265
FIGURE 5-32: CORRECTABLE ERROR MASK REGISTER 265
FIGURE 5-33: ADVANCED ERROR CAPABILITIES AND CONTROL REGISTER 266
FIGURE 5-34: HEADER LOG REGISTER 267
FIGURE 5-35: ROOT ERROR COMMAND REGISTER 268
FIGURE 5-36: ROOT ERROR STATUS REGISTER 269
FIGURE 5-37: ERROR SOURCE IDENTIFICATION REGISTER 270
FIGURE 5-38: PCI EXPRESS VIRTUAL CHANNEL CAPABILITY STRUCTURE 271
FIGURE 5-39: VIRTUAL CHANNEL ENHANCED CAPABILITY HEADER 272
FIGURE 5-40: PORT VC CAPABILITY REGISTER 1 273

FIGURE 5-41: PORT VC CAPABILITY REGISTER 2 275
FIGURE 5-42: PORT VC CONTROL REGISTER 276
FIGURE 5-43: PORT VC STATUS REGISTER 277
FIGURE 5-44: VC RESOURCE CAPABILITY REGISTER 277
FIGURE 5-45: VC RESOURCE CONTROL REGISTER 279
FIGURE 5-46: VC RESOURCE STATUS REGISTER 281
FIGURE 5-47: STRUCTURE OF AN EXAMPLE VC ARBITRATION TABLE WITH 32-PHASES.283
FIGURE 5-48: EXAMPLE PORT ARBITRATION TABLE WITH 128 PHASES AND 2-BIT TABLE ENTRIES 284
FIGURE 5-49: PCI EXPRESS DEVICE SERIAL NUMBER CAPABILITY STRUCTURE 285
FIGURE 5-50: DEVICE SERIAL NUMBER ENHANCED CAPABILITY HEADER 285
FIGURE 5-51: SERIAL NUMBER REGISTER 286
FIGURE 5-52: PCI EXPRESS POWER BUDGETING CAPABILITY STRUCTURE 287
FIGURE 5-53: POWER BUDGETING ENHANCED CAPABILITY HEADER 287
FIGURE 5-54: POWER BUDGETING DATA REGISTER 289
FIGURE 5-55: POWER BUDGET CAPABILITY REGISTER 291
FIGURE 6-1: LINK POWER MANAGEMENT STATE TRANSITIONS 297
FIGURE 6-2: ENTRY INTO L1 LINK STATE 303
FIGURE 6-3: EXIT FROM L1 LINK STATE INITIATED BY UPSTREAM COMPONENT 306
FIGURE 6-4: A CONCEPTUAL PME CONTROL STATE MACHINE 313
FIGURE 6-5: L1 TRANSITION SEQUENCE ENDING WITH A REJECTION 324
FIGURE 6-6: L1 SUCCESSFUL TRANSITION SEQUENCE 324
FIGURE 6-7: EXAMPLE OF L1 EXIT LATENCY COMPUTATION 326
FIGURE 6-8: EXAMPLE OF PME MESSAGE ADDRESSING IN A PCI EXPRESS-TO-PCI BRIDGE 334
FIGURE 7-1: ERROR CLASSIFICATION 339
FIGURE 7-2: AN EXAMPLE OF SYMMETRICAL TC TO VC MAPPING 349
FIGURE 7-3: AN EXAMPLE OF ASYMMETRICAL TC TO VC MAPPING 350
FIGURE 7-4: AN EXAMPLE OF TRAFFIC FLOW ILLUSTRATING INGRESS AND EGRESS 351
FIGURE 7-5: AN EXAMPLE OF DIFFERENTIATED TRAFFIC FLOW THROUGH A SWITCH 351
FIGURE 7-6: SWITCH ARBITRATION STRUCTURE 352
FIGURE 7-7: VC ID AND PRIORITY ORDER – AN EXAMPLE 354
FIGURE 7-8: HOT PLUG LOGIC 378
FIGURE A-1: AN EXAMPLE SHOWING ENDPOINT-TO-ROOT-COMPLEX AND PEER-TO-PEER COMMUNICATION MODELS 384
FIGURE A-2: TWO BASIC BANDWIDTH RESOURCING PROBLEMS: OVER-SUBSCRIPTION AND CONGESTION 385
FIGURE A-3: A SIMPLIFIED EXAMPLE ILLUSTRATING PCI EXPRESS ISOCHRONOUS PARAMETERS 389
FIGURE A-4: AN EXAMPLE OF PCI EXPRESS TOPOLOGY SUPPORTING ISOCHRONOUS APPLICATIONS 392
FIGURE C-1: SCRAMBLING SPECTRUM FOR DATA VALUE OF 0 415

Tables

TABLE 2-1: TRANSACTION TYPES FOR DIFFERENT ADDRESS SPACES 44
TABLE 2-2: ORDERING ATTRIBUTES 51
TABLE 2-3: CACHE COHERENCY MANAGEMENT ATTRIBUTE 51
TABLE 2-4 DEFINITION OF TC FIELD ENCODINGS 52
TABLE 2-5: ORDERING RULES SUMMARY TABLE 53
TABLE 2-6: TC TO VC MAPPING EXAMPLE 59
TABLE 2-7: TD AND EP FIELD VALUES 63
TABLE 2-8: FMT[1:0] AND TYPE[4:0] FIELD ENCODINGS 63
TABLE 2-9: MESSAGE ROUTING 65
TABLE 2-10: MSG CODES 73
TABLE 2-11: MSGD CODES 76
TABLE 2-12: SWITCH MAPPING FOR INTX 91
TABLE 2-13: POWER MANAGEMENT SYSTEM MESSAGES 93
TABLE 2-14: ERROR MESSAGES 94
TABLE 2-15: HOT PLUG SIGNALING MESSAGES 97
TABLE 2-16: FLOW CONTROL CREDIT TYPES 100
TABLE 2-17: TLP FLOW CONTROL CREDIT CONSUMPTION 101
TABLE 2-18: MINIMUM FLOW CONTROL ADVERTISEMENTS 102
TABLE 2-19: UPDATEFC TRANSMISSION LATENCY GUIDELINES BY LINK WIDTH AND MAX PAYLOAD (SYMBOL TIMES) 108
TABLE 2-20: MAPPING OF BITS INTO ECRC FIELD 110
TABLE 3-1: DLLP TYPE ENCODINGS 125
TABLE 3-2: MAPPING OF BITS INTO CRC FIELD 129
TABLE 3-3: MAPPING OF BITS INTO LCRC FIELD 133
TABLE 3-4: REPLAY_TIMER LIMITS BY LINK WIDTH AND MAX_PAYLOAD_SIZE (SYMBOL TIMES) TOLERANCE: -0% / +100% 136
TABLE 3-5: ACK TRANSMISSION LATENCY LIMIT AND ACKFACTOR BY LINK WIDTH AND MAX PAYLOAD (SYMBOL TIMES) 147
TABLE 4-1: SPECIAL SYMBOLS 152
TABLE 4-2: TS1 ORDERED-SET 159
TABLE 4-3: TS2 ORDERED-SET 160
TABLE 4-4: DIFFERENTIAL TRANSMITTER (TX) OUTPUT SPECIFICATIONS 206
TABLE 4-5: DIFFERENTIAL RECEIVER (RX) INPUT SPECIFICATIONS 211
TABLE 5-1: CONFIGURATION ADDRESS MAPPING 218
TABLE 5-2: REGISTER (AND REGISTER BIT-FIELD) TYPES 221
TABLE 5-3: COMMAND REGISTER 224
TABLE 5-4: STATUS REGISTER 225
TABLE 5-5: SECONDARY STATUS REGISTER 230
TABLE 5-6: BRIDGE CONTROL REGISTER 231
TABLE 5-7: POWER MANAGEMENT CAPABILITIES 232
TABLE 5-8: POWER MANAGEMENT STATUS/CONTROL 233
TABLE 5-9: PCI EXPRESS CAPABILITY LIST REGISTER 235
TABLE 5-10: PCI EXPRESS CAPABILITIES REGISTER 236

TABLE 5-11: DEVICE CAPABILITIES REGISTER 237
TABLE 5-12: DEVICE CONTROL REGISTER 241
TABLE 5-13: DEVICE STATUS REGISTER 245
TABLE 5-14: LINK CAPABILITIES REGISTER 246
TABLE 5-15: LINK CONTROL REGISTER 248
TABLE 5-16: LINK STATUS REGISTER 250
TABLE 5-17: SLOT CAPABILITIES REGISTER 251
TABLE 5-18: SLOT CONTROL REGISTER 253
TABLE 5-19: SLOT STATUS REGISTER 255
TABLE 5-20: ROOT CONTROL REGISTER 257
TABLE 5-21: ROOT STATUS REGISTER 258
TABLE 5-22: PCI EXPRESS ENHANCED CAPABILITY HEADER 259
TABLE 5-23: ADVANCED ERROR REPORTING ENHANCED CAPABILITY HEADER 261
TABLE 5-24: UNCORRECTABLE ERROR STATUS REGISTER 262
TABLE 5-25: UNCORRECTABLE ERROR MASK REGISTER 263
TABLE 5-26: UNCORRECTABLE ERROR SEVERITY REGISTER 264
TABLE 5-27: CORRECTABLE ERROR STATUS REGISTER 265
TABLE 5-28: CORRECTABLE ERROR MASK REGISTER 266
TABLE 5-29: ADVANCED ERROR CAPABILITIES REGISTER 266
TABLE 5-30: HEADER LOG REGISTER 267
TABLE 5-31: ROOT ERROR COMMAND REGISTER 268
TABLE 5-32: ROOT ERROR STATUS REGISTER 269
TABLE 5-33: ERROR SOURCE IDENTIFICATION REGISTER 270
TABLE 5-34: VIRTUAL CHANNEL ENHANCED CAPABILITY HEADER 272
TABLE 5-35: PORT VC CAPABILITY REGISTER 1 273
TABLE 5-36: PORT VC CAPABILITY REGISTER 2 275
TABLE 5-37: PORT VC CONTROL REGISTER 276
TABLE 5-38: PORT VC STATUS REGISTER 277
TABLE 5-39: VC RESOURCE CAPABILITY REGISTER 278
TABLE 5-40: VC RESOURCE CONTROL REGISTER 279
TABLE 5-41: VC RESOURCE STATUS REGISTER 282
TABLE 5-42: DEFINITION OF THE 4-BIT ENTRIES IN THE VC ARBITRATION TABLE 283
TABLE 5-43 LENGTH OF THE VC ARBITRATION TABLE 283
TABLE 5-44: LENGTH OF PORT ARBITRATION TABLE 284
TABLE 5-45: DEVICE SERIAL NUMBER ENHANCED CAPABILITY HEADER 285
TABLE 5-46: SERIAL NUMBER REGISTER 286
TABLE 5-47: POWER BUDGETING ENHANCED CAPABILITY HEADER 288
TABLE 5-48: POWER BUDGETING DATA REGISTER 289
TABLE 5-49: POWER BUDGET CAPABILITY REGISTER 291
TABLE 6-1: SUMMARY OF PCI EXPRESS LINK POWER MANAGEMENT STATES 298
TABLE 6-2: RELATION BETWEEN POWER MANAGEMENT STATES OF LINK AND COMPONENTS 302
TABLE 6-3: ENCODING OF THE ACTIVE STATE LINK PM SUPPORT FIELD 327
TABLE 6-4: DESCRIPTION OF THE SLOT CLOCK CONFIGURATION FIELD 327
TABLE 6-5: DESCRIPTION OF THE COMMON CLOCK CONFIGURATION FIELD 328
TABLE 6-6: ENCODING OF THE L0S EXIT LATENCY FIELD 328

TABLE 6-7: ENCODING OF THE L1 EXIT LATENCY FIELD 329
TABLE 6-8: ENCODING OF THE ENDPOINT L0S ACCEPTABLE LATENCY FIELD 329
TABLE 6-9: ENCODING OF THE ENDPOINT L1 ACCEPTABLE LATENCY FIELD 330
TABLE 6-10: ENCODING OF THE ACTIVE STATE LINK PM CONTROL FIELD 330
TABLE 6-11: POWER MANAGEMENT SYSTEM MESSAGES AND DLLPS 333
TABLE 7-1: ERROR MESSAGES 341
TABLE 7-2: PHYSICAL LAYER ERROR LIST 344
TABLE 7-3: DATA LINK LAYER ERROR LIST 344
TABLE 7-4: TRANSACTION LAYER ERROR LIST 345
TABLE 7-5: ELEMENTS OF THE STANDARD USAGE MODEL 367
TABLE 7-6: ATTENTION INDICATOR STATES 368
TABLE 7-7: POWER INDICATOR STATES 369
TABLE 7-8: EVENT BEHAVIOR 371
TABLE A-1: ISOCHRONOUS BANDWIDTH RANGES AND GRANULARITIES 387
TABLE A-2: MAXIMUM NUMBER OF VIRTUAL TIMESLOTS ALLOWED FOR DIFFERENT PCI EXPRESS LINKS AT 2.5 GHZ 393
TABLE B-1: 8B/10B DATA SYMBOL CODES 399
TABLE B-2: 8B/10B SPECIAL CHARACTER SYMBOL CODES 407

Preface

Traditional multi-drop, parallel bus technology is approaching its practical performance limits. It is clear that balancing system performance requires I/O bandwidth to scale with processing and application demands. There is an industry mandate to re-engineer I/O connectivity within cost constraints. PCI Express comprehends the many I/O requirements presented across the spectrum of computing and communications platforms, and rolls them into a common scalable and extensible I/O industry specification. Alongside these increasing performance demands, the enterprise server and communications markets have the need for improved reliability, security, and quality of service guarantees. This specification will therefore be applicable to multiple market segments.

Technology advances in high-speed, point-to-point interconnects enable us to break away from the bandwidth limitations of multi-drop, parallel buses. The PCI Express basic physical layer consists of a differential transmit pair and a differential receive pair. Dual simplex data on these point-to-point connections is self-clocked and its bandwidth increases linearly with interconnect width and frequency. PCI Express takes an additional step of including a message space within its bus protocol that is used to implement legacy "side-band" signals. This further reduction of signal pins produces a very low pin count connection for components and adapters. The PCI Express Transaction, Data Link, and Physical Layers are optimized for chip-to-chip and board-to-board interconnect applications.

An inherent limitation of today's PCI-based platforms is the lack of support for isochronous data delivery, an attribute that is especially important to streaming media applications. To enable these emerging applications, PCI Express adds a virtual channel mechanism. In addition to use for support of isochronous traffic, the virtual channel mechanism provides an infrastructure for future extensions in supporting new applications. By adhering to the PCI Software Model, today's applications are easily migrated even as emerging applications are enabled.

Key PCI Express architectural attributes include:

- Continuation of the PCI Software Model
- Serial, differential, low-voltage signaling
- Layered architecture enabling physical layer attachment to copper, optical, or emerging physical signaling media
- Predictable, low latency suitable for applications requiring isochronous data delivery
- Robust data integrity and error handling in support of highly reliable systems
- Embedded clocking scheme using 8 bit/10 bit encoding
- High bandwidth per pin
- Bandwidth scalability through Lane width and frequency
- Hot attach and detach capability
- Aggressive power management capabilities

Objective of the Specification

This specification describes the PCI Express architecture, interconnect attributes, bus 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), 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. At this time, the PCI Express Base Specification and the PCI Express Card Electromechanical Specification are being published. As the PCI Express definition evolves, other companion 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.

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

Documentation Conventions

Capitalization

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

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

Numbers and Number Bases

Hexadecimal numbers are written with a lower case "h" suffix, e.g., 0FFFFh 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.

Reference Information

Reference information is provided in various places to assist the reader and does not represent a requirement of this document. Such references are indicated by the abbreviation "(ref)." For example, in some places, a clock that is specified to have a minimum period of 400 ps also includes the reference information maximum clock frequency of "2.5 GHz (ref)." Requirements of other specifications also appear in various places throughout this document and are marked as reference information. Every effort has been made to guarantee that this information accurately reflects the referenced document; however, in case of a discrepancy, the original document takes precedence.

Implementation Notes

Implementation Notes should not be considered to be part of this specification. They are included for clarification and illustration only. Implementation Notes within this document are enclosed in a box and set apart from other text.

Terms and Abbreviations

8b/10b	The data encoding scheme[1] used in the PCI Express Physical Layer.
Advertise (Credits)	The term Advertise is used in the context of Flow Control to refer to the act of a Receiver sending information regarding its Flow Control Credit availability by using a Flow Control Update Message.
asserted	The active logical state of a conceptual or actual signal.
attribute	Transaction handling preferences indicated by specified Packet header bits and fields (for example, non-snoop).
core features	A set of required features that must be supported by a device for it to be considered compliant to the PCI Express Specification.
Beacon	30 kHz-500 MHz signal used to exit L2.
Bridge	A device which virtually or actually connects a PCI/PCI-X segment or PCI Express Port with an internal component interconnect or another PCI/PCI-X segment or PCI Express Port. A Bridge must include a software configuration interface as described in this document.
x8	Refers to a Link or Port with eight Physical Lanes.
x1	Refers to a Link or Port with one Physical Lane.
xN	Refers to a Link with "N" Physical Lanes.
Character	An 8 bit quantity treated as an atomic entity; a Byte.
cold reset	A "Power Good Reset" following the application of power.
Completer	The logical device addressed by a Request.
Completer ID	The combination of a Completer's Bus Number, Device Number, and Function Number which uniquely identifies the Completer of the Request.
Completion	A Packet used to terminate, or to partially terminate, a Sequence is referred to as a *Completion*. A *Completion* always corresponds to a preceding Request, and in some cases includes data.
Configuration Space	One of the four address spaces within the PCI Express architecture. Packets with a *Configuration Space* address are used to configure a device.
conventional PCI	Protocol conforming to the *PCI Local Bus Specification, Rev. 2.3*.
component	A physical device (a single package).
Data Link Layer	The intermediate layer of the PCI Express architecture that sits between the Transaction Layer and the Physical Layer.
DLLP or Data Link Layer Packet	Packet generated in the Data Link Layer to support Link management functions.

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

Data Payload	Some Packets include information following the header that is destined for consumption by the logical device receiving the Packet (for example, Write Requests or Read Completions). This information is called a Data Payload.
deasserted	The term deasserted refers to the inactive logical state of a conceptual or actual signal.
device	A logical device, corresponding to a PCI device configuration space. May be used to refer to either a single or multi-function device.
Downstream	Downstream refers either to the relative position of an interconnect/system element (Link/device) as something that is farther from the Root Complex, or to a direction of information flow, i.e., when information is flowing away from the Root Complex. The Ports on a Switch which are not the Upstream Port are Downstream Ports. All Ports on a Root Complex are Downstream Ports. The Downstream component on a Link is the component farther from the Root Complex.
DFT	Acronym for Design for Testability.
DWORD, DW	Four bytes of data on a naturally aligned four-byte boundary (i.e., the least significant two bits of the address are 00b).
egress	Refers to direction. Means outgoing, i.e., transmitting direction.
Egress Port	Transmitting port, i.e., the port that sends outgoing traffic. Typically used as a reference to the role that port of the Switch has in the context of a transaction or more broadly in the context of traffic flow.
Electrical Idle	State of the output driver where both lines, D+ and D-, are driven to the DC common mode voltage.
Electrical Idle Exit	When a receiver currently in Electrical Idle detects a signal at its input port.
Endpoint	A PCI Express device with a Type 00h Configuration Space header.
Error Recovery,	
Error Detection	Refers to the mechanisms for ensuring integrity of data transfer, including the management of the transmit side retry buffer(s).
Flow Control	A method for communicating receive buffer information from a Receiver to a Transmitter to prevent receive buffer overflow and allow Transmitter compliance with ordering rules.
FCP or Flow Control Packet	DLLP used to send Flow Control information from the Transaction Layer in one component to the Transaction Layer in another component.
function	A logical function corresponding to a PCI function configuration space. May be used to refer to one function of a multi-function device, or to the only function in a single-function device.
header	A set of fields that appear at the front of a Packet that contain the information required to determine the characteristics and purpose of the Packet.
Hierarchy	The Hierarchy defines the I/O interconnect topology supported by the PCI Express Architecture.

Hierarchy Domain	A PCI Express Hierarchy is segmented into multiple fragments by the Root Complex that sources more than one PCI Express interface. These sub-hierarchies are called Hierarchy Domains.
Host Bridge	A Host Bridge is a part of a Root Complex which connects a host CPU or CPUs to a PCI Express Hierarchy.
hot reset	A reset propagated in-band across a Link using a Physical Layer Mechanism.
ingress	Refers to direction. Means incoming, i.e., receiving direction.
Ingress Port	Receiving port, i.e., the port that accepts incoming traffic. Typically used as a reference to the role that port of the Switch has in the context of a transaction or more broadly in the context of traffic flow.
I/O Space	One of the four address spaces of the PCI Express architecture. Identical to the I/O space defined in PCI.
isochronous	Refers to data associated with time-sensitive applications, such as audio or video applications.
invariant	An *invariant* field of a TLP Header contains a value which 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	Unit of distinction applied to the PCI Express Specification to clarify the behavior of key elements of the interface. The use of the term Layer is not intended to imply a specific implementation.
Link	A dual-simplex communications path between two components. The collection of two Ports and their interconnecting Lanes.
LinkUp	Status from the Physical layer to the Link layer indicating both ends of the Link are connected.
Logical Bus	The logical connection among a collection of devices that have the same bus number in Configuration Space.
logical device	An element of a PCI Express system that responds to a unique device number in Configuration Space. As for physical devices in PCI 2.3, logical devices either include a single function or are multi-function devices. Furthermore, the term "logical device" is often used when describing requirements that apply individually to all functions within the logical device. Unless otherwise specified, logical device requirements in this specification apply to single function logical devices and to each function individually of a multi-function logical device.
Logical Idle	A period of one or more symbol times when no information: TLPs, DLLPs, or any special symbol is being transmitted or received. Unlike electrical idle, during logical idle the idle character is being transmitted and received.
Malformed Packet	A TLP which violates TLP formation rules.
Memory Space	One of the four address spaces of the PCI Express architecture. Identical to the memory space defined in PCI.
Message	A Packet with a Message Space type.

Message Signaled Interrupt, MSI	An optional feature that enables a device to request service by writing a system-specified DW of data to a system-specified address using a Memory Write semantic Request.
Message Space	One of the four address spaces of the PCI Express architecture.
naturally aligned	Used in reference to a data payload which is some power of two in length (L), indicates that the starting address of the data payload equals an integer multiple of L.
Packet	A fundamental unit of information transfer consisting of a header that, in some cases, is followed by a Data Payload.
PCI bus	The PCI Local Bus, as specified in the PCI 2.3 and PCI-X 1.0a specifications.
PCI Software Model	The software model necessary to initialize, discover, configure, and use PCI device, as specified in PCI 2.3, PCI-X 1.0a, and PCI BIOS specifications.
Phantom Function Number, PFN	An unclaimed function number that may be used to expand the number of outstanding transaction identifiers by logically combining the PFN with the Tag identifier to create a unique transaction identification tuple.
Physical Lane	See Lane.
Physical Layer	The layer of the PCI Express architecture that directly interacts with the communication medium between the two components.
Port	In a logical sense, an interface associated with a component, between that component and a PCI Express Link. In physical terms, a group of transmitters and receivers physically located on the same chip that define a Link.
PPM	Parts per Million –Applied to frequency, this is the difference, in millionths of a Hertz, between some stated ideal frequency, and the measured *long-term* average of a frequency.
QWORD, QW	Sixty-four bits (eight bytes) of data on a naturally aligned eight-byte boundary (i.e., the least significant three bits of the address are 000b).
Receiver	The component receiving Packet information across a Link.
Receiving Port	A Port on which a Packet is received.
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) in the PCI Express Specification is not permitted. Any use of the reserved areas of the PCI Express Specification will result in a product that is not PCI Express-compliant. The functionality of any such product cannot be guaranteed in this or any future revision of the PCI Express Specification.
Request	A Packet used to initiate a Sequence is referred to as a Request. A Request includes some operation code, and, in some cases, it includes address and length, data, or other information.
Requester	A logical device that first introduces a 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. In most cases, a PCI Express bridge or Switch forwards Requests from one interface to another without modifying the Requester ID. A bridge from a bus other than PCI Express (including a PCI bus operating in conventional mode) must store the Requester ID for use when creating a Completion for the Request.
Root Complex	An entity that includes a Host Bridge and one or more Root Ports.
Root Port	A PCI Express Port, on a Root Complex, that maps a portion of the PCI Express interconnect Hierarchy through an associated virtual PCI-PCI Bridge.
Sequence	A single Request and zero or more Completions associated with carrying out a single logical transfer by a Requester.
Standard Hot-Plug Controller (SHPC)	A PCI hot-plug controller compliant with SHPC 1.0.
Split Transaction	A single logical transfer containing an initial transaction (the Split Request) that the target (the completer or a bridge) terminates with Split Response, followed by one or more transactions (the Split Completions) initiated by the completer (or bridge) to send the read data (if a read) or a completion message back to the requester.
Switch	A Switch connects two or more Ports to allow Packets to be routed from one Port to another. To configuration software, a Switch presents the appearance of an assemblage of PCI-to-PCI Bridges.
Symbol	A 10 bit quantity produced as the result of 8b/10b encoding.
Symbol Time	The period of time required to place a Symbol on a Lane (ten times the Unit Interval).
Tag	A number assigned to a given Non-posted Request to distinguish Completions for that Request from other Requests.
TBD	To be defined by PCI-SIG.
Transaction Descriptor	An element of a Packet header that, in addition to Address, Length, and Type, describes the properties of the Transaction.
TLP or Transaction Layer Packet	A Packet generated in the Transaction Layer to convey a Request or Completion.
Transaction Layer	The outermost layer of the PCI Express architecture that operates at the level of transactions (for example, read, write).
Transceiver	The physical transmitter and receiver pair on a single chip.
Transmitter	The component sending Packet information across a Link is the *Transmitter*.
Unsupported Request, UR	A Request Packet that specifies some action or access to some space that is not supported by the Target.

Unit Interval, UI	Given a data stream of 1010..pattern, the Unit Interval is the value measured by averaging the time interval between voltage transitions, over a time interval long enough to make all intentional frequency modulation of the source clock negligible.
Upstream	Upstream refers either to the relative position of an interconnect/system element (Link/device) as something that is closer to the Root Complex, or to a direction of information flow, i.e., when information is flowing towards the Root Complex. The Port on a Switch which is closest topologically to the Root Complex is the Upstream Port. The Port on an Endpoint or Bridge component is an Upstream Port. The Upstream component on a Link is the component closer to the Root Complex.
variant	A variant field of a TLP Header contains a value which is subject to possible modification according to the rules of this specification as the TLP flows through the PCI Express fabric.
warm reset	A reset caused by driving "Power Good" inactive and then active, but without cycling the supplied power.

Reference Documents

PCI Express Card Electromechanical Specification, Rev. 1.0

PCI Local Bus Specification, Rev. 2.3

PCI-X Addendum to the PCI Local Bus Specification, Rev. 1.0a

PCI Hot-Plug Specification, Rev. 1.1

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

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

PCI Power Management Interface Specification, Rev. 1.1

Advanced Configuration and Power Interface Specification, Rev. 2.0

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

26

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 bandwidth-limiting, 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
- **New mechanical form factors:**
    - Mobile, PCI-like form factor and modular, cartridge form factor
- **PCI compatible software model:**
    - Ability to enumerate and configure PCI Express hardware using PCI system firmware 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 propagate an event to wake the system
        - Ability to sequence device power-up to allow graceful platform policy in power budgeting.
    - Ability to support differentiated services, i.e. different qualities of service (QoS)
        - Ability to create end-to-end isochronous (time-based, injection rate control) solutions
        - Ability to have dedicated Link resources per QoS data flow to improve fabric efficiency / effective 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
    - Hot Plug and Hot Swap support
        - Ability to support existing PCI hot-plug and hot-swap solutions
        - Ability to support native hot-plug and hot-swap solutions (no side-band signals required)
        - Ability to support a unified software model for all form factors
    - Multi-hierarchy and advanced peer-to-peer communications
        - Ability to support vendor-specific and PCI Express-standard peer-to-peer communications messaging
        - Ability to Cross Link multiple hierarchies to support peer-to-peer communications across large fabric topologies

- 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 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.

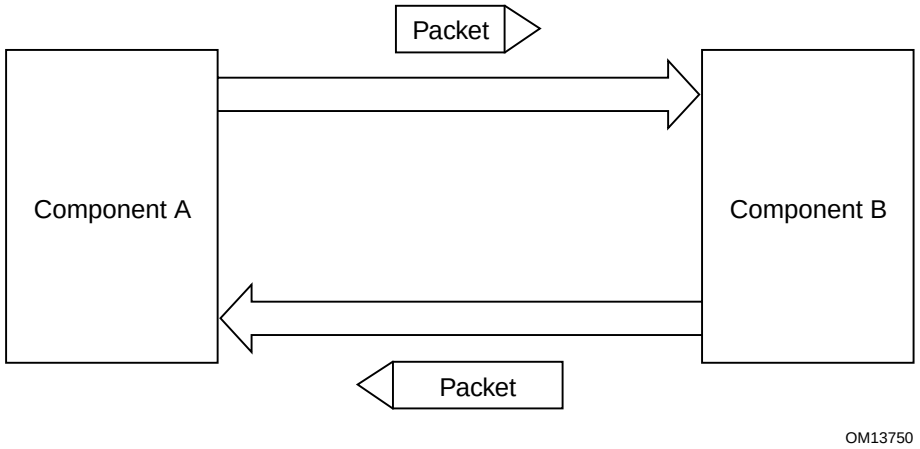

Figure 1-1: PCI Express Link

The primary Link attributes are:

- The basic Link – PCI Express Link consists of dual unidirectional differential Links, implemented as a transmit pair and a receive pair. A data clock is embedded using the 8b/10b-encoding scheme to achieve very high data rates.

- Signaling rate – Once initialized, each Link must only operate at one of the supported signaling levels. For this version of the specification, there is only one signaling rate, which provides an effective 2.5 Gigabits/second/Lane/direction of raw bandwidth. The data rate is expected to increase with the technology advances in future.
- 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. For example, an x8 Link represents an aggregate bandwidth of 20 Gigabits / second of raw bandwidth in each direction. This version of the Physical Layer supports 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., an 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 called a hierarchy – composed of a Root Complex (RC), multiple Endpoints (I/O devices), a Switch, and PCI Express-PCI Bridge all interconnected via PCI Express Links. Each of the components of the topology are mapped in a single flat address space and can be addressed by PCI-like load store accesses.

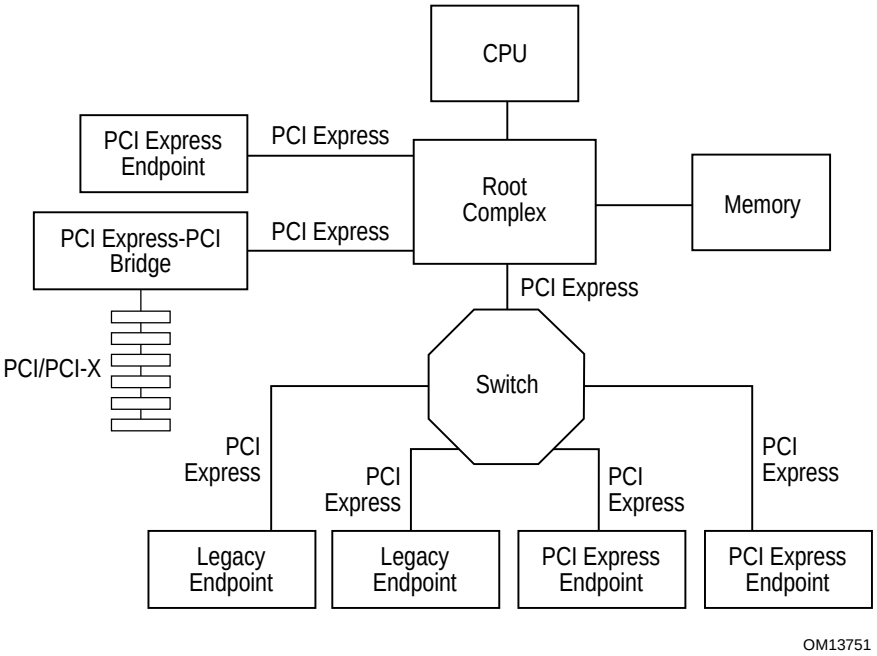

Figure 1-2: Example Topology

1.3.1. Root Complex

- A Root Complex (RC) denotes the root of an I/O hierarchy that connects the CPU/memory subsystem to the I/O.
- As illustrated in the previous figure, a Root Complex may support one or more PCI Express Ports. Each interface defines a separate I/O hierarchy domain. Each hierarchy domain may be composed of a single I/O Endpoint or a sub-hierarchy containing one or more Switch components and I/O Endpoints.
- The capability to route peer-to-peer transactions between hierarchy domains through a Root Complex is optional and implementation dependent. For example, an implementation may incorporate a real or virtual switch internally within the Root Complex to enable full peer-to-peer support in a software transparent way.
- A Root Complex must support generation of configuration requests as a Requester.
- A Root Complex is permitted to support the generation of I/O requests as a Requester.
- A Root Complex must not support Lock semantics as a Completer.
- A Root Complex is permitted to support generation of Locked Requests as a Requester.

1.3.2. Endpoints

"Endpoint" refers to a type of device that can be the Requester or Completer of a PCI Express transaction either on its own behalf or on behalf of a distinct non-PCI Express device (other than a PCI device or Host CPU), e.g., a PCI Express attached graphics controller or a PCI Express-USB interface. Endpoints are classified as either legacy or PCI Express Endpoints. The specific rules for each are described in Sections 1.3.2.1 and 1.3.2.2.

1.3.2.1. Legacy Endpoint Rules

- A Legacy Endpoint must be a device with a Type 00h Configuration Space header.
- A Legacy Endpoint must support Configuration Requests as a Completer
- A Legacy Endpoint may support I/O Requests as a Completer.
- A Legacy Endpoint may generate I/O Requests.
- A Legacy Endpoint may support Lock memory semantics as a Completer if that is required by the device's legacy software support requirements.
- A Legacy Endpoint must not issue a Locked Request.

1.3.2.2. PCI Express Endpoint Rules

- A PCI Express Endpoint must be a device with a Type 00h Configuration Space header.
- A PCI Express Endpoint must support Configuration Requests as a Completer
- A PCI Express Endpoint must not require I/O resources claimed through BAR(s).
- A PCI Express Endpoint must not generate I/O Requests.
- A PCI Express Endpoint must not support Locked Requests as a Completer or generate them as a Requestor. PCI Express-compliant software drivers and applications must be written to prevent the use of lock semantics when accessing a PCI Express Endpoint.

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 (advanced Switch components will support additional capabilities beyond those described below).

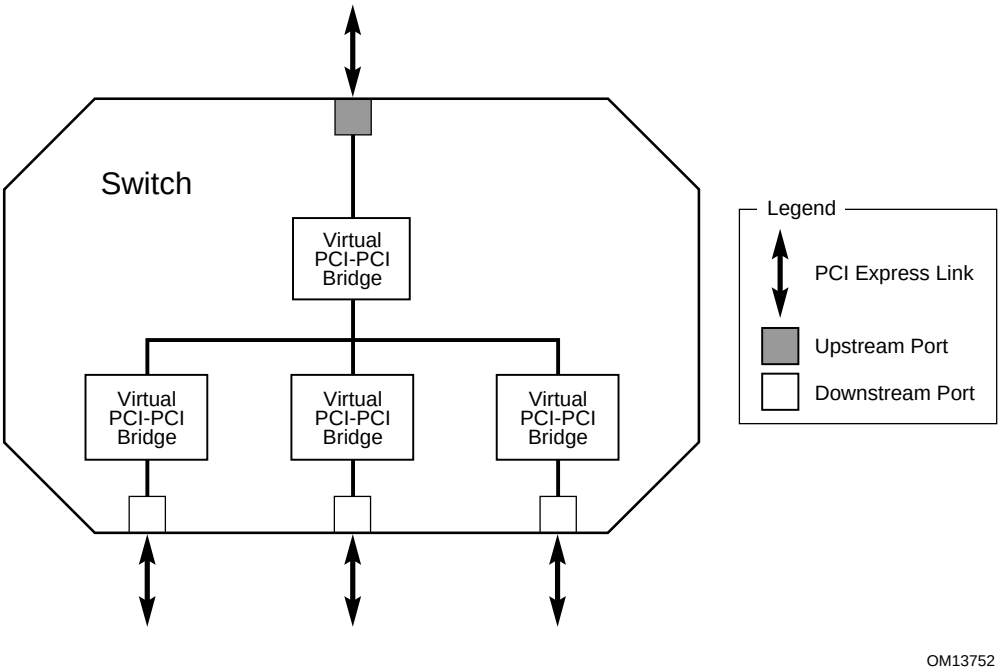

Figure 1-3: Logical Block Diagram of a Switch

- 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.
- A Switch may only forward peer-to-peer transactions between two downstream ports.
- Except as noted in this document, a Switch must forward all types of TLPs (Transaction Layer Packets) between any set of ports.
- Locked Requests must be supported as specified in Section 7.2. 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.
- Each Switch must comply with the Link-level data integrity 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 each of 128 bytes payload.
- 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.

1.3.4. PCI Express-PCI Bridge

- A PCI Express to PCI/PCI-X Bridge has one PCI Express Port, and one or multiple PCI/PCI-X bus interfaces.
- A PCI Express to PCI/PCI-X Bridge must support all required PCI and/or PCI-X transactions on its PCI interface.
- Locked Requests must be supported as specified in Chapter 7. PCI Express-PCI Bridges must not generate (propagate) Locked Requests from PCI to PCI Express, but are required for deadlock prevention to support Locked Requests from PCI Express to PCI.
- The PCI Express Port of PCI Express-PCI Bridge must comply with the flow control specification within this document.
- The PCI Express Port of PCI Express-PCI Bridge must comply with the Link-level data integrity specification within this document.

1.4. PCI Express Fabric Topology Configuration

The PCI Express Configuration model supports two mechanisms:

- **PCI compatible configuration mechanism:** The PCI compatible mechanism supports 100% binary compatibility with PCI 2.3 or later aware operating systems and their corresponding bus enumeration and configuration software.
- **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 PCI-to-PCI Bridge structure and has a logical PCI bus associated with it. A PCI Express Link is represented using a PCI-to-PCI Bridge structure and may either be a PCI Express Root Complex port, a Switch upstream port, or a Switch downstream port. The Root Port is a PCI-to-PCI bridge structure that originates a PCI Express Hierarchy domain from a PCI Express Root Complex. Logical devices are mapped into configuration space such that each will 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.

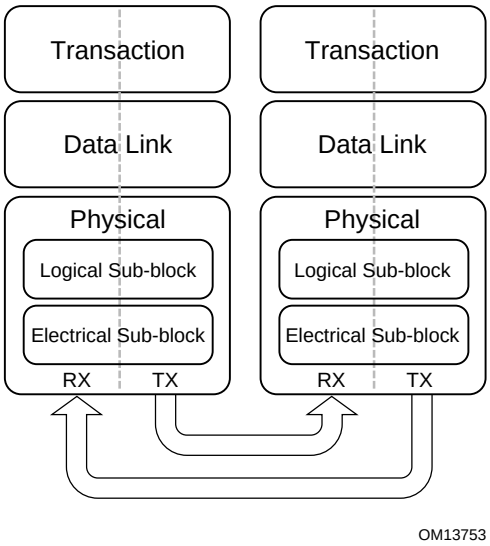

Figure 1-4: High-Level Layering Diagram

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.

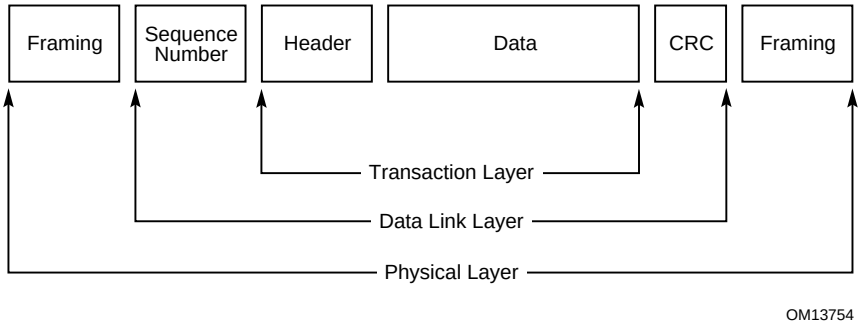

Figure 1-5: Packet Flow Through the Layers

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 Transaction Layer Packets (TLP). TLP 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 TLP.

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 (memory, I/O, configuration, and message) of addressing depending on the type of the transaction. The Packets may also have attributes such as "no-snoop," and "relaxed-ordering" which may be used to optimally route these packets through the system.

The transaction layer supports four address spaces: it includes the three PCI address spaces (memory, I/O, and configuration) and adds a Message Space. This specification uses the Message Signaled Interrupt concept as a primary method for interrupt processing and uses Message Space to support all prior side-band 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. Responsibilities of Data Link Layer include Link management, error detection, and error correction.

The transmission side of the Data Link Layer accepts TLP assembled by the Transaction Layer, calculates and applies 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 TLP 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 TLP 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 includes also logical functions related to interface initialization and maintenance. The Physical Layer exchanges information with the Data Link Layer in an implementation-specific format. This layer is responsible for converting information received from Data Link Layer in to an appropriate serialized format and transmitting it across the PCI Express Link at a frequency and width compatible with the remote device.

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 TLP, 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

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

- Generate TLP from device core Requests
- Convert received Request TLP into Requests for the device core
- Convert received Completion Packets into a payload, or status information, deliverable to the core
- Capability to generate "no-snoop required" transactions
- Detect unsupported TLP and invoke appropriate mechanisms for handling them
- Transaction level support for the switching and advanced communication applications

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

Ordering rules:

- PCI/PCI-X compliant producer consumer ordering model
- Extensions to support relaxed ordering

Power management services:

- ACPI/PCI power management, as dictated by system software.
- Hardware-controlled autonomous power management minimizes power during full-on power states.

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 class of applications.
- Virtual Channels: Virtual Channels provide a means to support multiple independent logical data flows over a 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. Packets with different labels do not have ordering requirements among each other and that allows independent traffic flows that are not subject of global blocking conditions.

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 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 acknowledgment and retry messages
- Error indication for error reporting and logging
- Link ACK timeout mechanism

1.5.4.3. Physical Layer Services

Interface initialization, maintenance control, and status tracking:

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

Symbol and special ordered-set generation:

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

Symbol transmission and alignment:

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

System DFT mechanism(s):

- Loop-back mode

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

1.6. Advanced Peer-to-Peer Communication Overview

Advanced peer-to-peer communication is an optional functionality used to support peer-to-peer communications across one or more hierarchies that constitute a single fabric instance. Figure 1-6 shows an example of a fabric with multiple hierarchies.

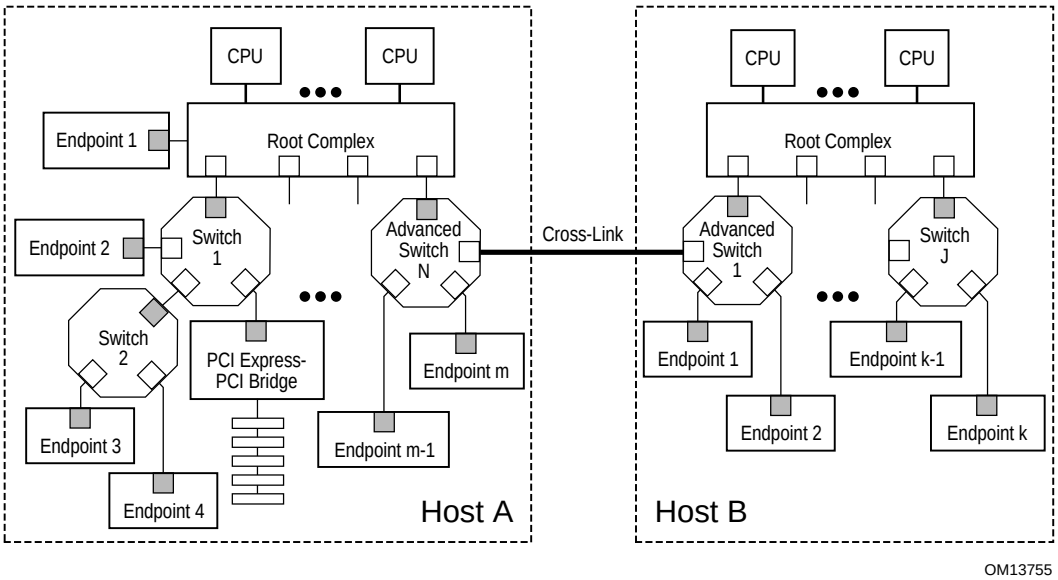

Figure 1-6: Advanced Peer-to-Peer Communication

The primary attributes/requirements are:

- Push-only communications paradigm – Endpoints use a "mailbox" approach to exchange control and data packets.
- Optional support for multicast packet replication within advanced Switch components. Multicast allows an Endpoint to inject a single packet into the fabric targeted at a multicast group identifier and have the advanced Switch components replicate this packet to all participating Endpoints. This eliminates the need for the injecting Endpoint having to know all of the participating Endpoints within the multicast group.
- Uses a 16-bit global address space extender to uniquely identify an Endpoint port or a multicast group within an I/O fabric. A global address is referred to as a Route Identifier.
- Each hierarchy defines an individual partition within the fabric.
    - At any given time, one Root Complex (RC) controls a partition.
    - Multiple partitions may be collapsed into a single partition by assigning ownership of the partition to one of the active RC. For example, dual-redundant fabrics that are inter-linked such that either RC may take over for the other should one RC fail.

- Cross-Link devices are used to facilitate non-tree enumeration and peer-to-peer communications. Software, using Cross-Link devices connected between Switches, establishes peer-to-peer connections between PCI Express agents residing either within the same hierarchy, or between agents that reside within different PCI Express hierarchies.
- Communication between two hierarchies is not allowed until both the hierarchies are initialized and configured. At this point, communications software can configure the Links between the hierarchies (contained within advanced Switch components). During this step, RIDs are assigned.
- RC, Switches, and Endpoints that support advanced peer-to-peer communication must support all mandatory PCI Express functionality to ensure interoperability with base RC, Switches, and Endpoints.
- A Switch that supports advanced peer-to-peer communication must translate RID forwarded packets to address-based routed packets if the attached RC or Endpoint does not support advanced peer-to-peer communication.

The scope of the information provided in this base specification is limited to providing definition of basic primitives required to support advanced PCI Express packet switching applications. Detailed description of typical usage models, and operation of the capabilities enabled by these optional features are beyond the scope of this document and will be described in a separate document, called *Advanced PCI Express Packet Switching Specification,* a companion specification to the PCI Express Base Specification.

2

2. Transaction Layer Specification

2.1. Transaction Layer Overview

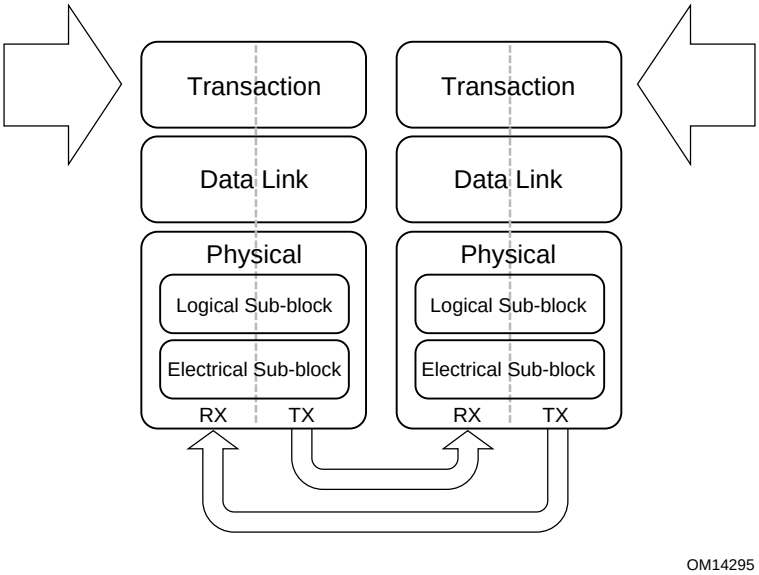

Figure 2-1: Layering Diagram Highlighting the Transaction Layer

One of the primary goals of the PCI Express Architecture is to maximize the efficiency of communication between devices. To this end, the Transaction Layer implements:

- A pipelined full split-transaction protocol
- Mechanisms for differentiating the ordering and processing requirements of Transaction Layer Packets (TLPs)
- Credit-based flow control which eliminates wasted Link bandwidth due to retries
- Optional support for data poisoning and end-to-end data integrity detection.

The Transaction Layer comprehends the following:

- TLP construction and processing
- Association of PCI Express transaction-level mechanisms with device resources including:
    - Flow Control
    - Virtual Channel management
- Rules for ordering and management of TLPs
    - Including Traffic Class differentiation

This chapter specifies the behaviors associated with the Transaction Layer.

2.2. Address Spaces, Transaction Types, and Usage

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

Table 2-1: Transaction Types for Different Address Spaces

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

2.2.1. Memory Transactions

Memory Transactions include the following types:

- Read Request/Completion
- Write Request

Memory Transactions use two different address formats:

- Short Address Format: 32-bit address
- Long Address Format: 64-bit address

Details about the rules associated with usage of these two address formats and the associated Transaction Layer Packet (TLP) formats are outlined in Section 2.7.

2.2.2. I/O Transactions

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

- Read Request/Completion
- Write Request/Completion

I/O Transactions use a single address format:

- Short Address Format: 32-bit address

Details about the rules associated with I/O address, and the associated TLP formats are outlined in Section 2.7.

2.2.3. Configuration Transactions

Configuration Transactions are used to access configuration registers of PCI Express devices. Mechanisms for generating these Transactions are platform specific.

Configuration Transactions include the following types:

- Read Request/Completion
- Write Request/Completion

Details about the rules associated with configuration address and the associated Packet formats are outlined in Section 2.7.

2.2.4. Message Transactions

The Message Transactions, or simply Messages, support two primary usage models:

- In-band communication of events between PCI Express devices
- Peer-to-peer communication between PCI Express devices

These two usage models map to two different groups of Messages in PCI Express. The first group supports the first usage model. The second group, which is associated with Advanced Switching support supports the second usage model.

In the terms of how Message Requests are routed, this specification differentiates between the following two routing mechanisms:

- Implied Routing – without specific address/routing information contained within Message packet header
    - Destination is the other component on the Link or
    - Destination is the Root Complex.
    - Message is broadcast from the Root Complex to all downstream devices.
- Explicit Routing – with specific address/routing information contained within Message packet header
    - Destination is another device within local PCI Express hierarchy or
    - Destination is a device within a different PCI Express hierachy

  Note that the explicit routing mechanism is used by the Messages that are defined for support of advanced switching applications.

PCI Express provides support for vendor-defined messages using specific reserved codes given in this document. The definition of specific vendor-defined messages is outside the scope of this document.

2.2.4.1. Types of Messages

Messages defined within the PCI Express specification include the following types of Messages:

- System Management Message Group
    - Interrupt Signaling
    - Error Signaling
    - Power Management
    - Locked Transaction Support
    - Payload Defined
    - Vendor Specific Messages
    - Hot Plug Signaling
- Advanced Switching Support Message Group
    - Data Packet Messages
    - Signal Packet Messages

2.2.4.2. Vendor-defined Messages

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 2.8.1.5 and 2.8.1.7).

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

2.3. Packet Format Overview

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

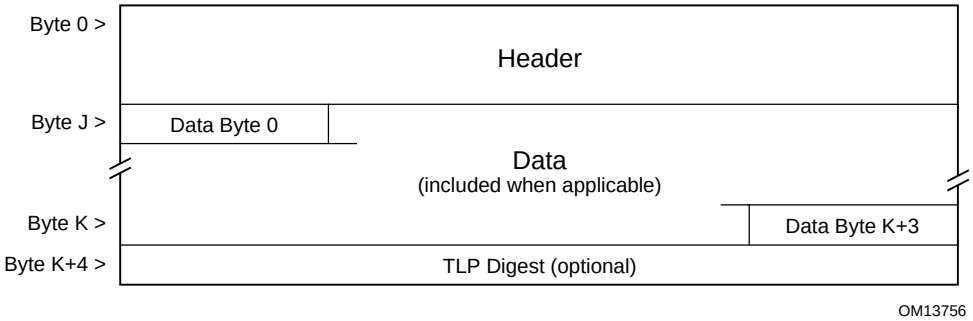

Figure 2-2: Generic Transaction Layer Packet Format

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

2.4. Transaction Descriptor

2.4.1. 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 2-3 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.

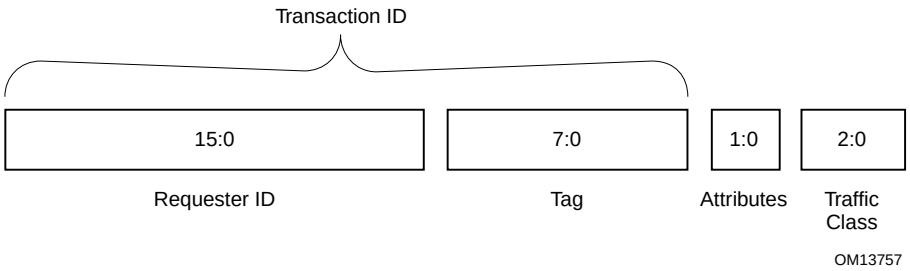

Figure 2-3: Transaction Descriptor

2.4.2. Transaction Descriptor – Transaction ID Field

The Transaction ID Field consists of two major sub-fields: Requester ID and Tag as shown in Figure 2-4.

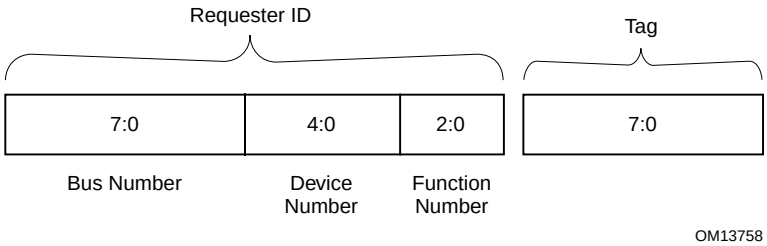

Figure 2-4: Transaction ID

- Tag[7:0] is a 8-bit field generated by each Requestor, and it must be unique for all outstanding Requests that require a Completion for that Requester
    - By default, the maximum number of outstanding Requests per device/function shall be limited to 32, and only the lower 5 bits of the Tag field are used with the remaining 3 required to be all 0's
    - If the Extended Tag Field Enable bit (see Section 5.8.4) is set, the maximum is increased to 256, and the entire Tag field is used
    - Receiver/Completer behavior is undefined if multiple Requests are issued non-unique Tag values
- For Requests which do not require Completion (Posted Requests), the value in the Tag[7:0] field is undefined and may contain any value
    - For Posted Requests, the value in the Tag[7:0] field must not affect Receiver processing of the Request
- Requester ID and Tag combined form a global identifier for each Transaction within a Hierarchy.
- Transaction ID is included with all Requests and Completions.
- The Requester ID field is a 16-bit value that is unique for every PCI Express function.
- Functions must capture the Bus and Device Numbers supplied with all Configuration Requests (Type 0) completed by the function and supply these numbers in the Bus and Device Number fields of the Requester ID for all Requests initiated by the device/function.
    - Note that the Bus Number and Device Number may be changed at run time, and so it is necessary to re-capture this information with each and every Configuration Request.
    - Exception: The assignment of bus numbers to the logical devices within a Root Complex may be done in an implementation specific way.

  Example: When a device (or function of a multi-function device) receives a Type 0 Configuration Read or Write Request, the device comprehends that it is the intended recipient of the Request because it is a Type 0 Request. The routing information fields of the Request include the recipient's Bus Number and Device Number values (Figure 2-12). These values are captured by the device and used to generate the Requester ID field.
- Prior to the initial Configuration Write to a device, the device is not permitted to initiate Requests.
    - Exception: Logical devices within a Root Complex are permitted to initiate Requests prior to software initiated configuration for accesses to system boot device(s).
    - Note that this rule and the exception are consistent with the existing PCI model for system initialization and configuration.

- Each function associated with a logical device must be designed to respond to a unique Function Number for Configuration Requests addressing that logical device. Note: Each logical device may contain up to eight logical functions.
- A Switch must forward Requests without modifying the Transaction ID
- A PCI Express-PCI-X Bridge operating in conventional PCI mode as well as a PCI Express-PCI Bridge must forward Requests initiated on PCI using the Bus Number, Device Number, and Function Number associated with the Bridge to form the Requester ID.

Implementation Note: Increasing Outstanding Requests

To increase the maximum possible number of outstanding Requests requiring Completion beyond 256, a single function device may, if the Phantom Function Number Enable bit is set (see Section 5.8.4), use Function Numbers 1-7 to logically extend the Tag identifier, allowing up to a 8-fold increase in the maximum number of outstanding Requests

Unclaimed function numbers are termed "Phantom Function Numbers (PFN)."

2.4.3. 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.

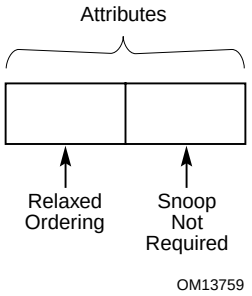

Figure 2-5: Attributes Field of Transaction Descriptor

2.4.3.1. Relaxed Ordering Attribute

Table 2-2 defines the states of the Relaxed Ordering attribute field. This Attribute is discussed in Section 2.5.

Table 2-2: Ordering Attributes

Ordering Attribute	Ordering Type	Ordering Model
0	Default Ordering	PCI Producer/Consumer-based Ordering Model
1	Relaxed Ordering	PCI-X Relaxed Ordering Model

2.4.3.2. "Snoop Not Required" Attribute

Table 2-3 defines the states of the "Snoop Not Required" attribute field. Note that the "Snoop Not Required" attribute does not alter Transaction ordering.

Table 2-3: Cache Coherency Management Attribute

Snoop Not Required Attribute	Cache Coherency Management Type	Coherency Model
0	Default	Hardware enforced cache coherency expected
1	Snoop Not Required	Hardware enforced cache coherency *not* expected

2.4.4. 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 2.6 covers the details of the VC mechanism.)

Table 2-4 defines the TC encodings:

Table 2-4 Definition of TC Field Encodings

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

It is up to the system software to determine TC labeling and TC/VC mapping in order to provide differentiated services that meet target platform requirements. For example, for a platform that supports isochronous data traffic, TC7 is reserved for isochronous transactions and TC7 must be mapped to the VC with the highest weight/priority. See Section 7.3.4 for details on isochronous support.

The concept of Traffic Class applies only within the PCI Express interconnect fabric. Specific requirements of how PCI Express TC service policies are translated into policies on non-PCI Express interconnects or within Root Complex or Endpoints is outside of the scope of this specification.

2.5. Transaction Ordering

Table 2-5 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 among transactions within different TCs. 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 2-5, the columns represent a first issued Transaction, and the rows represent a subsequently issued Transaction. The table entry indicates the ordering relationship between the two Transactions. The table entries are defined as follows:

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

Table 2-5: Ordering Rules Summary Table

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

Explanation of entries in Table 2-5:

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

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

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

A5, A6 a — Endpoints, Switches, and Root Complex may allow Memory Write and Message Requests to pass Completions or be blocked by Completions.

A5, A6 b — PCI Express to PCI Bridges and PCI Express to PCI-X Bridges, when operating PCI segment in conventional mode, must allow Memory Write and Message Requests to pass Completions traveling in the PCI Express to PCI direction (Primary side of Bridge to Secondary side of Bridge) to avoid deadlock.

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

B3, B4, C3, C4	Read Requests and I/O or Configuration Write Requests are permitted to be blocked by or to pass other Read Requests and I/O or Configuration Write Requests.
B5, B6, C5, C6	These Requests are permitted to be blocked by or to pass Completions.
D2 a	If the Relaxed Ordering attribute bit is not set, then a Read Completion cannot pass a previously enqueued Memory Write or Message Request.
D2 b	If the Relaxed Ordering attribute bit is set, then a Read Completion is permitted to pass a previously enqueued Memory Write or Message Request.
D3, D4, E3, E4	Completions must be allowed to pass Read and I/O or Configuration Write Requests to avoid deadlocks.
D5 a	Read Completions associated with different Read Requests are allowed to be blocked by or to pass each other.
D5 b	Read Completions for one Request (will have the same Transaction ID) must return in address order.
D6	Read Completions are permitted to be blocked by or to pass I/O or Configuration Write Completions.
E2	I/O or Configuration Write Completions are permitted to be blocked by or to pass Memory Write and Message Requests. Such Transactions are actually moving in the opposite direction and, therefore, have no ordering relationship.
E5, E6	I/O or Configuration Write Completions are permitted to be blocked by or to pass Read Completions and other I/O or Configuration Write Completions.

Additional Rules:

- PCI Express Switches are permitted to allow a Memory Write or Message Request with the Relaxed Ordering bit to set 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). PCI Express-PCI-X Bridge devices must forward the Relaxed Ordering attribute unmodified but must treat all transactions as if the Relaxed Ordering attribute bit is not set.

  Note: This maintains compatibility with PCI-X relaxed ordering usage models and corresponding rules. For more details, refer to the *PCI-X Addendum to the PCI Local Bus Specification, Rev 1.0a*.

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

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

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

- The "Snoop Not Required" bit does not affect the required ordering behavior.

  Note: Main memory writes from the CPU accepted by the Root Complex are architecturally part of the system memory image; the Root Complex must ensure coherency for subsequent device reads from main memory.

Implementation Note: Large Memory Reads vs. Multiple Smaller Memory Reads

Note that the rule associated with entry D5b in Table 2-5 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 256B from location 1000h, and the Request is returned using two Completions (see Section 2.7.6.2.1) of 128B 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 128B 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.

2.6. Virtual Channel (VC) Mechanism

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

Traffic is associated with VCs by mapping packets with particular TC labels to their corresponding VCs. The PCI Express VC mechanism allows flexible mapping of TCs onto the VCs. In the simplest form, TCs can be mapped to VCs on a 1:1 basis. To allow performance/cost tradeoffs, PCI Express provides the capability of mapping multiple TCs onto a single VC. Section 2.6.3 covers details of TC to VC mapping.

A Virtual Channel is established when one or multiple TCs are associated with physical VC resource designated by VC ID. This process is controlled by the PCI Express configuration software as described in Sections 5.11 and 7.3.

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

- A Requester must only generate requests with TC0 label. (Note that if it initiates requests with a TC label other than TC0, the requests may be treated as illegal by the component on the other side of the Link that implements the extended VC capability and applies TC filtering.)
- A Completer must accept requests with TC label other than TC0, and must preserve the TC label, i.e., any completion that it generates must have the same TC label as the label of the request.
- A Switch must map all TCs to VC0 and must forward all transactions regardless of the TC label.

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

Figure 2-6 illustrates the concept of Virtual Channel. The enlarged area shows VC resources in one direction (Switch to RC). Conceptually, traffic that flows through VCs is muxed onto a common physical Link resource on the transmit side and de-muxed into separate VC paths on the receive side.

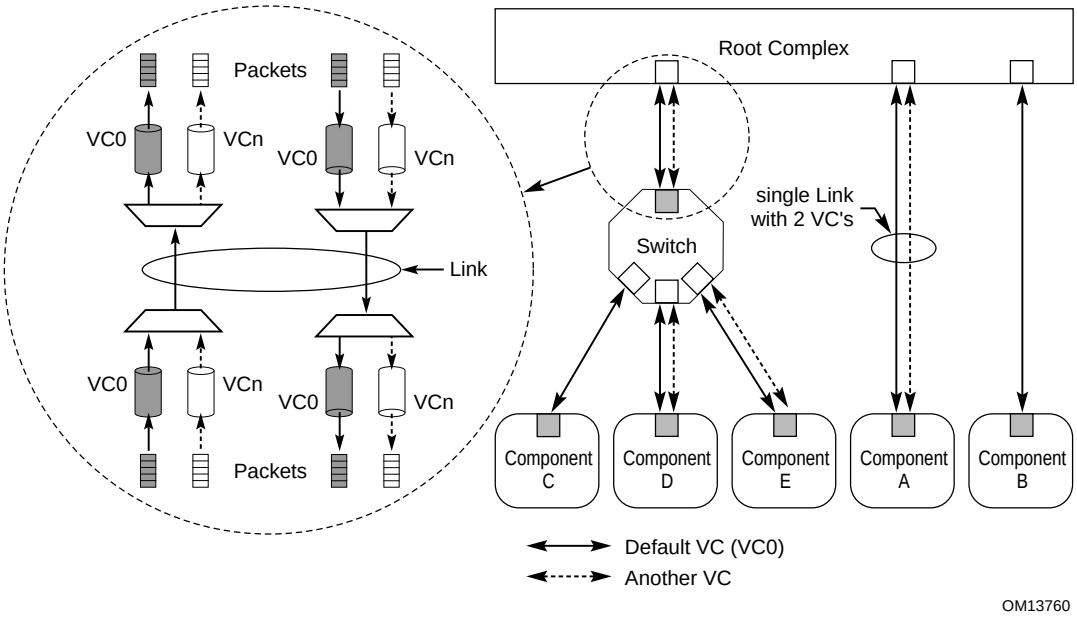

Figure 2-6: Virtual Channel Concept – An Illustration

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 2-7 shows conceptually the VC resources within the Switch (shown in Figure 2-6) that are required to support traffic flow in the upstream direction.

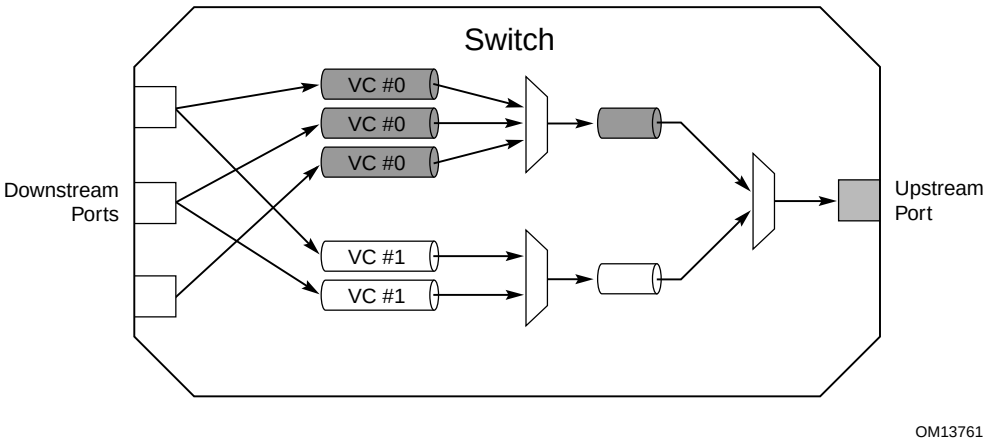

Figure 2-7: Virtual Channel Concept – Switch Internals (Upstream Flow)

2.6.1. Virtual Channel Identification (VC ID)

A PCI Express Port can support up to eight Virtual Channels. These VCs are uniquely identified using the Virtual Channel Identification (VC ID) mechanism.

Note that TLPs do not include VC ID information. The association of TLPs with VC ID for the purpose of Flow Control accounting is done at each Port of the Link using TC to VC mapping as discussed in Section 2.6.3.

All PCI Express Ports that support more than VC0 must provide the VC Capability Structure according to the definition in Section 5.11. Providing this extended structure is optional for Ports that support only the default TC0/VC0 configuration. PCI Express configuration software is responsible for configuring Ports on both sides of the Link for a matching number of VCs. This is accomplished by scanning the PCI Express hierarchy and using VC Capability registers associated with ports (that support more than default VC0) to establish number of VCs for the Link. Rules for assigning VC ID for VC hardware resources are as follows:

- VC ID assignment must be unique per PCI Express Port – Same VC ID cannot be assigned to different VC hardware resources within the same Port.
- VC ID assignment must be the same (matching in the terms of numbers of VCs and their IDs) for the two PCI Express Ports on both sides of a PCI Express Link.
- VC ID 0 is assigned and fixed to the default VC.
- For a PCI Express Port that supports the VC Capability Structure, the first VC hardware resource must be the default VC.
- VC ID assignment must be in increasing order (but not necessarily contiguous) for a PCI Express Port that supports multiple VCs.

2.6.2. VC Support Options

To simplify the interoperability when configuring number of supported VCs per Link, the PCI Express specification limits the set of valid VC configuration options to: 1, 2, 4, and 8. Other VC configurations such as 3, 5, 6, and 7 are not allowed.

2.6.3. 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 PCI Express Port.
- TC/VC mapping must be identical for PCI Express Ports on both sides of a PCI Express Link.
- For any two TCs (TCb > TCa), TCb must be mapped to the same VC as TCa or be mapped to a VC with higher VC ID. It is not allowed to map TCb on a VC with a lower VC ID than the one TCa is mapped to.

Table 2-6 provides an example of TC to VC mapping.

Table 2-6: 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 2-8 provides a graphical illustration of TC to VC mapping in several different Link configurations. For additional considerations on TC/VC mapping including symmetrical and asymmetrical mapping within PCI Express Switches, refer to Section 7.3.

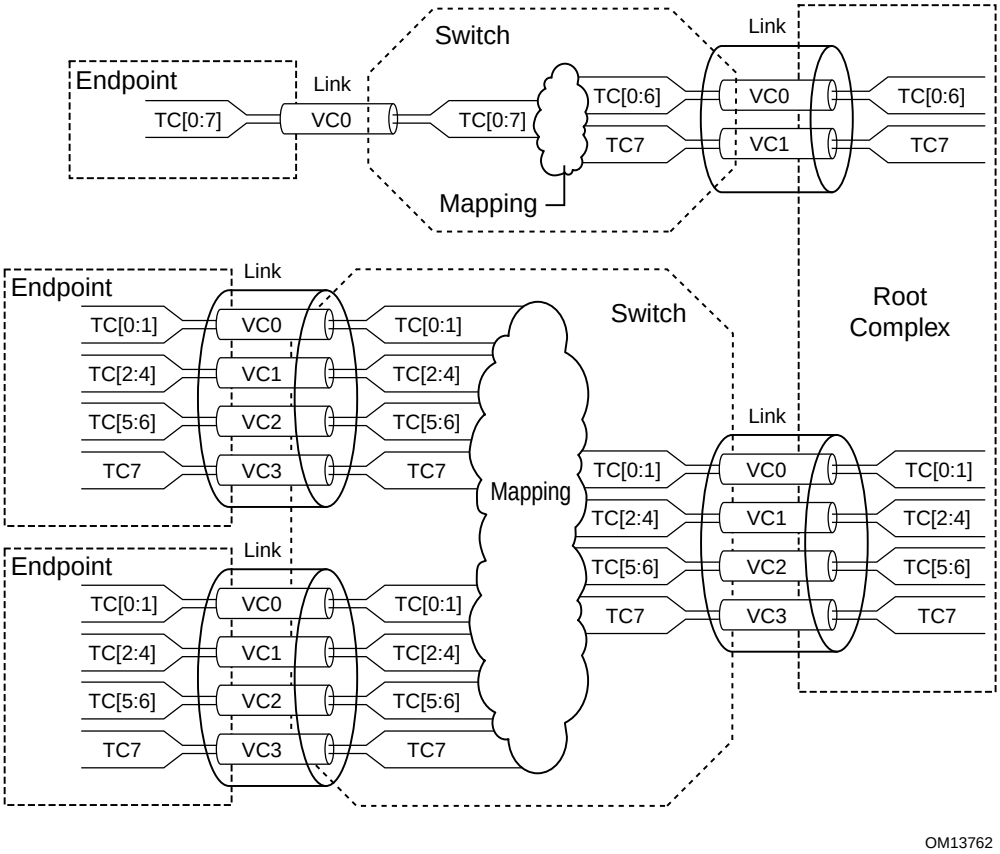

Figure 2-8: An Example of TC/VC Configurations

2.6.4. VC and TC Rules

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

- All PCI Express devices must support 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.
- Transactions with TC that is not mapped to any enabled VC in a PCI Express Ingress Port are treated as malformed transaction by the receiving device.
- For Switches, transactions with TC that is not mapped to any of enabled VCs in the target Egress Port are treated as illegal transaction.

- For a Root Port, transactions with a TC that is not mapped to any of enabled VCs in the target RCRB are treated as illegal transaction.
- Switches must support independent TC/VC mapping configuration for each port.
- Root Complex must support independent TC/VC mapping configuration for each RCRB and the associated Root Ports.

For more details on the VC and TC mechanisms, including configuration, mapping, and arbitration, refer to Chapter 7.3.

2.7. Transaction Layer Protocol - Packet Definition and Handling

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 Requester ID field of the Packet header.

2.7.1. Transaction Layer Packet Definition Rules

- All Transaction Layer Packets (TLPs) must start with one of the headers defined in this section.
    - Some TLPs include data following the header as determined by the Fmt[1:0] field specified in the TLP header.
- TLP data must be four-byte naturally aligned and in increments of four-Byte Double Words (DW).
- All TLP headers include the following fields:
    - Fmt[1:0] – Specifies global Format of TLP:
        - 00 - 3DW header, no data
        - 01 - 4DW header, no data
        - 10 - 3DW header, with data
        - 11 - 4DW header, with data

- Type[4:0] – See Table 2-8 for type encodings
  - Both Fmt[1:0] and Type[4:0] must be decoded to determine specifics of TLP format.
- Length[9:0] – Length of data payload in DW
  - 00 0000 0001 = 1DW
  - 00 0000 0010 = 2DW
  - …..
  - 11 1111 1111 = 1023DW
  - 00 0000 0000 = 1024DW
- Permitted Fmt[1:0] and Type[4:0] field values are shown in Table 2-8.
  - All other encodings are reserved.
- TD - '1' indicates presence of TLP "digest" in the form of a single DW at the end of the TLP (see Figure 2-2)
- EP -
  - If TD='1', EP ='0' means TLP digest is used for data poisoning; EP='1' means TLP digest is used for and end-to-end CRC (ECRC) field
  - If TD='0', EP='0' means TLP is not poisoned, EP='1' means TLP is poisoned
  - Thus, the combination of TD and EP is interpreted as shown in Table 2-7

62

Table 2-7: TD and EP Field Values

TD	EP	TLPD Digest	End-to-End Data Integrity	Error Forwarding	Digest Value	Comments
0	0	Not present	No	No	N/A	
0	1	Not present	No	Yes	N/A	Poisoned data See Section 2.11
1	0	Present	No	Yes	FFFFFFFFh	Poisoned data
					Otherwise	Not poisoned data
1	1	Present	Yes	Yes	Valid ECRC See Section 2.10	No errors
					FFFFFFFFh[2]	Poisoned data
					Otherwise	ECRC error

Different types of TLPs are discussed in more detail in the following sections.

Table 2-8: Fmt[1:0] and Type[4:0] Field Encodings

TLP Type	Fmt [1:0][3]	Type [4:0]	Description
MRd	00 01	0 0000	Memory Read Request
MRdLk	00 01	0 0001	Memory Read Request–Locked
MWr	10 11	0 0000	Memory Write Request
IORd	00	0 0010	I/O Read Request
IOWr	10	0 0010	I/O Write Request
CfgRd0	00	0 0100	Configuration Read Type 0
CfgWr0	10	0 0100	Configuration Write Type 0
CfgRd1	00	0 0101	Configuration Read Type 1
CfgWr1	10	0 0101	Configuration Write Type 1

[2] Note that FFFFFFFFh cannot occur as a valid ECRC value.

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

TLP Type	Fmt [1:0][3]	Type [4:0]	Description
Msg	01	1 $0r_2r_1r_0$	Message Request – The sub-field r[2:0] specifies Message routing mechanism – See Table 2-9
MsgD	11	1 $0r_2r_1r_0$	Message Request with data payload – The sub-field r[2:0] specifies Message routing mechanism – See Table 2-9
MsgAS	01	1 $1n_2n_1n_0$	Message for Advanced Switching –The sub-field n[2:0] specifies the message type: $1n_2n_1n_0$ – Signaling Packet Messages *A detailed description of message types and message headers will be presented in a separate document entitled Advanced PCI Express Packet Switching Specification. This is a companion specification to the PCI Express Base Specification.*
MsgASD	11	1 $1c_2c_1c_0$	Message for Advanced Switching – The sub-field c[2:0] specifies the message type: $1c_2c_1c_0$ – Data Packet Messages *A detailed description of message types and message headers will be presented in a separate document entitled Advanced PCI Express Packet Switching Specification. This is a companion specification to the PCI Express Base Specification.*
Cpl	00	0 1010	Completion without Data – used for I/O and Configuration Write Completions, and Memory Read Completions with Completion Status other than Successful Completion”
CplD	10	0 1010	Completion with Data – used for Memory, I/O, and Configuration Read Completions
CplLk	00	0 1011	Completion for Locked Memory Read without Data – used only in error case
CplDLk	10	0 1011	Completion for Locked Memory Read – otherwise like CplD
			All encodings not shown above are Reserved

Table 2-9: Message Routing

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

2.7.2. TLP Digest Rules

- For any TLP, a value of '1' in the TD field indicates the presence of the TLP Digest field at the end of the TLP
  - The presence or absence of the TLP Digest field must be checked for all TLPs
  - A TLP with a '1' in the TD field but without a TLP Digest, or a TLP with a TLP Digest but without a '1' in the TD field, is a Malformed TLP
    - This is a reported error associated with the Receiving Port (see Section 7.2)
- For any TLP with a TLP Digest field, a value of '1' in the EP field indicates that the TLP Digest field is used for an end-to-end CRC (ECRC)
  - The presence or absence of the ECRC must be checked for all TLPs
- If the device at the ultimate destination of the TLP-
  - supports neither data poisoning nor ECRC checking, the device must ignore the TLP Digest
  - supports data poisoning but not ECRC checking, the device interprets the value in the TLP Digest field according to Section 2.11
  - supports ECRC checking, the device interprets the value in the TLP Digest field as an ECRC value, according to the rules in Section 2.10.2

[4] Similar to a Completion or a Configuration Request.

2.7.3. TLPs with Data Payloads - Rules

- Length is specified as a number of naturally aligned DW
- Length[9:0] is reserved for all Messages except those which explicitly refer to a Data Length
    - See Message Code table in Section 2.7.4.4.
- The data payload of a TLP must not exceed the length specified by the value in the Max_Payload_Size field of the Link Command Register (see Section 5.8.7).
    - 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
    - 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 7.2)
- For TLPs, that include data, the value in the Length field and the actual amount of data included in the TLP must be equal.
    - 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 7.2)
- Requests must not specify an Address/Length combination which causes a Memory Space access to cross a 4K 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 7.2)
- Note: The Length specified in the Length field applies only to data – the Transaction Digest is not included in the Length
- When a data payload is included in a TLP, the first Byte of data following the header corresponds to the Byte address closest to zero and the succeeding Bytes are in increasing Byte address sequence.
    - Example: For a 16B write to location 100h, the first byte following the header would be the byte to be written to location 100h, and the second byte would be written to location 101h, and so on, with the final byte written to location 10Fh.

Implementation Note: Maintaining Alignment in Data Payloads

Section 2.7.6.2.1 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.

2.7.4. Requests

Requests include a Request header which for some types of Requests will be followed by some number of DW of data. The rules for each of the fields of the Request header are defined in the following sections.

2.7.4.1. Address Field Rules

Two Address formats are specified, a 32b format and a 64b format. Figure 2-9 shows the Request header format for 32b Addressing and Figure 2-10 shows the Request header format for 64b Addressing.

- Memory Read Requests and Memory Write Requests can use either format.
  - For Addresses below 4 GB, Requesters must use the 32b format.
- I/O Read Requests and I/O Write Requests use the format shown in Figure 2-11.
- Configuration Read Requests and Configuration Write Requests use the format shown in Figure 2-12.
- Msg and MsgD Requests use the format shown in Figure 2-13
- MsgAS and MsgASD Requests use the format shown in Figure 2-15
- All PCI Express Agents must decode all address bits in the header - address aliasing is not allowed.

Implementation Note: Prevention of Address Aliasing

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

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0 >	R	Fmt x 0			Type				R	TC			Reserved				T D	E P	Attr		R				Length							
Byte 4 >	Requester ID																Tag								Last DW BE				1st DW BE			
Byte 8 >	Address[31:2]																															R

OM13763

Figure 2-9: Request Header Format for 32b Addressing of Memory

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0 >	R	Fmt x 1			Type				R	TC			Reserved				T D	E P	Attr		R				Length							
Byte 4 >	Requester ID																Tag								Last DW BE				1st DW BE			
Byte 8 >	Address[63:32]																															
Byte 12 >	Address[31:2]																															R

OM13764

Figure 2-10: Request Header Format for 64b Addressing of Memory

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0 >	R	Fmt x 0			Type				R	TC 0 0 0			Reserved				T D	E P	Attr 0 0		R				Length 0 0 0 0 0 0 0 0 0 1							
Byte 4 >	Requester ID																Tag								Last DW BE 0 0 0 0				1st DW BE			
Byte 8 >	Address[31:2]																															R

OM13765

Figure 2-11: Request Header Format for I/O Transactions

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0 >	R	Fmt x 0			Type				R	TC 0 0 0			Reserved				T D	E P	Attr 0 0		R				Length 0 0 0 0 0 0 0 0 0 1							
Byte 4 >	Requester ID																Tag								Last DW BE 0 0 0 0				1st DW BE			
Byte 8 >	Bus Number								Device Number						Function Number		Reserved				Ext. Reg. Address				Register Address							R

OM13766

Figure 2-12: Request Header Format for Configuration Transactions

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0 >	R	Fmt 0	1	Type					R	TC			Reserved				T D	E P	Attr 0	0	R				Length - Reserved 0	0	0	0	0	0	0	0
Byte 4 >	Requester ID																Tag								Message Code							
Byte 8 >	Address[63:32]/Reserved																															
Byte 12 >	Address[31:2]/Reserved																															R

OM13767

Figure 2-13: Request Header Format for Msg Request

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0 >	R	Fmt 1	1	Type					R	TC			Reserved				T D	E P	Attr 0	0	R				Length							
Byte 4 >	Requester ID																Tag								Message Code							
Byte 8 >	Address[63:32]/Reserved																															
Byte 12 >	Address[31:2]/Reserved																															R

OM14296

Figure 2-14: Request Header Format for MsgD Request

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0 >	R	Fmt 0	1	Type					R	TC			Reserved				T D	E P	Attr		R				Length–Reserved 0	0	0	0	0	0	0	0
Byte 4 >	Reserved for MsgAS																															
Byte 8 >	(A detailed description of message types and message headers will be included in a separate document, called the *Advanced PCI Express Packet Switching Specification*, a companion specification to the *PCI Express Base Specification*.)																															
Byte 12 >																																

OM13768

Figure 2-15: Request Header Format for MsgAS Request

	+0								+1								+2								+3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Byte 0 >	R	Fmt 1	1	Type					R	TC			Reserved				T D	E P	Attr		R				Length							
Byte 4 >	Reserved for MsgASD																															
Byte 8 >	(A detailed description of message types and message headers will be included in a separate document, called the *Advanced PCI Express Packet Switching Specification*, a companion specification to the *PCI Express Base Specification*.)																															
Byte 12 >																																

OM14298

Figure 2-16: Request Header Format for MsgASD Request

2.7.4.2. First/Last DW Byte Enable Rules

- The first DW Byte Enables[3:0] Field contains byte enables for the first DW of any Memory Read or Write Request, and for the only DW of an I/O or Configuration Request.
  - If there is only one DW for a Memory Request, this byte enable field is used.
  - If the Length field for a Request indicates a length of greater than 1DW, this field must not be inactive (must not equal 0000b)
- The Last DW Byte Enables[3:0] Field contains byte enables for the last DW of any Memory Read or Write Request.
  - If the Length field for a Request indicates a length of 1DW, this field must be inactive (must equal 0000b).
  - If the Length field for a Request indicates a length of greater than 1DW, this field must not be inactive (must not equal 0000b).
- These fields are never used with Msg, MsgD, MsgAS, or MsgASD Requests
  - Note that these fields overlap the Message Code/Dest RID[7:0] fields
- For each bit of the Byte Enables Fields:
  - a value of '0' 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 '1' indicates that the corresponding Byte of Data must be written or read at the Completer.
- If a Read Request of 1 DW specifies that no Bytes are enabled to be read (1st DW Byte Enables[3:0] field = b'0000), the corresponding Completion must specify a Length of 1 DW, and include a data payload of 1 DW
  - The contents of the data payload are unspecified and may be any value
- Receiver/Completer behavior is undefined for a TLP violating the Byte Enables rules specified in this section.
- Receivers may optionally check for violations of the Byte Enables rules specified in this section. If a Receiver implementing such checks determines that a TLP violates one or more Byte Enable rules, the TLP is a Malformed TLP
  - If Byte Enable rules are checked, a violation is a reported error associated with the Receiving Port (see Section 7.2)

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.

The "flush" semantic has wide application, and all Completers must implement the functionality associated with this semantic. Because a Requester may use the "flush" semantic without comprehending the characteristics of the Completer, Completers must ensure that zero length reads do not have side-effects. This is really just a specific case of the rule that in a non-prefetchable space, non-enabled Bytes must not be read at the Completer. Note that the "flush" applies only to traffic in the same Traffic Class as the zero length Read.

- Of the first DW Byte Enables[3:0] Field:
    - Bit 0 corresponds to Byte 0 of the first DW of data.
    - Bit 1 corresponds to Byte 1 of the first DW of data.
    - Bit 2 corresponds to Byte 2 of the first DW of data.
    - Bit 3 corresponds to Byte 3 of the first DW of data.
- Of the last DW Byte Enables[3:0] Field:
    - Bit 0 corresponds to Byte 0 of the last DW of data.
    - Bit 1 corresponds to Byte 1 of the last DW of data.
    - Bit 2 corresponds to Byte 2 of the last DW of data.
    - Bit 3 corresponds to Byte 3 of the last DW of data.

Figure 2-9, Figure 2-10, Figure 2-11, and Figure 2-12 show the Byte Enable fields for Memory, I/O, and Configuration Requests.

2.7.4.3. Rules for Tag, Requester ID, Traffic Class, and Attribute Fields

- The Tag[7:0] field contains the Tag as described in Section 2.4.2.
- The Requester ID[15:0] field contains the Requester ID as described in Section 2.4.2.
- The TC[2:0] field contains the Traffic Class identification as described in Section 2.4.4.
- The Attr[1:0] field contains the Transaction Descriptor attribute as described in Section 2.4.3.

Figure 2-9, Figure 2-10, Figure 2-11, Figure 2-12, Figure 2-13, and Figure 2-15 show these fields.

2.7.4.4. Message Space Rules

- Message codes and support requirements are defined in Table 2-10
- All devices must fully decode all Messages to distinguish supported Messages from unsupported Messages (aliasing is not permitted).
- Message Requests are posted and do not require Completion.
- Message Requests follow the same ordering rules as Memory Write Requests.
- Except as noted, the Address field is Reserved
- Except as noted, the Attr(Attribute) field is set to 00b
- Except as noted, Messages use Traffic Class = 0
    - 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 7.2)
- Message Codes in the range 10000000 – 11111111 are reserved for Vendor Specific use
- Receipt of an unsupported Message is an Unsupported Request
    - An Unsupported Request is a reported error associated with the Receiving device/function (see Section 7.2)
    - Note that many Messages are specified to be simply discarded by the Receiver without effect – such Messages are not considered Unsupported Requests, and are, therefore, not errors

      Example: A PCI Express Endpoint receiving an Unlock Message.

Table 2-10: Msg Codes

Name	Code[7:0]	Routing r[2:0]	Support[5]				Req ID[6]	Description/Comments
			RC	Ep	Sw	Br		
Unlock	0000 0000	011	t	r		r	BD	Unlock Completer
ERR_COR	0011 0000	000	r	t		t	B BD BDF	Signal detection of a correctable error
ERR_NONFATAL	0011 0001	000	r	t		t	B BD BDF	Signal detection of an uncorrectable error
ERR_FATAL	0011 0011	000	r	t		t	B BD BDF	Signal detection of a fatal error
PM_Active_State_Nak	0001 0100	100	t	r	tr	r	B	Power Management related – see Chapter 6
PM_PME	0001 1000	000	If PME supported:				BDF	Power Management related – see Chapter 6
				t				
PME_Turn_Off	0001 1001	011	t	r		r	BDF	Power Management related – see Chapter 6

[5] Abbreviations:
RC = Root Complex
Sw = Switch (only used with “Link” routing)
Ep = Endpoint
Br = PCI Express/PCI Bridge

r = Supports as Receiver
t = Supports as Transmitter

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

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

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

Name	Code[7:0]	Routing r[2:0]	Support				Req ID	Description/Comments
			RC	Ep	Sw	Br		
PME_TO_Ack	0001 1011	000	r	t		t	BDF	Power Management related – see Chapter 6
			(Note: Switch handling is special)					
Assert_INTA	0010 0000	100	All:				B	Assert INTA virtual signal
			r					Note: These Messages are used for PCI 2.3 compatible INTx emulation
			As Required:					
				t		t		
Assert_INTB	0010 0001	100	All:				B	Assert INTB virtual signal
			r					
			As Required:					
				t		t		
Assert_INTC	0010 0010	100	All:				B	Assert INTC virtual signal
			r					
			As Required:					
				t		t		
Assert_INTD	0010 0011	100	All:				B	Assert INTD virtual signal
			r					
			As Required:					
				t		t		
Deassert_INTA	0010 0100	100	All:				B	De-assert INTA virtual signal
			r					
			As Required:					
				t		t		
Deassert_INTB	0010 0101	100	All:				B	De-assert INTB virtual signal
			r					
			As Required:					
				t		t		
Deassert_INTC	0010 0110	100	All:				B	De-assert INTC virtual signal
			r					
			As Required:					
				t		t		
Deassert_INTD	0010 0111	100	All:				B	De-assert INTD virtual signal
			r					
			As Required:					
				t		t		

Name	Code[7:0]	Routing r[2:0]	Support				Req ID	Description/Comments
			RC	Ep	Sw	Br		
Attention_Indicator_On	0100 0001	100	t	r	tr	r	BDF	Attention Indicator On
			Required for Hot Plug Support					
Attention_Indicator_Blink	0100 0011	100	t	r	tr	r	BDF	Attention Indicator Blink
			Required for Hot Plug Support					
Attention_Indicator_Off	0100 0000	100	t	r	tr	r	BDF	Attention Indicator Off
			Required for Hot Plug Support					
Power_Indicator_On	0100 0101	100	t	r	tr	r	BDF	Power Indicator On
			Required for Hot Plug Support					
Power_Indicator_Blink	0100 0111	100	t	r	tr	r	BDF	Power Indicator Blink
			Required for Hot Plug Support					
Power_Indicator_Off	0100 0100	100	t	r	tr	r	BDF	Power Indicator Off
			Required for Hot Plug Support					
Attention_Button_Pressed	0100 1000	100	r	t	r	t	BDF	Attention Button Pressed
			Required for Hot Plug Support					
Vendor Specific	1000 0000 to 1111 1111	000 to 100[7]	See note.				See note.	Codes in this range are reserved for vendor definition.

Note: Implementation specific.

[7] Any value in this range is permitted.

Table 2-11: MsgD Codes

Name	Code[7:0]	Routing r[2:0]	Support				Req ID	Description/Comments
			RC	Ep	Sw	Br		
Set_Slot Power_Limit	0101 0000	100	t	r	tr	r	BDF	Set Slot Power Limit in Upstream Port
Payload_Defined	0111 1111	000 to 100	See note.				See note.	See Section 2.8.1.5
Vendor Specific	1000 0000 to 1111 1111	000 to 100	See note.				See note.	Codes in this range are reserved for vendor definition.

Note: Implementation specific.

2.7.5. Completions

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

Figure 2-17 shows the format of a Completion header.

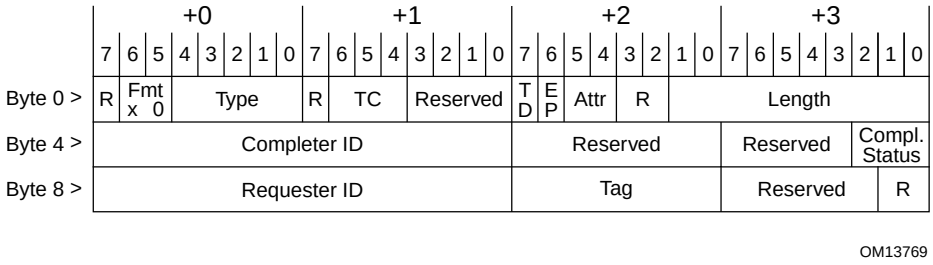

Figure 2-17: Completion Header Format

2.7.5.1. Rules for Completers

- The Completion Status[2:0] field indicates the status for a Completion:
  - 000b – Successful Completion (SC)
  - 001b – Unsupported Request (UR)
  - 010b – Configuration Request Retry Status (CRS)
  - 100b – Completer Abort (CA)
  - All others Reserved

- Rules for determining the value in the Completion Status[2:0] field are in Section 2.7.6.2
- The Completer ID[15:0] field is a 16-bit value that is unique for every PCI Express function (see Figure 2-18)
- Functions must capture the Bus and Device Numbers supplied with all Configuration Requests (Type 0) completed by the function, and supply these numbers in the Bus and Device Number fields of the Completer ID for all Completions generated by the device/function.
    - If a function must generate a Completion prior to the initial device Configuration 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 Request.
    - Exception: The assignment of bus numbers to the logical devices within a Root Complex may be done in an implementation specific way.
- In some cases, a Completion with the UR status may be generated by a multi-function device without associating the Completion with a specific function within the device – in this case, the Function Number field is Reserved, and is set to all '0's
    - 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

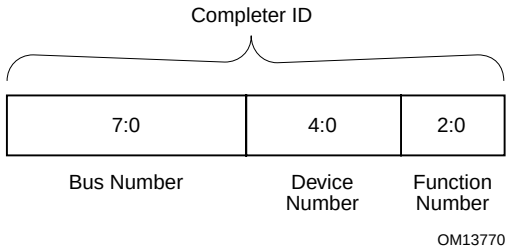

Figure 2-18: Completer ID

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

  Note: Prior to system initialization, Requester ID values may not be established. It is required that all Requests made prior to system initialization be initiated by the Root Complex, as all Completions will be routed to the Root Complex.
- The Completion ID field is not meaningful prior to the software initialization and configuration of the completing device (using at least one Configuration Write Request), and the Requestor must ignore the value returned in the Completer ID field.

- A Completion including data must specify the actual amount of data returned in that Completion, and must include the amount of data specified.
    - It is a TLP formation error to include more or less data than specified in the Length field, and the resulting TLP is a malformed TLP.

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

2.7.6. Handling of Received TLPs

2.7.6.1. Handling of Received TLPs – Rules

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 2-19.

- Values in Reserved fields must be ignored by the Receiver.
- All Received TLPs which fail the required (and implemented optional) checks of TLP formation rules described in this section, or which use undefined Type field values, are Malformed TLPs (MP) and must be discarded without updating Receiver Flow Control information
    - This is a reported error associated with the Receiving Port (see Section 7.2)
- If the value in the Type field is a defined value, update Receiver Flow Control tracking information (see Section 2.9)
- 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)

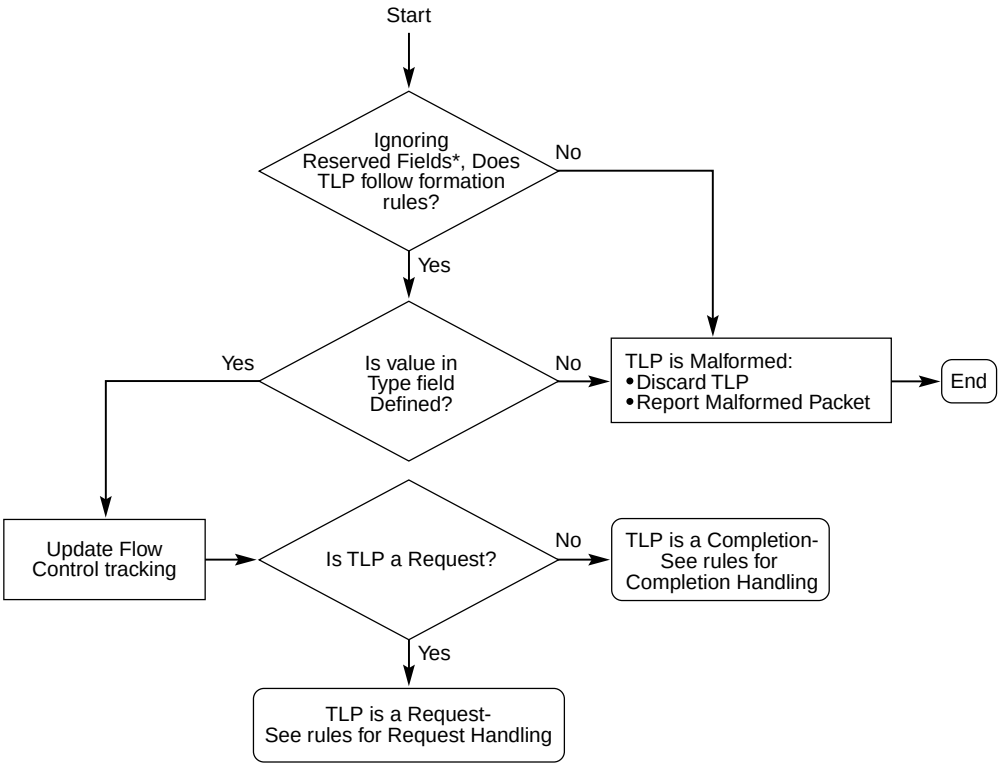

Figure 2-19: Flowchart for Handling of Received TLPs

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 2-20):

- 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 Switch routing rules
- 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 terminates at the Receiver, or if the Message Code field value is defined and corresponds to a Message which must be comprehended by the Switch, the Switch must process the message according to the Message processing rules

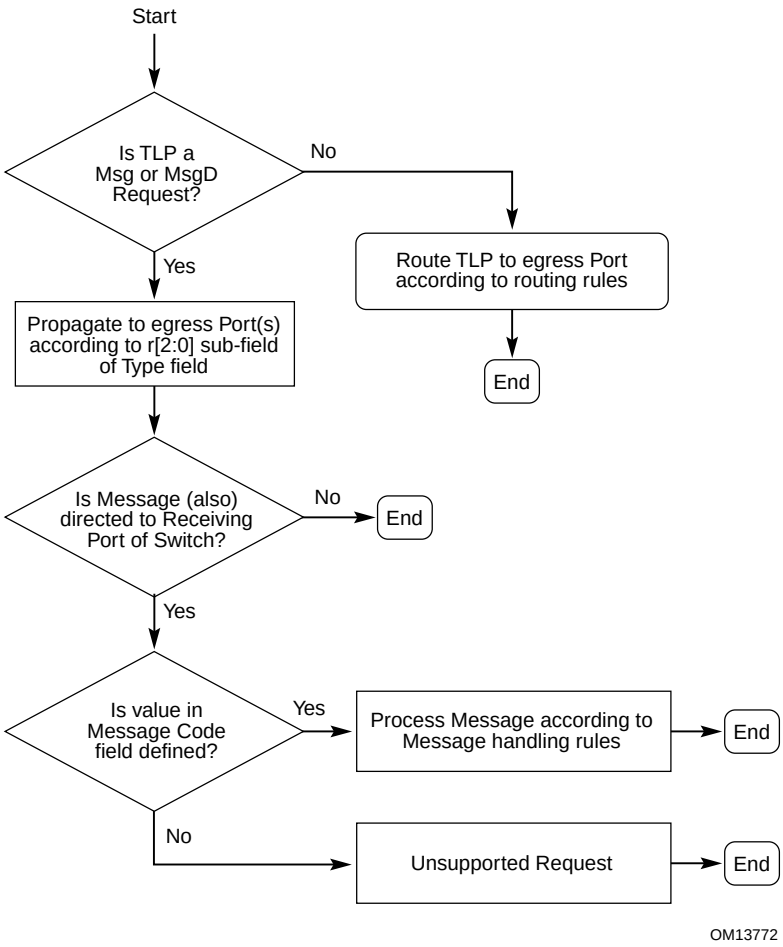

Figure 2-20: Flowchart for Switch Handling of TLPs

2.7.6.2. 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 2-21.

- If the Request Type is not supported by the device, the Request is an Unsupported Request, and is reported according to Section 7.2
    - If the Request requires Completion, a Completion Status of UR is returned (see Section 2.7.5)
- If the Request is a Message, and the Message Code specifies an undefined or unsupported value, the Request is an Unsupported Request, and is reported according to Section 7.2
    - If the Message Code is a supported value, process the Message according to the corresponding Message processing rules

If the Request is not a Message, and is a supported Type, specific implementations may be optimized based on a defined programming model which ensures that certain types of

(otherwise legal) Requests will never occur. Such implementations may take advantage of the following rule:

- If the Request violates the programming model of the device, the device may optionally treat the Request as a Completer Abort, instead of handling the Request normally
    - If the Request is treated as a Completer Abort, this is a reported error associated with the device/function (see Section 7.2)
    - If the Request requires Completion, a Completion Status of CA is returned (see Section 2.7.5)

Implementation Note: Optimizations Based on Restricted Programming Model

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

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

- Otherwise (supported Request Type, not a Message), process the Request
    - If the Completer is permanently unable to process the Request due to a device-specific error condition the Completer must, if possible, handle the Request as a Completer Abort
        - This is a reported error associated with the Receiving device/function, if the error can be isolated to a specific device/function in the component, or to the Receiving Port if the error cannot be isolated (see Section 7.2)
    - For Configuration Requests only, following reset it is possible for a device to indicate that it is temporarily unable to process the Request – in this case, the Configuration Request Retry Status Completion Status is used (see Section 7.6)
    - In the process of servicing the Request, the Completer may determine that the (otherwise acceptable) Request must be handled as an error, in which case the Request is handled according to the type of the error
        - Example: A PCI Express/PCI Bridge may initially accept a Request because it specifies a memory range mapped to the secondary side of the Bridge, but the Request may Master Abort or Target Abort on the PCI side of the Bridge. From the PCI Express perspective, the status of the Request in this case is UR (for Master Abort) or CA (for Target Abort). If the Request requires Completion on PCI Express, the corresponding Completion Status is returned.

- If the Request is a type which requires a Completion to be returned, generate a Completion according to the rules for Completion Formation (see Section 2.7.5)
    - The Completion Status is determined by the result of handling the Request

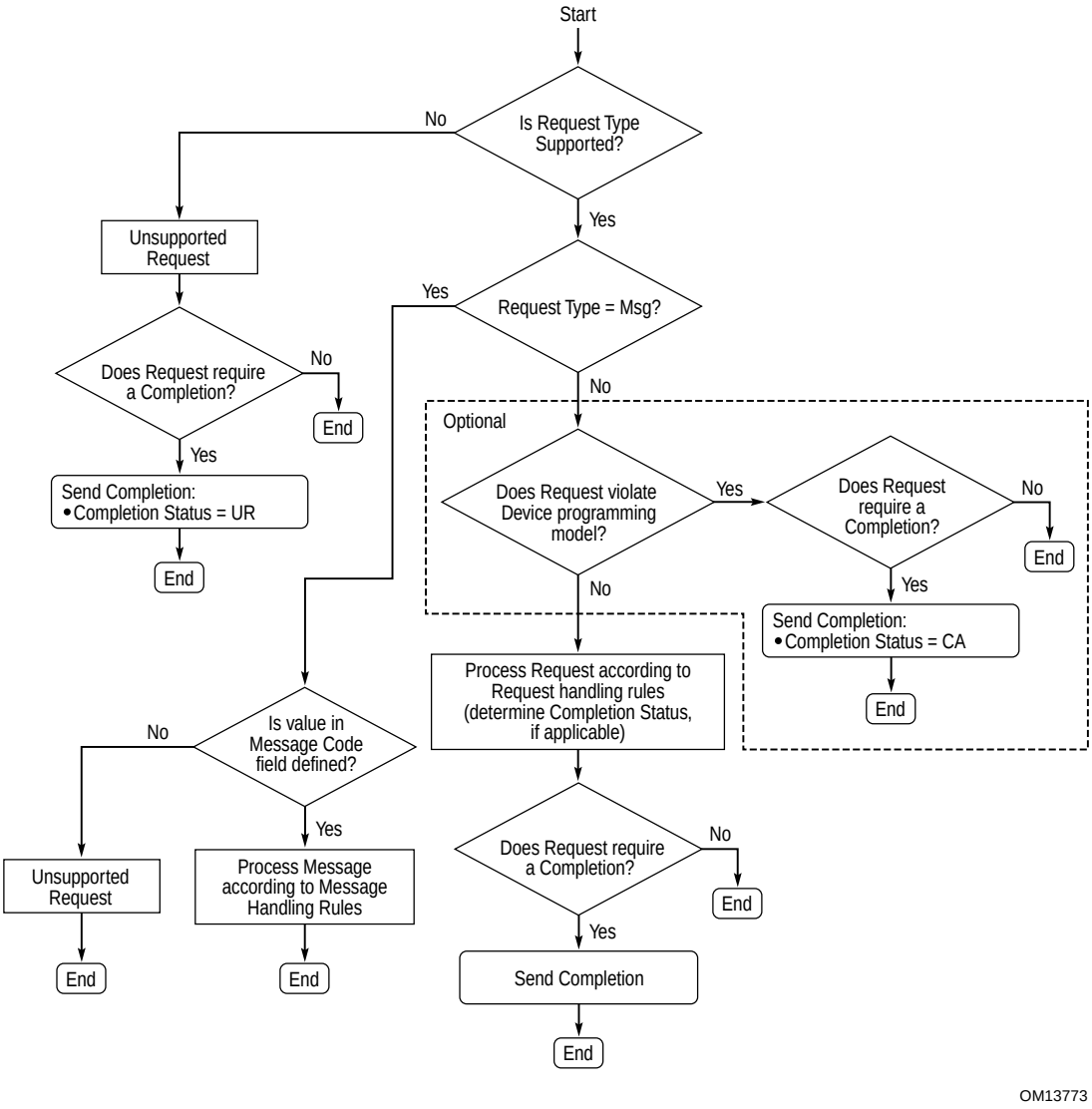

Figure 2-21: Flowchart for Handling of Received Request

Implementation Note: Configuration Retry Status

Some devices require a lengthy self-initialization sequence complete before they are able to service Configuration Requests (common with intelligent I/O solutions on PCI). PCI/PCI-X architecture has specified a 2^{25} (PCI) or 2^{26} (PCI-X) clock "recovery time" following reset to provide the required self-initialization time for such devices. PCI Express "softens" the need for this time based recovery period by implementing a Configuration Request Retry Completion Status. A device in receipt of a Configuration Request may respond with a Configuration Request Retry Completion Status to 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 is only legal to respond with a configuration retry completion status in response to a Configuration Request. Sending this Completion Status in response to any other Request type will result in the generation of an error condition (Malformed TLP – see Section 7.2).

A Root Complex in receipt of a Configuration Request Retry Completion Status in response to its Configuration Request may choose to re-issue the Configuration Request as a new Request on PCI Express or complete the Request to the host as a failed transaction. Root Complex implementations may further choose to only allow a fixed number of Configuration Request/Retry Completion Status loops before determining that something is wrong with the target of the Request and taking appropriate action. When used in systems including PCI Express to PCI/PCI-X bridges, the Root Complex must comprehend the limit T_{rhfa} for PCI/PCI-X agents.

The net result is that existing enumeration and configuration code will not see the lower level retry protocol semantics that are kept at the hardware level. The CPU will only see latency associated with the initial Configuration Request. See Section 7.6 for more information on reset.

2.7.6.2.1. 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 sub-Completion corresponds only to the status associated with the data returned with that sub-Completion
        - A sub-Completion with status other than Successful Completion, or for a Configuration Read only, Configuration Retry Status, terminates the Completions for a single Read Request
            - In this case, the value in the Length field is undefined, and must be ignored by the Receiver

- Completions must not include more data than permitted by the Max_Payload_Size parameter, calculated as a naturally aligned boundary.
    - Receivers must check for violations of this rule – TLPs in violation are Malformed TLPs
        - This is a reported error associated with the Receiving Port (see Section 7.2)

        Note: This is simply a special case of the rules which apply to all TLPs with data payloads
- Read Requests may be completed with one, or in some cases, multiple Completions
- There is a parameter, R, which determines the naturally aligned address boundaries on which a Read Request may be serviced with multiple Completions
    - For a Root Complex, R is 64B or 128B
        - This value is reported through a configuration register (see Section 5.8)

        Note: Bridges and Endpoints may implement a corresponding command bit which may be set by system software to indicate the R value for the Root Complex, allowing the Bridge/Endpoint to optimize its behavior when the Root Complex's R is 128B.
    - For all other system elements, R is 128B
- Completions for Requests which do not cross the naturally aligned address boundaries at integer multiples of R Bytes must include all data specified in the Request
- Requests which do cross the address boundaries at integer multiples of R Bytes may be completed using more than one Completion, but the data must not be fragmented except along the address boundaries.
    - The first Completion must start with the address specified in the Request, and must end at one of the following:
        - the address specified in the Request plus the length specified by the Request (i.e. the entire Request)
        - an address boundary between the start and end of the Request at an integer multiple of R Bytes
    - The final Completion must end with the address specified in the Request plus the length specified by the Request
    - All Completions between, but not including, the first and final Completions must be an integer multiple of R Bytes in length
- Receivers may optionally check for violations of R. 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 7.2)
- Multiple Memory Read Completions for a single Read Request must return data in increasing address order.

- 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.

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 R. However, in all cases, the boundaries specified by R 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 R value of 64 Bytes with one of the following combinations of Completions (Bytes):
192 –or–
128, 64 –or–
64, 128 –or–
64, 64, 64

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

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

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

2.7.6.3. Completion Handling Rules

- When a device receives Completion which does not correspond to any of outstanding Requests issued by that device, the Completion is called an "Unexpected Completion."
- Receipt of an Unexpected Completion is an error and must be handled according to the following rules:
  - The Agent receiving an Unexpected Completion must discard the Completion.
  - An Unexpected Completion is a reported error associated with the Receiving Port (see Section 7.2)

  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 2.12) 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 any Flow Control credits and other resources associated with the Request.
  - Report the error according to the rules in Section 7.2.

- Completions with a Configuration Request Retry Status in response to a Request other than a Configuration Request are Malformed TLPs
  - This is a reported error associated with the Receiving Port (see Section 7.2)

- Completions with a Reserved Completion Status value are treated as if the Completion Status was Unsupported Request (UR)
  - This is a reported error associated with the Receiving device/function, normally the same as the Requestor (see Section 7.2)

- When a Read Completion is received with a Completion Status other than "Successful Completion":
  - No data is included with the Completion
    - The Cpl (or CplLk) encoding is used instead of CplD (CplDLk)
  - 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 32B of Read data for a 128B 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.

2.8. Messages

The PCI Express specification defines the following Messages:

- Baseline Message Group
  - Interrupt Signaling
  - Power Management
  - Error Signaling
  - Locked Transaction Support
  - Slot Power Limit Support
  - Payload Defined
  - Vendor Specific Messages
  - Hot Plug Signaling
- Advanced Switching Support Message Group
  - Data Packet Messages
  - Signal Packet Messages

2.8.1. Baseline Messages

2.8.1.1. Interrupt Signaling - Rules

- MSIs follow the rules defined for PCI.
- MSIs are expressed as Memory Writes, and follow rules for Packet formation, Flow Control, and Data Integrity in the same way as Memory Writes
- MSIs enforce data consistency by pushing ahead of them any previously posted write data using the same TC (as required by the ordering rules in Section 2.5).
- When MSIs are not enabled, interrupts are signaled using the Assert_INTx and Deassert_INTx messages

- Assert_INTx/Deassert_INTx messages are only issued Upstream (towards the Root Complex)
    - 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 7.2)
- The following have no effect, but are not errors:
    - For a particular 'x' (A, B, C or D), receipt of an Assert_INTx message following an earlier Assert_INTx without a Deasssert_INTx message between
    - For a particular 'x' (A, B, C or D), receipt of a Deassert_INTx message following an earlier Deassert_INTx without an Asssert_INTx message between
- All 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 7.2)

Implementation Note: Synchronization of Data Traffic and Interrupts

All Assert_INTx and Deassert_INTx interrupts Requests must use TC0, which ensures that the classic ordering behavior expected in legacy hardware is maintained. MSIs may 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 is 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 2.7.4.2) using each additional TC used for data traffic prior to issuing the MSI. Other methods are also possible. Note, however, that platform software (e.g., a device driver) is generally only capable of issuing transactions using TC0.

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 above rules for INTx messaging apply to all PCI Express compliant components, and facilitate the logical emulation of level-sensitive interrupt lines. A set of four virtual INTx wires is associated with each and every PCI Express Link in a hierarchy, and the components at both ends of each Link must track their logical state. Further rules apply to Switches, Bridges and Root Complexes to enable proper emulation of level-sensitive PCI interrupts.

Rules for Assert_INTx/Deassert_INTx specific to Switches and Bridges:

- Components providing multiple Downstream Links must track the state of the four "virtual wires" embodied in the Assert_INTx/Deassert_INTx pairs independently for each of its Downstream Links, and present a "collapsed" set of Assert_INTx/Deassert_INTx pairs on its Upstream Link following the rules outlined above
    - Collapsing of Downstream virtual wire state onto Upstream virtual wire state must follow the mapping rules provided below.
- In the event that a Downstream Link goes to the DL_Down status (due to surprise removal, hardware failure, software-initiated reset, etc.), the "virtual wires" embodied in the Assert_INTx/Deassert_INTx pairs associated with that Link must be de-asserted. If that results in de-assertion of any Upstream Assert_INTx/Deassert_INTx "virtual wires," then the appropriate Deassert_INTx message(s) must be sent Upstream.

Rules for Assert_INTx/Deassert_INTx specific to the Root Complex:

- The Root Complex must track the state of the four "virtual wires" embodied in the Assert_INTx/Deassert_INTx pairs independently for each of its Downstream Links, and map these virtual signals to system interrupt resources.
    - Details of mapping to system interrupt resources are beyond the scope of this specification
- In the event that a Link attached to the Root Complex goes to the DL_Down status, the "virtual wires" embodied in the Assert_INTx/Deassert_INTx pairs associated with that Link must be de-asserted, and any associated system interrupt resource request must also be discarded.

Within a Switch or below a Bridge, there are typically multiple devices (the virtual PCI bridges for each Downstream Port in the case of a Switch) which must have their associated INTx states mapped to the Upstream Port. The following rules describe how this mapping must be done.

- Switches must collapse the INTx "virtual wires" from each of their Downstream PCI Express Links according to Table 2-12.
    - The mapping is based on the device number (irrespective of the function number) of the PCI to PCI bridge structure representing the Downstream Port of the Switch.

Table 2-12: Switch Mapping for INTx

Dev # of P2P Representing Switch Downstream Port	INTx Message from Downstream PCI Express Link	Mapping to INTx Message on Upstream PCI Express Link
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

- PCI Express-PCI/X Bridges must collapse the INTA-INTD pins from each of their Downstream PCI/X buses into just four INTx "virtual wires" on their Upstream Port.
    - The mapping between the INTx pin on PCI/X bus and the corresponding INTx messages on PCI Express is based on the device number of the PCI/X device requesting the interrupt. The mapping is essentially the same as for switches (shown in Table 2-12) except that the "device numbers" column represents the PCI/X device numbers.
    - Multi-headed PCI Express-PCI/X bridges will collapse the interrupts across the multiple PCI/X buses following the same rules as described for switches. For example, a dual-headed PCI Express-PCI-X bridge would collapse the INTA pins from its two Downstream PCI/X buses as shown in Figure 2-22.

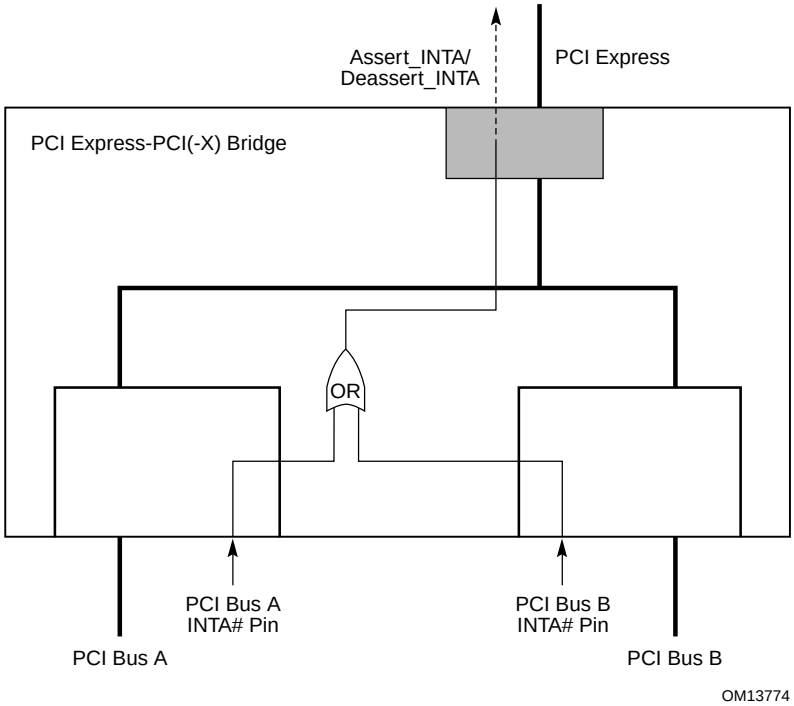

Figure 2-22: INTx Collapsing in a Dual-Headed Bridge

- All internal devices integrated within a Switch or Bridge follow the same mapping rules for interrupt collapsing as described for Switches above.
    - Mapping is based on the device number of the integrated device and is shown in Table 2-12, with the modification that the device number represents the device number of the integrated device.
    - The Bridge/Switch consolidates interrupts from their Downstream Ports/Links and the internal integrated devices following all of the rules stated above and creates just four INTx "virtual wires" for its Upstream Port

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

Implementation Note: System Interrupt Mapping

Note that system software (including BIOS and operating system) needs to comprehend the remapping of legacy interrupts (INTx mechanism) in the entire topology of the system (including hierarchically connected Switches and subordinate PCI Express/PCI Bridges) to establish proper correlation between PCI Express device interrupt and associated interrupt resources in the system interrupt controller. The remapping described by Table 2-12 is applied hierarchically at every Switch. In addition, PCI Express/PCI and PCI/PCI Bridges perform a similar mapping function.

2.8.1.2. Power Management Group

This Message group is used to support PCI Express power management. Table 2-13 summarizes the list of Messages that belong to this group.

Table 2-13: Power Management System Messages

Message	Parameters	Comments
PM_Active_State_Nak	None	Terminate at Receiver
PM_PME	None	Sent Upstream by PME-requesting component. Propagates Upstream.
PME_Turn_Off	None	Broadcast Downstream
PME_TO_Ack	None	Sent Upstream by Endpoint. Sent Upstream by Switch when received on all Downstream Ports.

Notes:

- Address field for all these messages is reserved.
- The Length Field is reserved for all Power Management Messages.
- All power management system messages must use the default Traffic Class designator (TC0).

For more details on the usage of Power Management Messages, refer to Chapter 6.

2.8.1.3. Error Signaling/Logging Group

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

All Error Messages must use the default Traffic Class Designator (TC0).

Table 2-14: Error Messages

Error Message	Description
ERR_COR	This Message is issued when the component or device detects a correctable error on the PCI Express interface. The Root Complex is the ultimate recipient for this Message.
ERR_NONFATAL	This Message is issued when the component or device detects a non-fatal, uncorrectable error on the PCI Express interface. The Root Complex is the ultimate recipient for this Message.
ERR_FATAL	This Message is issued when the component or device detects a fatal, uncorrectable error on the PCI Express interface. The Root Complex is the ultimate recipient for this Message.

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 7.2 for details on uses for these messages.

2.8.1.4. Messages for Support of Locked Transactions

The PCI Express specification defines the Unlock Message to support Lock Transaction sequences. The following rules apply to Unlock Message:

- The Unlock Message must use the default Traffic Class designator (TC0)

See Section 7.5 for details on implementing support for Lock Transaction sequences.

2.8.1.5. Slot Power Limit Support

The Set_Slot_Power_Limit message includes a one DW data payload. 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 component (Endpoint, Switch or a PCI Express-PCI Bridge) attached to the same Link. The data payload is copied from the Slot Capabilities Register of the Downstream Port and is written into the Device Capabilities Register of the Upstream Port on the other side of the Link. Bits 9:8 of the data payload map to the Slot Power Limit Scale field and Bits 7:0 map to the Slot Power Limit Value field. This message is 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 5.8.9) when the Data Link Layer reports DL_Up status.
- Anytime when Link transitions from a non-DL_Up status to a DL_Up status (see Section 2.14).

The component on the other side of the Link (Endpoint, Switch or PCI Express-PCI Bridge) 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. motherboard) as well as components that are targeted for integration on a card/module where power consumption of the entire card/module is below the lowest

power limit specified for the card/module form-factor (as defined in the corresponding electromechanical specification) are permitted to hardwire the value "0" in the Slot Power Limit Scale and Slot Power Limit Value fields of the Device Capabilities Register, and are not required to copy the Set_Slot_Power limit payload into that register.

For more details on Power Limit control mechanism see Section 7.9.

2.8.1.6. Payload_Defined Message

The Payload_Defined Message allows expansion of PCI Express messaging capabilities, either as a general extension to the PCI Express specification or a vendor-specific extension. Such extensions are not covered specifically in this document. This section defines the rules associated with this Message generically.

- The Payload_Defined Message includes at least 1 DW of data (see Figure 2-23)
    - The DW of data immediately following the Header includes two 16 bit fields:
        - Vendor ID (same value as used in Configuration Space Header – see Section 5.5)
            - the value 0000h is reserved for non-vendor-specific extensions
        - Message Sub-Type
    - There may be additional data following this 1 DW
    - The value in the Length field must correspond to the size of the entire data payload associated with the TLP, including the required DW and any additional data payload
- Receivers silently discard Payload_Defined Messages which they are not designed to receive – this is not an error condition.

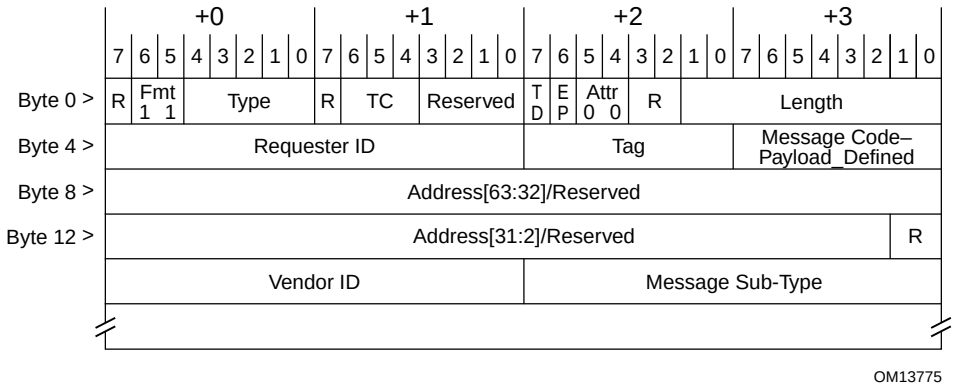

Figure 2-23: Payload_Defined Message

2.8.1.7. Vendor Specific Message

Vendor specific Messages use the Code values 128 to 255.

- Receivers silently discard Payload_Defined Messages which they are not designed to receive – this is not an error condition.

2.8.1.8. Hot Plug Signaling Messages

The Hot plug Signaling Messages are virtual signals between Switches/Root Ports that support Hot plug Event signaling and devices on cards that support Removal Request functionality (doorbell mechanism) on the card. The Messages are defined to replicate the events and registers defined for doorbell mechanisms wired directly to the Switch/Root Port. For more information see Section 7.7.

Note that only devices on cards that support Remove request functionality (doorbell mechanism) on the card and the switch ports/root ports that support such cards are required to implement the hot plug signaling messages.

Table 2-15: Hot Plug Signaling Messages

Message	Description
Attention_Indicator_On	This message is issued by the Switch/Root Port when the Attention Indicator Control is set to 01b. The end device receiving the message will terminate the message and initiate appropriate action for to cause the Attention Indicator located on the card to turn on. If no indicators are present on the card, the message is discarded. For more implementation information see Section 7.7.
Attention_Indicator_Blink	This message is issued by the Switch/Root Port when the Attention Indicator Control is set to 10b. The end device receiving the message will terminate the message and initiate appropriate action for to cause the Attention Indicator located on the card to blink. If no indicators are present on the card, the message is discarded. For more implementation information see Section 7.7.
Attention_Indicator_Off	This message is issued by the Switch/Root Port when the Attention Indicator Control is set to 11b. The end device receiving the message will terminate the message and initiate appropriate action for to cause the Attention Indicator located on the card to turn off. If no indicators are present on the card, the message is discarded. For more implementation information see Section 7.7.
Power_Indicator_On	This message is issued by the Switch/Root Port when the Power Indicator Command is set to 01b. The end device receiving the message will terminate the message and initiate appropriate action for to cause the Power Indicator located on the card to turn on. If no indicators are present on the card, the message is discarded. For more implementation information see Section 7.7.
Power_Indicator_Blink	This message is issued by the Switch/Root Port when the Power Indicator Control is set to 10b. The end device receiving the message will terminate the message and initiate appropriate action for to cause the Power Indicator located on the card to blink. If no indicators are present on the card, the message is discarded. For more implementation information see Section 7.7.
Power_Indicator_Off	This message is issued by the Switch/Root Port when the Power Indicator Command is set to 11b. The end device receiving the message will terminate the message and initiate appropriate action for to cause the Power Indicator located on the card to turn off. If no indicators are present on the card, the message is discarded. For more implementation information see Section 7.7.

Message	Description
Attention_Button_Pressed	This message is issued by a device in a slot that implements an Attention Button on the card to signal the Switch/Root Port to generate the Attention Button Pressed Event. The Switch Switch/Root Port terminates the message and sets the Attention Button Pressed register to 1b which may result in an interrupt being generated. For more implementation information see Section 7.7.

All Endpoint devices must be able to handle the Attention and Power Indicator messages even if the device does not implement the indicators. All down stream ports of switches and root ports must be able to handle the Attention_Button_Pressed message.

2.8.2. Advanced Switching Support Message Group

The Messages that belong to this group can be divided into the following two types:

- Data Packet Messages:
  - Unicast, Data Packet
  - Multicast, Data Packet
- Signaling Packet Messages:
  - Signaling Packet, without interrupt
  - Null signaling Packet, interrupt to Host in the destination Hierarchy
  - Null signaling Packet, interrupt to destination device
  - Signaling Packet, with interrupt to Host in the destination Hierarchy
  - Signaling Packet, with interrupt to destination device

A detailed description of message types and message headers will be presented in a separate document entitled *Advanced PCI Express Packet Switching Specification*. This is a companion specification to the PCI Express Base Specification.

2.9. Ordering and Receive Buffer Flow Control

2.9.1. Overview and Definitions

Flow Control (FC) is used to prevent overflow of receiver buffers and to enable compliance with the ordering rules defined in Section 2.5. 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 2-24. 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.

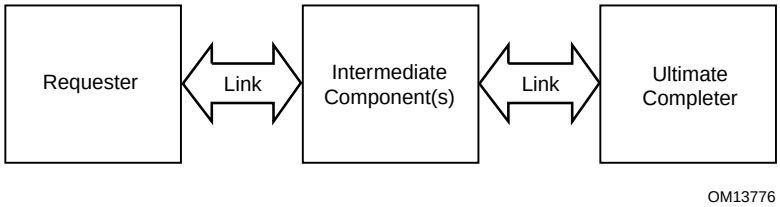

Figure 2-24: Relationship between Requester and Ultimate Completer

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

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

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

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

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

2.9.2. Flow Control Rules

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

- Flow Control information is transferred using Flow Control Packets (FCPs), which are a type of DLLP (see Section 3.4)
- The unit of Flow Control credit is 16 Bytes for Data
- For headers, the unit of Flow Control credit is one header
- Each Virtual Channel has independent Flow Control
- Flow Control distinguishes three types of TLPs (note relationship to ordering rules – see Section 2.5):
    - Posted Requests (P) – Messages and Memory Writes
    - Non-Posted Requests (NP) – All Reads, I/O, and Configuration Writes
    - Completions (CPL) – Associated with corresponding NP Requests
- In addition, Flow Control distinguishes the following types of TLP information within each of the three types:
    - Headers (H)
    - Data (D)
- Thus, there are six types of information tracked by Flow Control for each Virtual Channel, as shown in Table 2-16.

Table 2-16: 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 2-17.

Table 2-17: TLP Flow Control Credit Consumption

TLP	Credit Consumed[8]
Memory, I/O, Configuration Read Request	1 NPH unit
Memory Write Request	1 PH + n PD units[9]
I/O, Configuration Write Request	1 NPH + 1 NPD Note: size of data written is never more than one (aligned) DW
Message Requests without data	1 PH unit
Message Requests with data	1 PH + n PD units
Memory Read Completion	1 CPLH + n CPLD units
I/O, Configuration Read Completions	1 CPLH unit + 1 CPLD unit
I/O, Configuration Write Completions	1 CPLH unit

- Components must implement independent Flow Control for all Virtual Channels that are supported by that component.
- Flow Control is initialized autonomously by hardware only for the default Virtual Channel (VC0)
    - VC0 is initialized when the Data Link Layer is in the DL_Init state following reset (see Sections 3.2 and 3.3)
- When other Virtual Channels are enabled by software, each newly enabled VC will follow the Flow Control initialization protocol (see Section 3.3)
    - Software enables a Virtual Channel by setting the VC Enable bits for that Virtual Channel in both components on a Link (see Section 5.11)
        - For a multi-function device, a given VC is enabled when the VC Enable bit is set for any function; enabling the same VC in additional functions enables those functions to use the VC, but the VC is initialized only once

    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
    - For a multi-function device, the VC Enable bit must be clear for all functions
    - Disabling a Virtual Channel for a component resets the Flow Control tracking mechanisms for that Virtual Channel in that component

---

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

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

- InitFC1 and InitFC2 FCPs are used only for Flow Control initialization (see Section 3.3)
- An InitFC1, InitFC2, or UpdateFC FCP which specifies a Virtual Channel which is not enabled 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 2-18.
    - 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 7.2)

Table 2-18: Minimum Flow Control Advertisements

Credit Type	Minimum Advertisement
PH	1 unit – credit value of 01h
PD	Largest possible setting of the Max_Payload_Size for the component divided by FC Unit Size. Example: If the largest Max_Payload_Size value supported is 1024B, the smallest permitted initial credit value would be 040h.
NPH	1 unit – credit value of 01h
NPD	1 unit – credit value of 01h
CPLH	Switch and PCI Express to PCI-X Bridge (PCI-X mode only): 1 FC unit – credit value of 01h Root Complex, Endpoint, and PCI Express to PCI Bridge: "infinite" FC units – initial credit value of all '0's[10]
CPLD	Switch and PCI Express to PCI-X Bridge (PCI-X mode only): Largest possible setting of the Max_Payload_Size for the component divided by FC Unit Size, or the size of the largest Read Request the component will ever generate, whichever is smaller. Root Complex, Endpoint, and PCI Express to PCI Bridge: "infinite" FC units – initial credit value of all '0's

- If an "infinite" credit advertisement has been made for CPL during initialization, no Flow Control updates (UpdateFC) are sent for CPL following initialization
    - 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)

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

        - If checked, this is a reported error associated with the Receiving Port (see Section 7.2)

- A TLP using an uninitialized VC is a Malformed TLP
    - This is a reported error associated with the Receiving Port (see Section 7.2)
- 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.
        - Set to all '0's at Interface Initialization
        - Incremented for each TLP the Transaction Layer allows to pass the Flow Control gate for Transmission
            - Size of increment corresponds to the number of credits consumed by the information committed to be sent
            - Incremented 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 sent, and [Field Size] is 8 for PH, NPH, and CPLH and 12 for PD, NPD and CPLD

    - CREDIT_LIMIT
        - The limit for total number of FC units which have been advertised by the Receiver since Flow Control initialization
        - Undefined at Interface Initialization
        - Set to the value indicated during Flow Control initialization

103

- For each FC update received,
  - if CREDIT_LIMIT is not equal to the update value, set CREDIT_LIMIT to update value
  - Optionally, check the update value validity by evaluating the equation:

    (update value - CREDIT_LIMIT) mod $2^{[\text{Field Size}]}$

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

    If the equation evaluates as true, the violation is a Flow Control Protocol Error

    - If checked, this is a reported error associated with the Receiving Port (see Section 7.2)

Note: In accordance with this rule, the largest permitted initial value, change in value, or advertised total for any PH, NPH or CPLH credit value is 128. The largest permitted initial value, change in value, or advertised total for any PD, NPD or CPLD credit value is 2048.

- 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 2.5, 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:

    CREDITS_REQUIRED =

    (CREDITS_CONSUMED + <credit units required>)) 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:

    (CREDIT_LIMIT - 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 2.5 when determining what types of TLPs must be permitted to bypass the stalled TLP.
- The return of FC credits for a Transaction must not be interpreted to mean that the Transaction has completed or achieved system visibility.
    - Flow Control credit return is used for receive buffer management only, and Agents must not make any judgment about the Completion status or system visibility of a Transaction based on the return or lack of return of Flow Control information.
- When a Transmitter sends a nullified TLP (with inverted LCRC and using EDB as the end Symbol), the Transmitter does not modify CREDITS_CONSUMED for that TLP (see Section 3.5.2.1)
- 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
        - Initially set according to the buffer size and allocation policies of the Receiver
        - This value is included in the InitFC and UpdateFC DLLPs (see Section 3.4)
        - Incremented as the Receiver Transaction Layer makes additional receive buffer space available by processing Received TLPs
            - Increment size corresponds to the size of the space made available
            - Incremented as shown:

              CREDITS_ALLOCATED :=

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

              Where [Field Size] is 8 for PH, NPH and CPLH and 12 for PD, NPD, and CPLD

105

- 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
  - Set to all '0's at Interface Initialization
  - Incremented for each Received TLP according to the number of FC units of the given type consumed by the Received TLP, provided that TLP:
    - passes the Data Link Layer integrity checks
    - is not malformed
    - does not consume more credits than have been allocated (see following rule)

- If a Receiver implements the CREDITS_RECEIVED counter, then when a nullified TLP (with inverted LCRC and using EDB as the end Symbol) is received, the Receiver does not modify CREDITS_RECEIVED for that TLP (see Section 3.5.2.1)
- A Receiver may optionally check for Receiver Overflow errors (TLPs exceeding CREDITS_ALLOCATED), by checking the following equation:

  $$(\text{CREDITS_ALLOCATED} - \text{CREDITS_RECEIVED}) \bmod 2^{[\text{Field Size}]} >= 2^{[\text{Field Size}]}/2$$

  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 7.2)

  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 NPH, NPD, PH and, if non-infinite, CPLH types, an UpdateFC FCP must be scheduled for Transmission each time the following sequence of events occurs:
  - all advertised FC units for a particular type of credit are consumed by TLPs received
  - one or more units of that type are made available by TLPs processed
- For PD and, if non-infinite, CPLD types, when the number of available credits is less than Max_Payload_Size, an UpdateFC FCP must be scheduled for Transmission each time one or more units of that type are made available by TLPs processed
- UpdateFC FCPs may be scheduled for Transmission more frequently than is required
- When the Link is Active, Update FCPs for each enabled type of FC credit must be scheduled for transmission at least once every 10 μs

107

Implementation Note: Flow Control Update Frequency

For components subject to receiving streams of TLPs, it is desirable to implement receive buffers larger than the minimum size required to prevent Transmitter throttling due to lack of available credits. Likewise, UpdateFC FCPs must be returned such that the time required to send, receive and process the UpdateFC is sufficient. Table 2-19 shows recommended values for the frequency of transmission based on Link Width and Max_Payload_Size values.

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

$$\frac{(Max_Payload_Size + TLPOverhead) * UpdateFactor}{LinkWidth} + InternalDelay$$

where

Max_Payload_Size	The value in the Max_Payload_Size field of the Link Command Register
TLP Overhead	Represents the additional TLP components which consume Link bandwidth (Header, LCRC, framing Symbols) and is treated here as a constant value of 24 Symbols
UpdateFactor	Used to balance Link bandwidth efficiency and receive buffer sizes – the value varies according to Max_Payload_Size and Link width, and is included in Table 2-19
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 11 Symbol Times

Table 2-19: UpdateFC Transmission Latency Guidelines by Link Width and Max Payload (Symbol Times)

		Link Operating Width						
		x1	**x2**	**x4**	**x8**	**x12**	**x16**	**x32**
Max_Payload_Size	**128B**	223 UF = 1.4	117 UF = 1.4	64 UF = 1.4	58 UF = 2.5	49 UF = 3.0	39 UF = 3.0	25 UF = 3.0
	256B	403 UF = 1.4	207 UF = 1.4	109 UF = 1.4	98 UF = 2.5	81 UF = 3.0	63 UF = 3.0	37 UF = 3.0
	512B	547 UF = 1.0	279 UF = 1.0	145 UF = 1.0	78 UF = 1.0	100 UF = 2.0	78 UF = 2.0	44 UF = 2.0
	1024B	1059 UF = 1.0	535 UF = 1.0	273 UF = 1.0	142 UF = 1.0	185 UF = 2.0	142 UF = 2.0	76 UF = 2.0
	2048B	2083 UF = 1.0	1047 UF = 1.0	529 UF = 1.0	270 UF = 1.0	356 UF = 2.0	270 UF = 2.0	140 UF = 2.0
	4096B	4131 UF = 1.0	2071 UF = 1.0	1041 UF = 1.0	526 UF = 1.0	697 UF = 2.0	526 UF = 2.0	268 UF = 2.0

2.10. Data Integrity

2.10.1. Introduction

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

2.10.2. 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 5.8.3).

- If a device is enabled to generate ECRC, it must calculate and apply ECRC for all TLPs originated by the device
- Switches must pass TLPs with ECRC unchanged from the Ingress Port to the Egress Port
- If a device reports the capability to check ECRC, it must support Advanced Error Reporting (see Section 7.2)
- If a device is enabled to check ECRC, it must do so for all TLPs received by the device including ECRC
    - Note that it is still possible for the device to receive TLPs without ECRC, and these are processed normally – this is not an error

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

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

- The ECRC value is calculated using the following algorithm (see Figure 2-25).
- The polynomial used has coefficients expressed as 04C1 1DB7h
- The seed value (initial value for ECRC storage registers) is FFFF FFFFh
- All invariant fields of the TLP header and the entire data payload (if present) are included in the ECRC calculation, all bits in variant fields must be set to '1' for ECRC calculations.
    - Bit 0 of the Type field is variant
    - The EP field is variant
    - all other fields are invariant
- ECRC calculation starts with bit 0 of Byte 0 and proceeds from bit 0 to bit 7 of each Byte of the TLP
- The result of the ECRC calculation is complemented, and the complemented result bits are mapped into the 32b TLP Digest field as shown in Table 2-20.

Table 2-20: Mapping of Bits into ECRC Field

ECRC Result Bit	Corresponding Bit Position in the 32b 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

ECRC Result Bit	Corresponding Bit Position in the 32b TLP Digest 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 32b ECRC value is placed in the TLP Digest field at the end of the TLP (see Figure 2-2)
- 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 32b TLP Digest field of the received TLP
    - comparing the calculated result with the value in the TLP Digest field of the received TLP

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

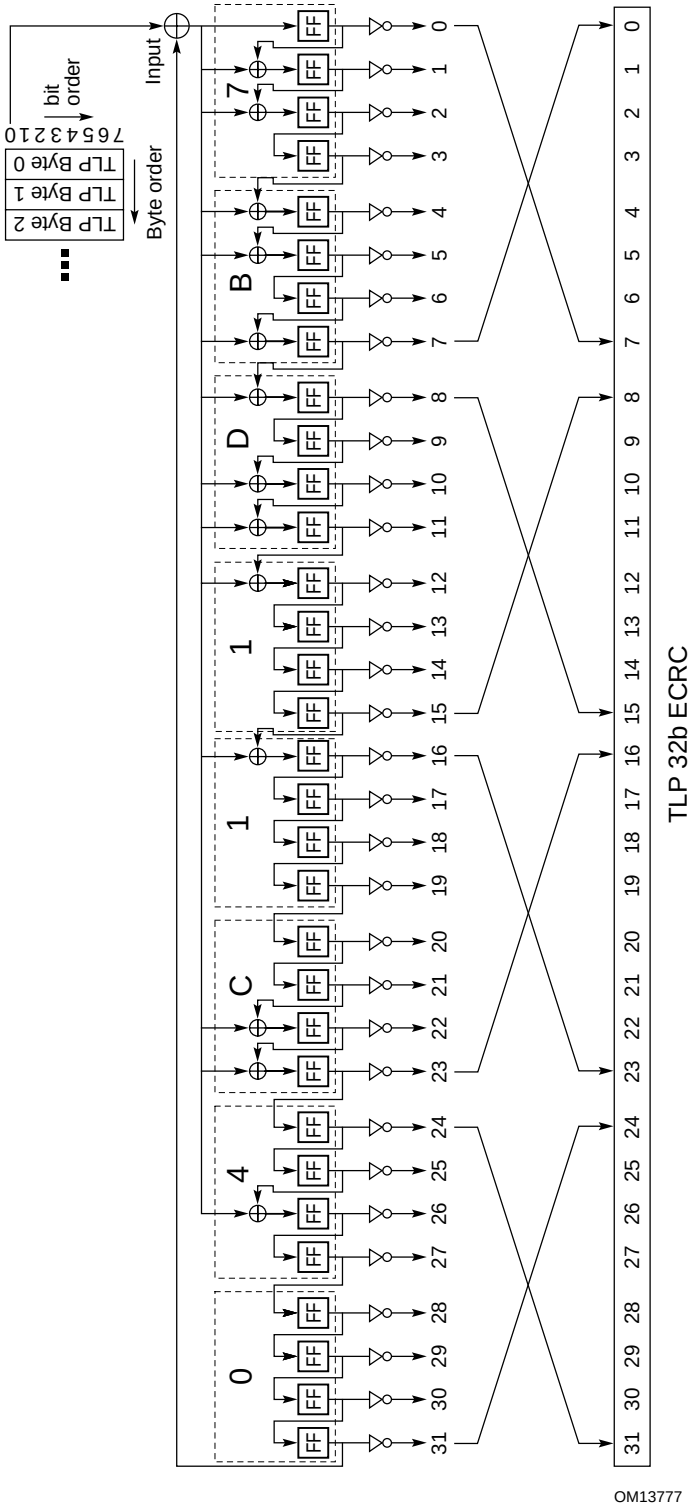

Figure 2-25: Calculation of 32b ECRC for TLP End to End Data Integrity Protection

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 a tandem 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.

2.11. Error Forwarding

Error Forwarding (also known as data poisoning), is enabled in PCI Express by either modifying the value placed in the TLP Digest field or by setting the proper value in the TD and EP fields. The rules for doing this are specified below 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.

2.11.1. Error Forwarding Usage Model

- Error Forwarding is only used for Read Completion Data or Write Data, never for the cases when the error is in the "header" (request phase, address/command, etc.). Requests/Completions with header errors cannot be forwarded in general since true destination cannot be positively known and, therefore, forwarding may cause direct or side effects such as data corruption, system failures, etc.
- Used for controlled propagation of error through the system, system diagnostics, etc.
- Does not cause Link Layer Retry – Poisoned TLPs will be retried only if there are transmission errors on PCI Express as determined by the TLP error detection mechanisms in the Data Link Layer. The Poisoned TLP may ultimately cause the originator of the request to re-issue it (at the Transaction Layer of above) in the case of read operation or to take some other action. Such use of Error Forwarding information is beyond the scope of this specification.

2.11.2. Rules For Use of Data Poisoning

- Support for TLP poisoning in a Transmitter is optional.
- Data Poisoning applies only to the data within a Write Request (Posted or Non-Posted) or a Read Completion.
- Poisoning of a TLP is indicated using one of the following two mechanisms:
  - TD field = '1': The value used for the TLP Digest field is the "stomp code" of all '1's
  - TD field = '0' and EP field = '1'
- If a Transmitter supports data poisoning , TLPs that are known to the Transmitter to include bad data must use one of the two poisoning mechanism defined above. The Receiver must consider all the information within a poisoned TLP to be affected
  - If applying Error Forwarding, the Receiver must cause all data from the indicated TLP to be tagged as bad ("poisoned").
- Receipt of a poisoned TLP is a reported error associated with the Receiving device/function (see Section 7.2)

Note: For some applications it may be desirable for the Receiver to use data marked corrupt – such use is not forbidden. How the Receiver makes use of the information that a TLP is poisoned is beyond the scope of this document.

2.12. 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.

The PCI Express elements that are capable of initiating Requests that invoke Completions must implement Completion Timeout mechanism. An exception is made for Configuration Requests (see below). This mechanism is activated for each Request which requires Completion when the Request is transmitted. Since PCI Express Switches do not autonomously initiate Requests (that need Completion), the requirement for Completion Timeout support is limited only to Root Complex, PCI Express-PCI Bridges, and Endpoint devices.

The PCI Express Specification defines the following range for the min/max acceptable timer values for the Completion Timeout mechanism:

- The Completion Timeout timer must not expire (i.e., cause timeout event) in less than 10 ms.
- The Completion Timeout timer must expire if a Request is not completed in 50 ms.

A Completion Timeout is a reported error associated with the Requestor device/function (see Section 7.2).

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

Configuration Requests have special requirements (see Sections 2.7.6.2 and 7.6). Because of these special requirements, the support and timer values for a Completion Timeout for Configuration Requests are implementation specific.

2.13. Transaction Layer Behavior in DL_Down Status

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

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

- returning all internal logic to the state specified for Link initialization
- forming completions for any Requests submitted by the device core for Transmission, returning "Unsupported Request" Completion Status, then discarding the Requests
    - This is a reported error associated with the device/function for the (virtual) Bridge associated with the Port (see Section 7.2)
    - Requests already being processed by the Transaction Layer, for which it may not be practical to return Completions, are discarded

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

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

  Note: The point at which a Request becomes "uncompletable" is implementation specific
- discarding all Completions submitted by the device core for Transmission

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

- returning all internal logic to the state specified for Link initialization
- discarding all TLPs being processed
- (for Switch and Bridge) propagating Link Reset to all other Ports

2.14. Transaction Layer Behavior in DL_Up Status

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

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

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

3

3. Data Link Layer Specification

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

3.1. Data Link Layer Overview

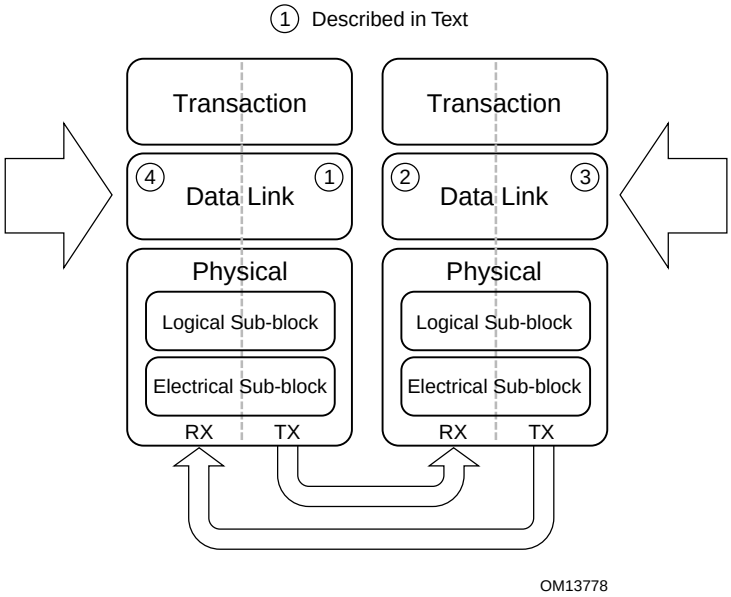

Figure 3-1: Layering Diagram Highlighting the Data Link Layer

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

Data Exchange:

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

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)
- Acknowledgement and Retry DLLPs
- Error indications for error reporting and logging mechanisms
- Link Acknowledgement Timeout replay mechanism

Initialization and power management services:

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

Data Link Layer Packets (DLLPs) are:

- used for Link Management functions including TLP acknowledgement, power management, and conveyance 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 16b CRC and TLPs (which can be much longer) use a 32b 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 supercede any information lost.

TLPs which fail the data integrity checks (LCRC and sequence number), or which are lost in transmission from one component to another, are re-sent by the transmitter. The transmitter 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. Because of these delays, the Transmit Data Link Layer (1 and 3) can apply backpressure to the Transmit Transaction Layer, and the Receive Data Link Layer (2 and 4)

communicates the presence or absence of valid information to the Receive Transaction Layer.

3.2. Data Link Control and Management State Machine

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

States:

- DL_Down – Physical Layer reporting Link is non-operational or Port is not connected
- DL_Init – Physical Layer reporting Link is operational, initialize Flow Control for the default Virtual Channel
- DL_Active – Normal operation mode

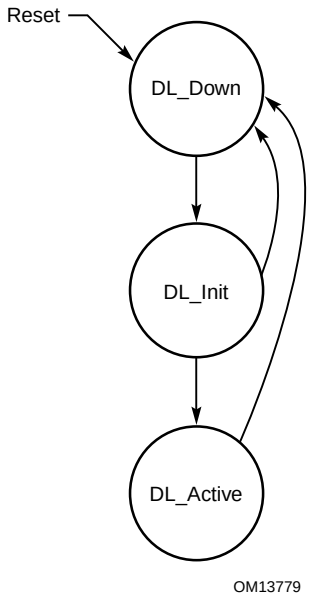

Figure 3-2: Data Link Control and Management State Machine

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 7.6)
  - Upon entry to DL_Inactive
    - Reset all Data Link Layer state information to default values
    - Discard the contents of the Data Link Layer Retry Buffer (refer Section 3.5)
  - While in DL_Inactive:
    - Report DL_Down status to the Transaction Layer as well as to the rest of the Data Link Layer

      Note: This will cause the Transaction Layer to discard any outstanding transactions and to terminate internally any attempts to transmit a TLP. For a Port on a Root Complex or at the "bottom" of a Switch, this is equivalent to a "hot remove." For Port on an Endpoint or at the "top" of a Switch, having the Link go down is equivalent to a hot reset (see Section 2.13).
    - Discard TLP information from the Transaction and Physical Layers
    - Do not generate or accept DLLPs
  - Exit to DL_Init if:
    - Indication from the Transaction Layer that the Link is not disabled by software and the Physical Layer reports Physical LinkUp = 1
- DL_Init
  - While in DL_Init:
    - Initialize Flow Control for the default Virtual Channel, VC0, following the Flow Control initialization protocol described in Section 3.3
    - Report DL_Down status while in state FC_INIT1; DL_Up status in state FC_INIT2
  - Exit to DL_Active if:
    - Flow Control initialization completes successfully, and the Physical Layer continues to report Physical LinkUp = 1
  - Terminate attempt to initialize Flow Control for VC0 and Exit to DL_Inactive if:
    - Physical Layer reports Physical LinkUp = 0

- DL_Active
  - DL_Active is referred to as the normal operating state
  - While in DL_Active:
    - Accept and transfer TLP information with the Transaction and Physical Layers as specified in this chapter
    - Generate and accept DLLPs as specified in this chapter
    - Report DL_Up status to the Transaction and Data Link Layers
  - Exit to DL_Inactive if:
    - Physical Layer reports Physical LinkUp = 0

3.3. Flow Control Initialization Protocol

Before starting normal operation following power-up or interconnect Reset, it is necessary to initialize Flow Control for the default Virtual Channel, VC0 (see Section 7.6). 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). This section describes the initialization process which is used for all VCs. Note that since VC0 is enabled before all other VCs, no TLP traffic of any kind will be active prior to initialization of VC0. However, when additional VCs are being initialized there will typically be TLP traffic flowing on other, already enabled, VCs. Such traffic has no direct effect on the initialization process for the additional VC(s).

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

- FC_INIT1
- FC_INIT2

The rules for this process are given in the following section. Figure 3-3 shows a flowchart of the process.

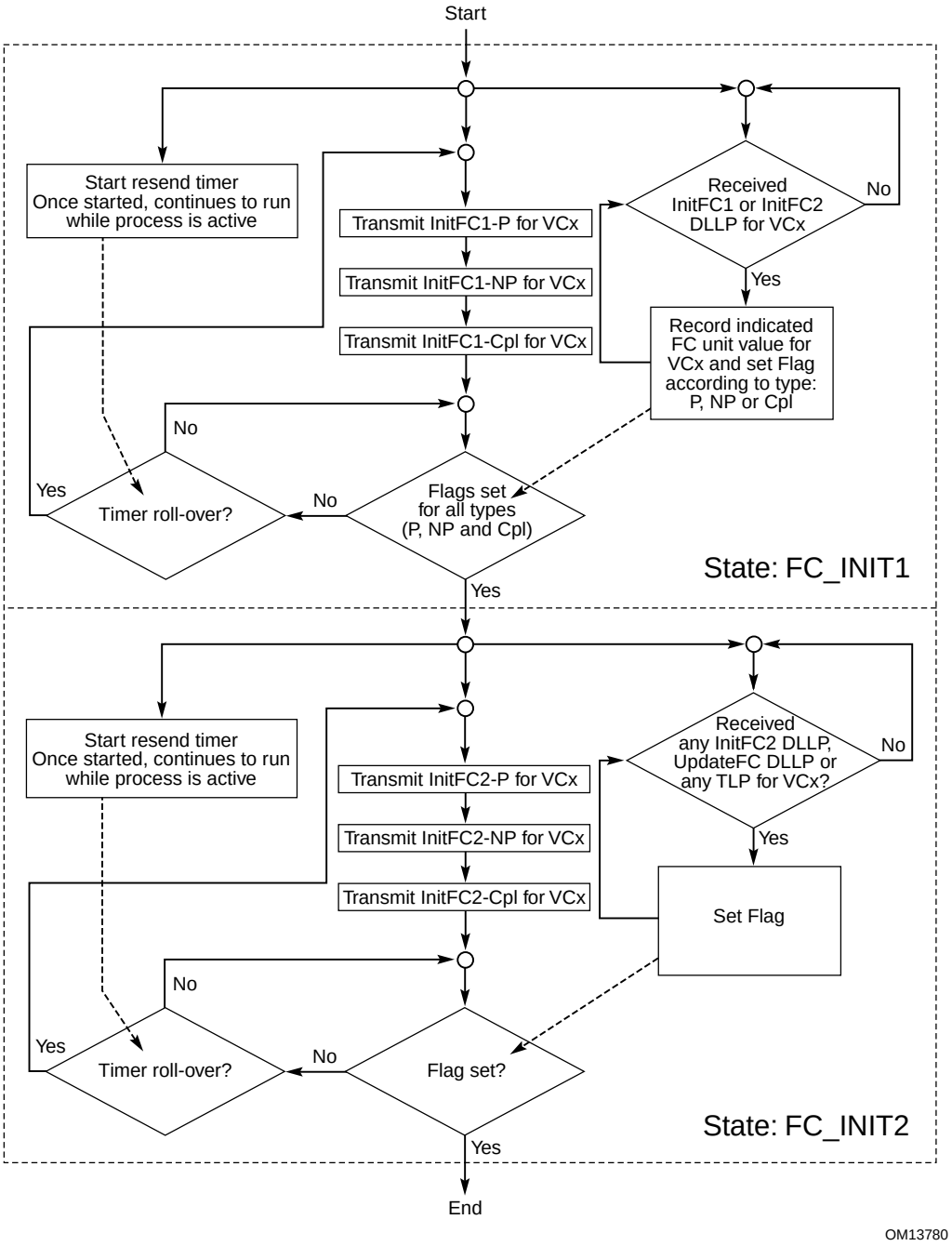

Figure 3-3: Flowchart Diagram of Flow Control Initialization Protocol

3.3.1. Flow Control Initialization State Machine Rules

- Rules for state FC_INIT1:
  - Entered when initialization of a VC (VCx) is required
    - Entrance to DL_Init state
    - When a VC is enabled by software (see Section 5.11)
  - While in FC_INIT1:
    - Transaction Layer must block transmission of TLPs using VCx
    - Transmit the following uninterrupted sequence of three successive InitFC1 DLLPs for VCx in the following pattern:
      - InitFC1 – P (first)
      - InitFC1 – NP (second)
      - InitFC1 – Cpl (third)
    - Repeat this InitFC1 DLLP transmission sequence as follows:
      - For VC0, transmit continuously at the maximum rate possible on the Link (resend timer value is 0)
      - For VCs other than VC0, repeat the sequence when no other TLPs or DLLPs are available for Transmission, but no less frequently than at an interval of 8 µs (-0% / +100%), measured from the start of transmission of the preceding sequence
    - Process received InitFC1 and InitFC2 DLLPs:
      - Record the indicated FC unit values
      - Set Flag FI1 once FC unit values have been recorded for each of P, NP and Cpl
  - 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 or VCx

- Rules for state FC_INIT2:
  - While in FC_INIT2:
    - Transmission of TLPs using VCx by the Transaction Layer is permitted
    - Transmit the following uninterrupted sequence of three successive InitFC2 DLLPs for VCx in the following pattern:
      - InitFC2 – P (first)
      - InitFC2 – NP (second)
      - InitFC2 – Cpl (third)
    - Repeat this InitFC2 DLLP transmission sequence as follows:
      - For VC0, transmit continuously at the maximum rate possible on the Link (resend timer value is 0)
      - For VCs other than VC0, repeat the sequence when no other TLPs or DLLPs are available for Transmission, but no less frequently than at an interval of 8 µs (-0% / +100%), measured from the start of transmission of the preceding sequence
    - Process received InitFC1 and InitFC2 DLLPs:
      - Ignore the indicated FC unit values
      - Set flag FI2 on receipt of any InitFC2 DLLP or VCx
    - Set flag FI2 on receipt of any TLP on VCx, or any UpdateFC DLLP for VCx
  - Signal completion and exit if:
    - Flag FI2 has been set
- Violations of Flow Control initialization protocol are Data Link Layer Protocol Errors (DLLPE). Checking for such errors in FC initialization protocol is optional. If checking is implemented, any detected error is a reported error associated with the Port (see Section 7.2)

3.4. Data Link Layer Packets (DLLPs)

The following DLLPs are used to support Link data integrity mechanisms:

- 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
- Plus additional DLLPs used for Power Management

3.4.1. Data Link Layer Packet Rules

All DLLPs include the following fields:

- DLLP Type - Specifies the type of DLLP. The defined encodings are shown in Table 3-1.
- 16b CRC

See Figure 3-4.

Table 3-1: DLLP Type Encodings

Encodings	DLLP Type
0000 0000	Ack
0001 0000	Nak
0010 0000	PM_Enter_L1
0010 0001	PM_Enter_L23
0010 0010	PM_Active_State_Request_L0s
0010 0011	PM_Active_State_Request_L1
0010 0100	PM_Request_Ack
0011 0000	Vendor Specific – Not used in normal operation
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
1100 $0v_2v_1v_0$	InitFC2-P
1101 $0v_2v_1v_0$	InitFC2-NP
1110 $0v_2v_1v_0$	InitFC2-Cpl
1000 $0v_2v_1v_0$	UpdateFC-P
1001 $0v_2v_1v_0$	UpdateFC-NP
1010 $0v_2v_1v_0$	UpdateFC-Cpl
All other encodings	Reserved

- For Ack and Nak DLLPs (see Figure 3-5):
  - 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 elsewhere in this chapter.
- For InitFC1, InitFC2, and UpdateFC DLLPs:
  - The HdrFC field contains the credit value for Headers of the indicated type (P, NP, or Cpl)
  - The DataFC field contains the credit value for payload Data of the indicated type (P, NP, or Cpl)
  - The packet formats are shown in Figure 3-6, Figure 3-7, and Figure 3-8
  - Transmission is triggered by the Data Link Layer when initializing Flow Control for a Virtual Channel (see Section 3.3), and following Flow Control initialization by the Transaction Layer according to the rules in Section 2.9
  - Checked for integrity on reception by the Data Link Layer, then the information content of the DLLP is passed to the Transaction Layer

  Note: InitFC1 and InitFC2 DLLPs are used only for VC initialization
- Power Management (PM) DLLPs (see Figure 3-9):
  - Transmission is triggered by the component's power management logic according to the rules in Chapter 6
  - Checked for integrity on reception by the Data Link Layer, then passed to the component's power management logic
- Vendor Specific (see Figure 3-10)

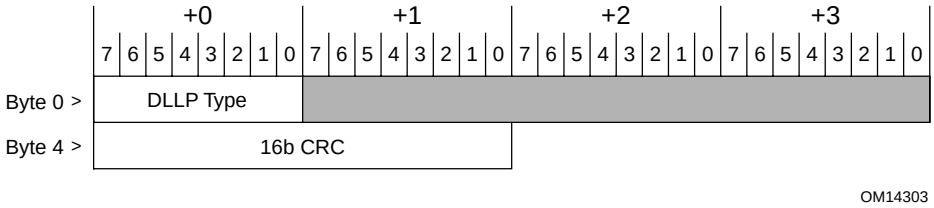

Figure 3-4: DLLP Type and CRC Fields

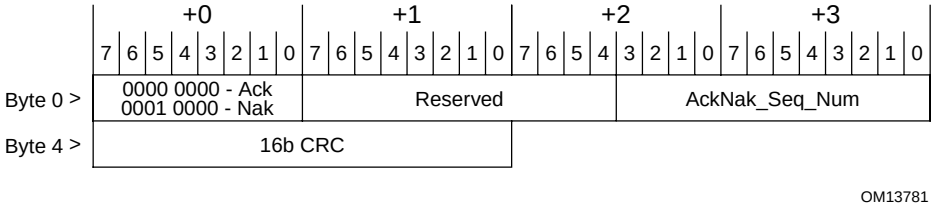

Figure 3-5: Data Link Layer Packet Format for Ack and Nak

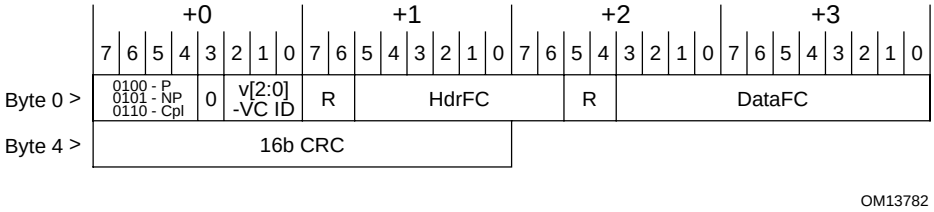

Figure 3-6: Data Link Layer Packet Format for InitFC1

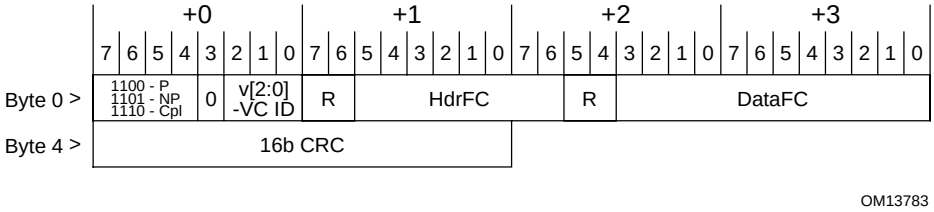

Figure 3-7: Data Link Layer Packet Format for InitFC2

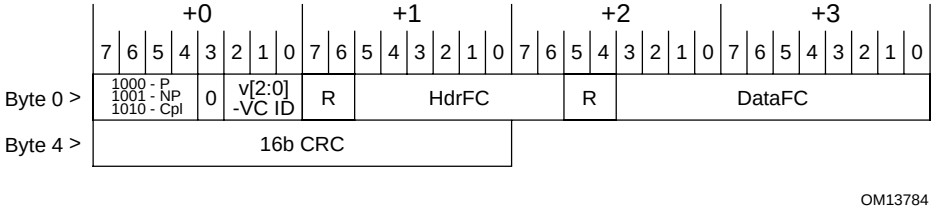

Figure 3-8: Data Link Layer Packet Format for UpdateFC

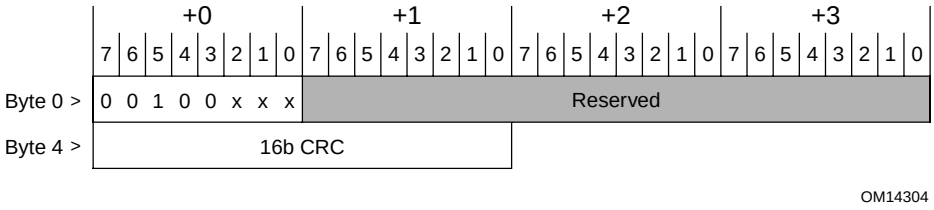

Figure 3-9: PM Data Link Layer Packet Format

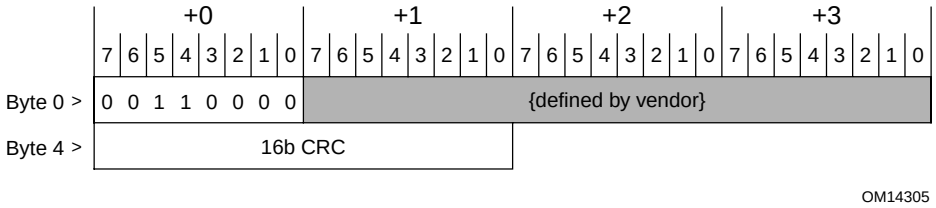

Figure 3-10: Vendor Specific Data Link Layer Packet Format

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

- DLLPs are differentiated from TLPs when they are presented to, or received from, the Physical Layer.
- DLLP data integrity is protected using a 16b CRC
- The CRC value is calculated using the following rules (see Figure 3-11):
    - The polynomial used for CRC calculation has coefficients 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 16b CRC field of the DLLP as shown in Table 3-2.

Table 3-2: Mapping of Bits into CRC Field

CRC Result Bit	Corresponding Bit Position in the 16b 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

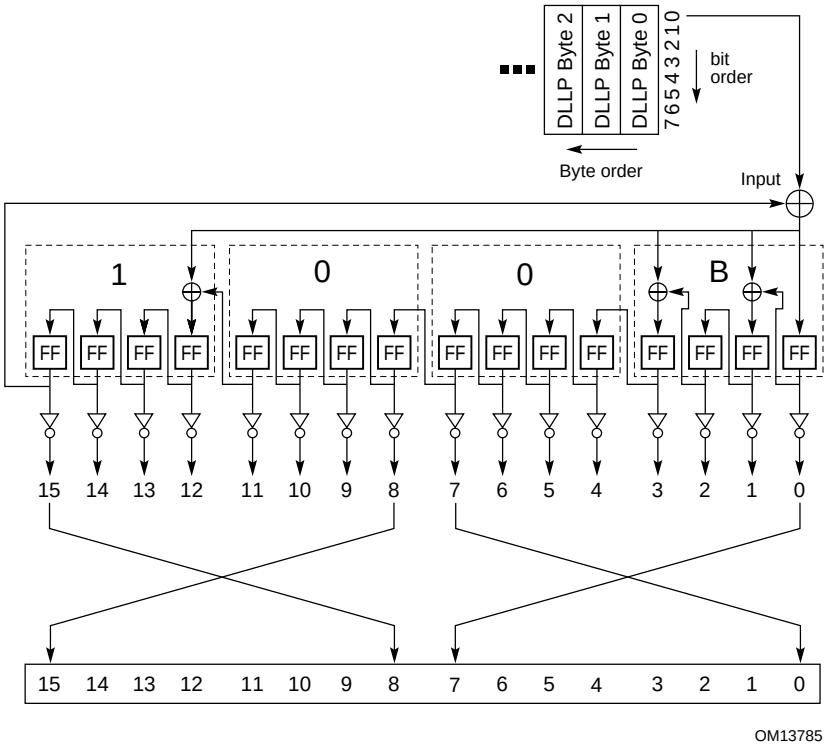

Figure 3-11: Diagram of CRC Calculation for DLLPs

3.5. Data Integrity

3.5.1. Introduction

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

The format of a TLP with the Sequence Number and LCRC code applied is shown in Figure 3-12.

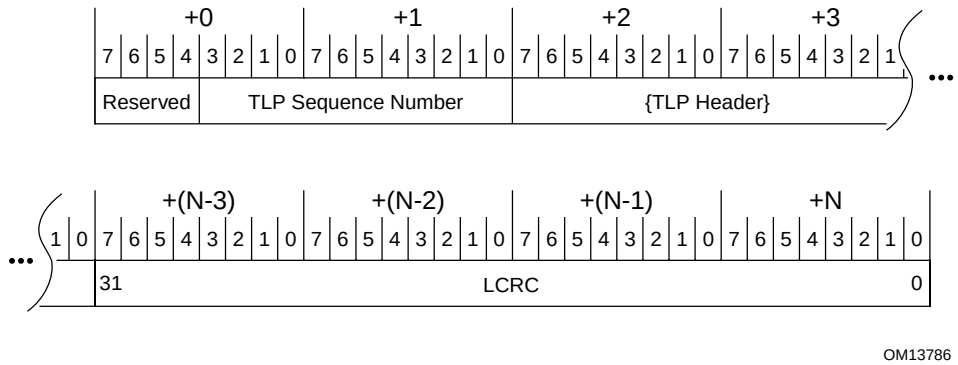

Figure 3-12: TLP with LCRC and Sequence Number Applied

3.5.2. LCRC, Sequence Number, and Retry Management (TLP Transmitter)

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

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

3.5.2.1. LCRC and Sequence Number Rules (TLP Transmitter)

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

- The following 12 bit counters are used:
  - NEXT_TRANSMIT_SEQ – Stores the packet sequence number applied to TLPs
    - Set to all '0's in DL_Inactive state
  - ACKD_SEQ – Stores the sequence number acknowledged in the most recently received Ack or Nak DLLP.
    - Set to all '1's in DL_Inactive state
- The following 2 bit counter is used:
  - REPLAY_NUM – Counts the number of times the Retry Buffer has been re-transmitted
    - Set to all '0's in DL_Inactive state
- The following timer is used:
  - REPLAY_TIMER - Counts time since last Ack or Nak DLLP received
    - Started at the start of any TLP transmission or retransmission, if not already running
    - Restarts for each Ack/Nak DLLP received while there are unacknowledged TLPs outstanding, if, and only if, the received Ack or Nak DLLP acknowledges some TLP in the retry buffer
      - Note: This ensures that REPLAY_TIMER is reset only when forward progress is being made
    - Resets and holds when there are no outstanding unacknowledged TLPs

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

- The Transaction Layer indicates the start and end of the TLP to the Data Link Layer while transferring the TLP
  - The Data Link Layer treats the TLP as a "black box" and does not process or modify the contents of the TLP

- Each TLP is assigned a 12 bit sequence number when it is accepted from the Transmit side of Transaction Layer
    - Upon acceptance of the TLP from the Transaction Layer, the packet sequence number is applied to the TLP by:
        - prepending the 12 bit value in NEXT_TRANSMIT_SEQ to the TLP
        - prepending four Reserved bits to the TLP, preceding the sequence number (see Figure 3-12)
    - If the equation:

      (NEXT_TRANSMIT_SEQ – ACKD_SEQ) mod 4096 >= 2048

    is true, the Transmitter must cease accepting TLPs from the Transaction Layer until the equation is no longer true
    - Following the application of NEXT_TRANSMIT_SEQ to a TLP accepted from the Transmit side of Transaction Layer, NXT_TRANSMIT_SEQ is incremented:

      NEXT_TRANSMIT_SEQ := (NEXT_TRANSMIT_SEQ + 1) mod 4096

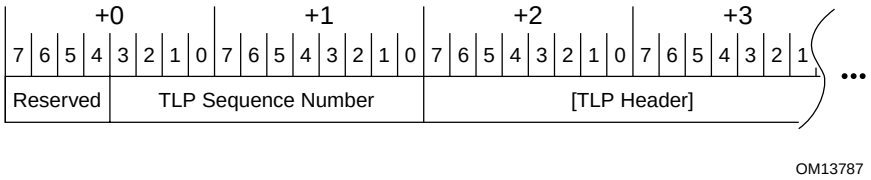

Figure 3-13: TLP Following Application of Sequence Number and Reserved Bits

- TLP data integrity is protected during transfer between Data Link Layers using a 32b LCRC
- The LCRC value is calculated using the following algorithm (see Figure 3-14)
    - 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 3-13)
    - 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 result of the LCRC calculation is complemented, and the complemented result bits are mapped into the 32b LCRC field as shown in Table 3-3.

Table 3-3: Mapping of Bits into LCRC Field

LCRC Result Bit	Corresponding Bit Position in the 32b 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
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 32b LCRC field is appended to the TLP following the bytes received from the Transaction Layer (see Figure 3-12)

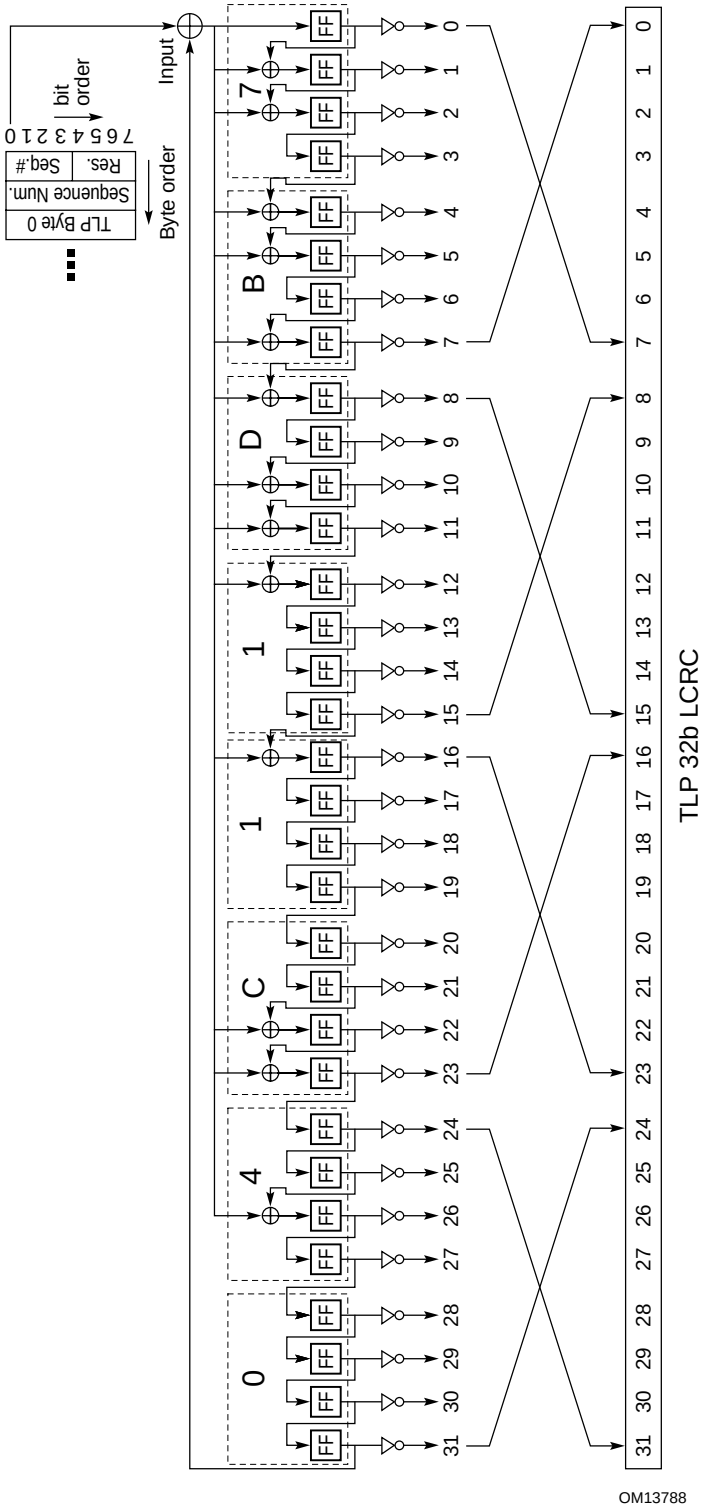

Figure 3-14: Calculation of LCRC

134

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

- A Transmitter is permitted to nullify a TLP being transmitted; to do this in a way which will robustly prevent misinterpretation or corruption, the Transmitter must do both of the following:
    - use the remainder of the calculated LCRC value without inversion
    - indicate to the Transmit Physical Layer that the final framing Symbol must be EDB instead of END
- When this is done, the Transmitter does not increment NEXT_TRANSMIT_SEQ

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

- Copies of Transmitted TLPs must be stored in the Data Link Layer Retry Buffer
- If the Transmit Retry Buffer contains TLPs for which no Ack or Nak DLLP has been received, and (as indicated by REPLAY_TIMER) no Ack or Nak DLLP has been received for a period exceeding the time indicated in Table 3-4, the Transmitter:
    - blocks acceptance of new TLPs from the Transmit Transaction Layer
    - completes transmission of the TLP currently being transmitted, if any
    - starts re-transmitting TLPs from the Retry Buffer, starting with the oldest TLP in the buffer and continuing in original transmission order
    - stops re-transmission from the Retry Buffer and increments REPLAY_NUM, if all entries in the Retry Buffer have been re-transmitted
    - Re-enables acceptance of new TLPs from the Transmit Transaction Layer

  This is a reported error associated with the Port (see Section 7.2).
- If REPLAY_NUM rolls over from "11" to "00" (indicating the Retry Buffer has been re-transmitted four times without receiving an Ack or Nak), the Transmitter signals the Physical Layer to retrain the Link. This is a reported error associated with the Port (see Section 7.2).
    - Note that Data Link Layer state, including the contents of the Retry Buffer, are not reset by this action unless the Physical Layer reports Physical LinkUp = 0 (causing the Data Link Control and Management State Machine to transition to the DL_Inactive state)

135

Table 3-4 defines the threshold values for the REPLAY_TIMER timer. The values are specified according to the largest TLP payload size and Link width.

The values are measured at the Port of the TLP Transmitter, from last Symbol of TLP to First Symbol of TLP retransmission. The values are calculated using the formula (note – this is simply three times the Ack Latency value – see Section 3.5.3.1):

$$\left(\frac{(Max_Payload_Size + TLPOverhead) * AckFactor}{LinkWidth} + InternalDelay\right) * 3$$

where

Max_Payload_Size	is the value in the Max_Payload_Size field of the Link Command Register
TLP Overhead	represents the additional TLP components which consume Link bandwidth (Header, LCRC, framing Symbols) and is treated here as a constant value of 24 Symbols
AckFactor	is used to balance Link bandwidth efficiency and retry buffer size – the value varies according to Max_Payload_Size and Link width, and is included in Table 3-5
LinkWidth	is the operating width of the Link
InternalDelay	represents the internal processing delays for received TLPs and transmitted DLLPs, and is treated here as a constant value of 11 Symbol Times

Table 3-4: REPLAY_TIMER Limits by Link Width and Max_Payload_Size (Symbol Times) Tolerance: -0% / +100%

		Link Operating Width						
		x1	**x2**	**x4**	**x8**	**x12**	**x16**	**x32**
Max_Payload_Size	**128B**	669	351	192	174	147	117	75
	256B	1209	621	327	294	243	189	111
	512B	1641	837	435	234	300	234	132
	1024B	3177	1605	819	426	555	426	228
	2048B	6249	3141	1587	810	1068	810	420
	4096B	12393	6213	3123	1578	2091	1578	804

Implementation Note: Recommended Priority of Scheduled Transmissions

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

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

1) completion of any transmission (TLP or DLLP) currently in progress (highest priority)

2) Nak DLLP transmissions

3) Ack DLLP transmissions scheduled for transmission as soon as possible due to
   receipt of a duplicate TLP –OR–
   expiration of the Ack latency timer (see Section 3.5.3.1)

4) FC DLLP transmissions required to satisfy Section 2.9

5) Retry Buffer re-transmissions

6) TLPs from the Transaction Layer

7) FC DLLP transmissions other than those required to satisfy Section 2.9

8) All other DLLP transmissions (lowest priority)

Since Ack/Nak and Flow Control DLLPs affect TLPs flowing in the opposite direction across the Link, the TLP transmission mechanisms in the Data Link Layer are also responsible for Ack/Nak and Flow Control DLLPs received from the other component on the Link. These DLLPs are processed according to the following rules (see Figure 3-15):

- 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 so 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 (above) to the received DLLP, not including the 16b CRC field of the received DLLP
    - comparing the calculated result with the value in the CRC field of the received DLLP
        - if not equal, the DLLP is corrupt
    - A corrupt received DLLP is discarded, and is a reported error associated with the Port (see Section 7.2).
- 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

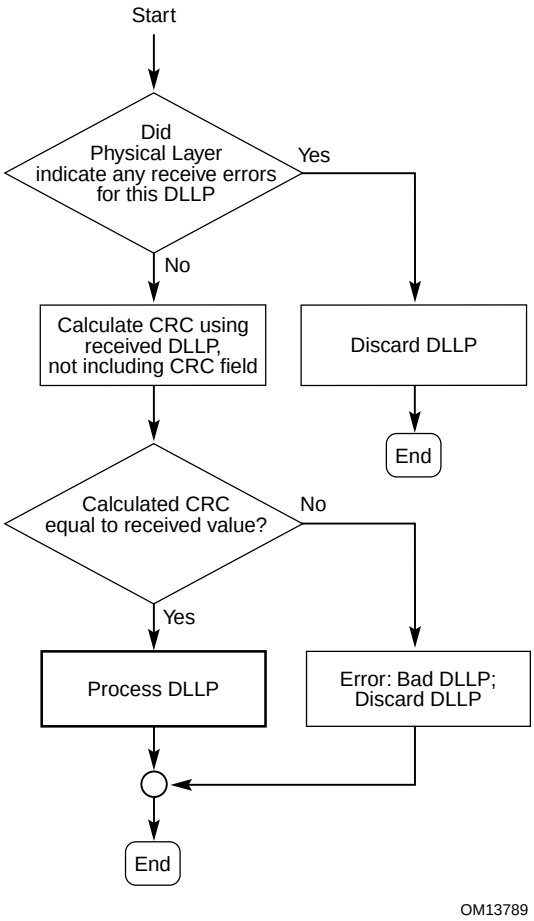

Figure 3-15: Received DLLP Error Check Flowchart

- Received FC DLLPs are passed to the Transaction Layer
- Received PM DLLPs are passed to the component's power management control logic
- For Ack and Nak DLLPs, the following steps are followed (see Figure 3-16):
    - If the AckNak_Seq_Num does not specify the Sequence Number of an unacknowledged TLP, or of the most recently acknowledged TLP, the DLLP is discarded
        - If the DLLP is an Ack DLLP, this is a DL Layer Protocol Error which is a reported error associated with the Port (see Section 7.2).

- If the AckNak_Seq_Num does not specify the Sequence Number of the most recently acknowledged TLP, then the DLLP acknowledges some TLPs in the retry buffer:
  - Purge from the retry buffer all TLPs from the oldest to the one corresponding to the AckNak_Seq_Num
  - Load ACKD_SEQ with the value in the AckNak_Seq_Num field
  - Reset REPLAY_NUM and REPLAY_TIMER
- If the DLLP is a Nak, initiate a replay (see below)
- If REPLAY_TIMER expires due to a failure to make progress on unacknowledged TLPs, initiate a replay. This is a reported error associated with the Port (see Section 7.2).

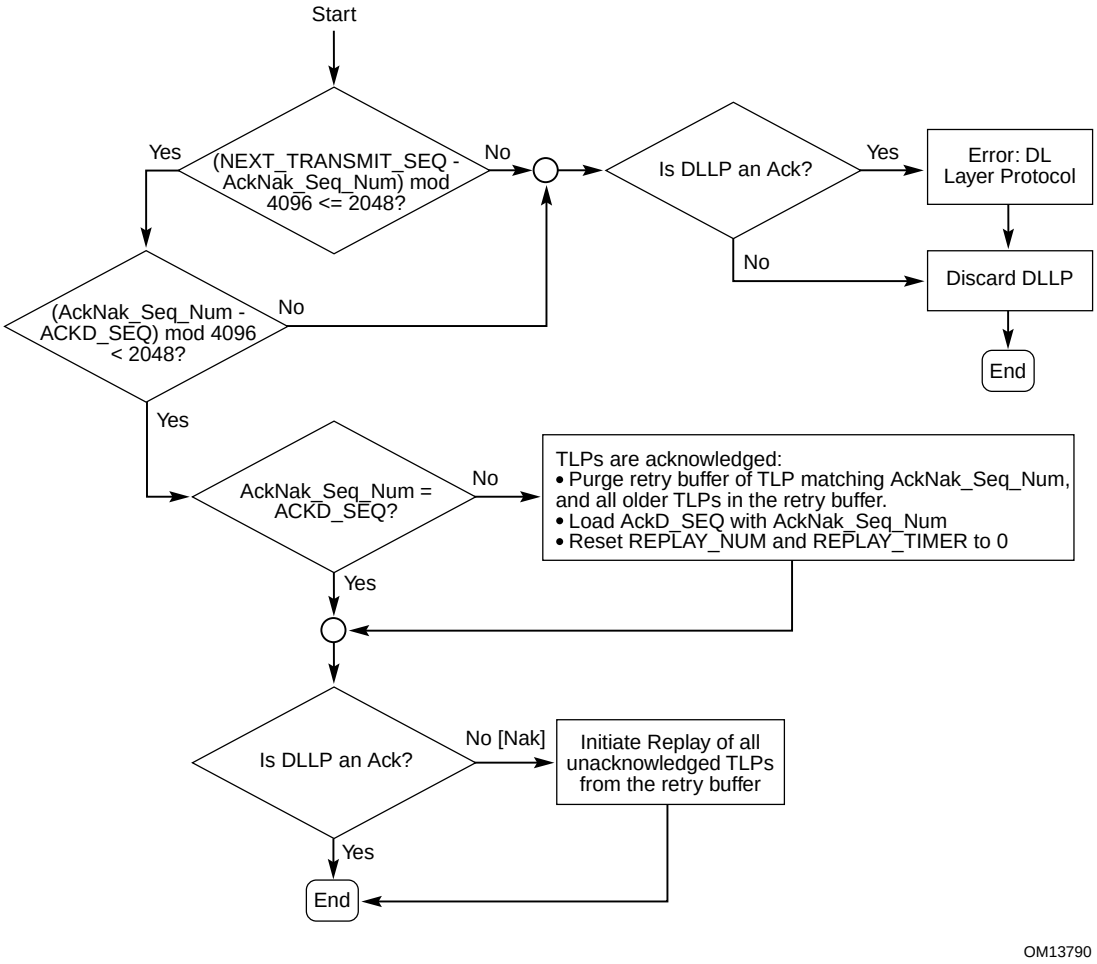

Figure 3-16: Ack/Nak DLLP Processing Flowchart

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

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

- If all TLPs transmitted have been acknowledged, terminate replay, otherwise continue
- Increment REPLAY_NUM
- Complete transmission of any TLP currently being transmitted
- Retransmit unacknowledged TLPs, starting with the oldest unacknowledged TLP and continuing in original transmission order
    - Once all unacknowledged TLPs have been re-transmitted, return to normal operation
    - If any Ack or Nak DLLPs are received during a replay, the transmitter is permitted to complete the replay without regard to the Ack or Nak DLLP(s), or to skip retransmission of any newly acknowledged TLPs
        - Once the transmitter has started to resend a TLP, it must complete transmission of that TLP in all cases
    - Ack and Nak DLLPs received during a replay must be processed, and may be collapsed
        - Example: If multiple Acks are received, only the one specifying the latest Sequence Number value must be considered – Acks specifying earlier Sequence Number values are effectively "collapsed" into this one
        - Example: During a replay, Nak is received, followed by an Ack specifying a later Sequence Number – the Ack supercedes the Nak, and the Nak is ignored
    - Note: Since all entries in the Retry Buffer have already been allocated space in the Receiver by the Transmitter's Flow Control gating logic, no further flow control synchronization is necessary.

141

3.5.3. LCRC and Sequence Number (TLP Receiver)

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

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

3.5.3.1. LCRC and Sequence Number Rules (TLP Receiver)

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

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

The following rules are applied in sequence to describe how received TLPs are processed, and what events trigger the transmission of Ack and Nak DLLPs (see Figure 3-17):

- If the Physical Layer indicates a Receiver Error, discard any TLP currently being received and free any storage allocated for the TLP. Note that reporting such errors to software is done by the Physical Layer (and so are not reported by the Data Link Layer).
    - If a TLP was being received at the time the receive error was indicated and the NAK_SCHEDULED flag is clear,
        - schedule a Nak DLLP for transmission
        - set the NAK_SCHEDULED flag
- If the Physical Layer reports that the received TLP end framing Symbol was EDB, and the LCRC is the logical NOT of the calculated value, discard the TLP and free any storage allocated for the TLP. This is not considered an error.
- The LCRC value is checked by:
    - applying the same algorithm used for calculation (above) to the received TLP, not including the 32b 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
                - set the NAK_SCHEDULED flag

            This is a reported error associated with the Port (see Section 7.2).
- 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)

143

- Otherwise, the TLP is out of sequence (indicating one or more lost TLPs):
  - if the NAK_SCHEDULED flag is clear,
    - schedule a Nak DLLP for transmission
    - set the NAK_SCHEDULED flag
    - report TLP missing

    This is a reported error associated with the Port (see Section 7.2).

- If the TLP Sequence Number is equal to the expected value stored in NEXT_RCV_SEQ:
  - The Reserved bits, Sequence Number, and LCRC are removed and the remainder of the TLP is forwarded to the Receive Transaction Layer
    - The Data Link Layer indicates the start and end of the TLP to the Transaction Layer 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

144

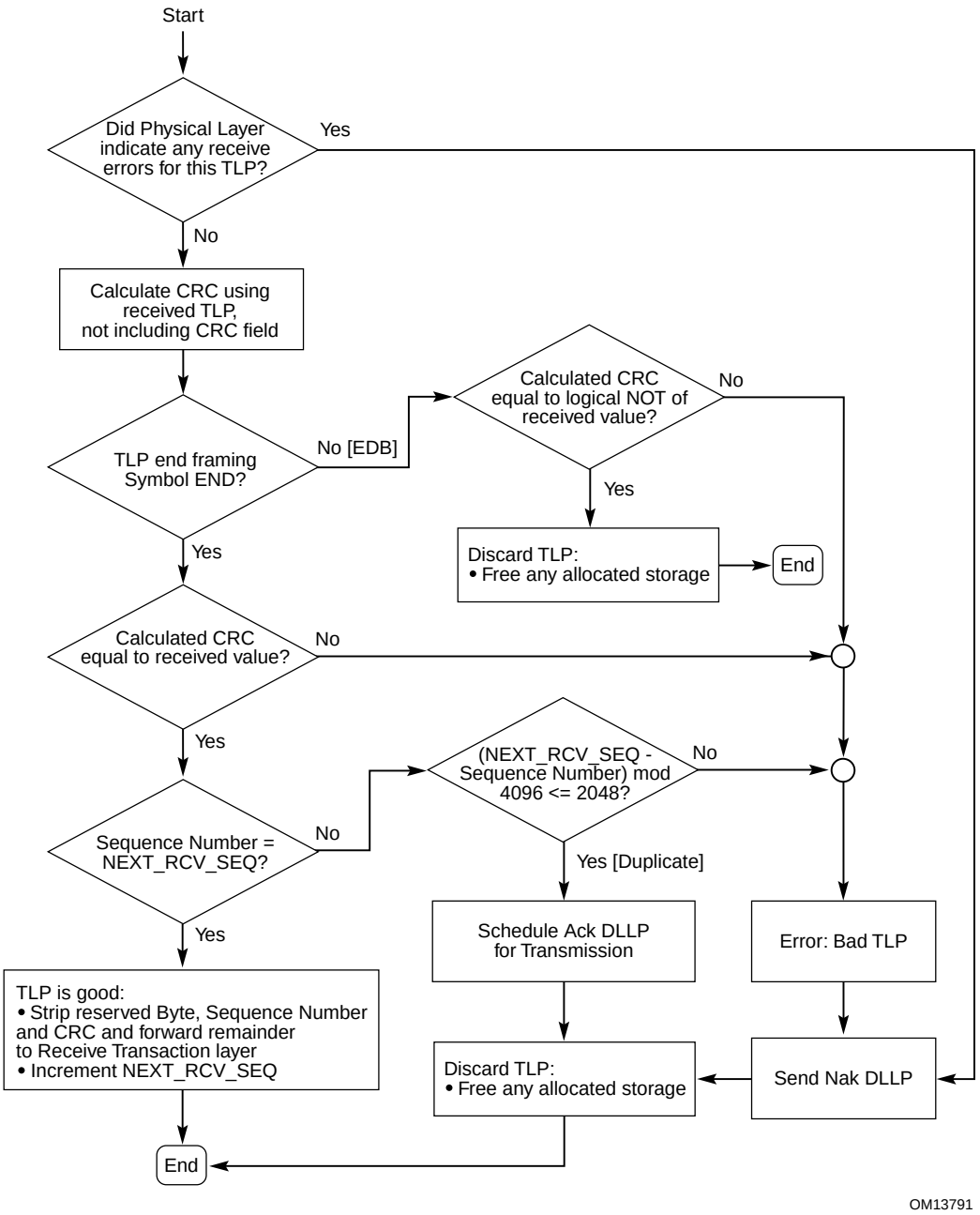

Figure 3-17: Receive Data Link Layer Handling of TLPs

- In addition to the other requirements for sending Ack DLLPs, an Ack or Nak DLLP must be transmitted when all of the following conditions are true:
    - The Data Link Control and Management State Machine is in the DL_Active state
    - TLPs have been accepted, but not yet acknowledged by sending an Acknowledgement DLLP
    - The AckNak_LATENCY_TIMER reaches or exceeds the value specified in Table 3-5
- Data Link Layer Acknowledgement DLLPs may be Transmitted more frequently than required
- Data Link Layer Ack and Nak DLLPs specify the value (NEXT_RCV_SEQ - 1) in the AckNak_Seq_Num field

Table 3-5 defines the threshold values for the AckNak_LATENCY_TIMER timer, which for any specific case is called the Ack Latency. The values are specified according to the largest TLP payload size and Link width. The values are measured at the Port of the TLP Receiver, starting with the time the last Symbol of a TLP is received to the first Symbol of the Ack/Nak DLLP being transmitted. The values are calculated using the formula:

$$\frac{(Max_Payload_Size + TLPOverhead) * AckFactor}{LinkWidth} + InternalDelay$$

where

Max_Payload_Size	is the value in the Max_Payload_Size field of the Link Command Register
TLP Overhead	represents the additional TLP components which consume Link bandwidth (Header, LCRC, framing Symbols) and is treated here as a constant value of 24 Symbols
AckFactor	is used to balance Link bandwidth efficiency and retry buffer size – the value varies according to Max_Payload_Size and Link width, and is defined in Table 3-5
LinkWidth	is the operating width of the Link
InternalDelay	represents the internal processing delays for received TLPs and transmitted DLLPs, and is treated here as a constant value of 11 Symbol Times

Table 3-5: Ack Transmission Latency Limit and AckFactor by Link Width and Max Payload (Symbol Times)

		Link Operating Width						
		x1	**x2**	**x4**	**x8**	**x12**	**x16**	**x32**
Max_Payload_Size	**128B**	223 AF = 1.4	117 AF = 1.4	64 AF = 1.4	58 AF = 2.5	49 AF = 3.0	39 AF = 3.0	25 AF = 3.0
	256B	403 AF = 1.4	207 AF = 1.4	109 AF = 1.4	98 AF = 2.5	81 AF = 3.0	63 AF = 3.0	37 AF = 3.0
	512B	547 AF = 1.0	279 AF = 1.0	145 AF = 1.0	78 AF = 1.0	100 AF = 2.0	78 AF = 2.0	44 AF = 2.0
	1024B	1059 AF = 1.0	535 AF = 1.0	273 AF = 1.0	142 AF = 1.0	185 AF = 2.0	142 AF = 2.0	76 AF = 2.0
	2048B	2083 AF = 1.0	1047 AF = 1.0	529 AF = 1.0	270 AF = 1.0	356 AF = 2.0	270 AF = 2.0	140 AF = 2.0
	4096B	4131 AF = 1.0	2071 AF = 1.0	1041 AF = 1.0	526 AF = 1.0	697 AF = 2.0	526 AF = 2.0	268 AF = 2.0

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 (Table 3-5), any differences between the actual implementation and the internal processing delay used to generate these values, and the delays caused by the physical Link interconnect.

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

148

4

4. Physical Layer Specification

4.1. Introduction

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

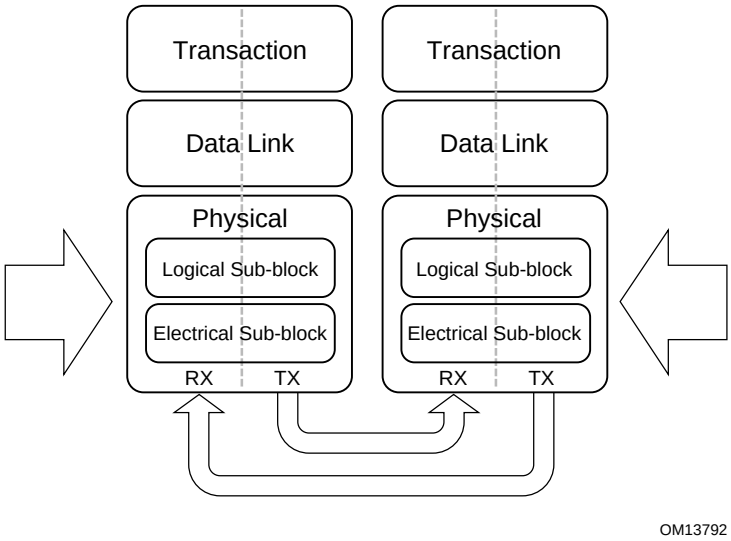

Figure 4-1: High Level Layering Diagram Highlighting Physical Layer

4.2. LOGICAL SUB-BLOCK

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

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

Receivers may optionally check for violations of the rules associated with Receiver functions such as Symbol decoding and the like. If such checking is implemented, violations cause the

indication of a Receiver Error to the Data Link Layer. A Receiver Error is a reported error associated with the Port (see Section 7.2).

4.2.1. Symbol Encoding

PCI Express uses an 8b/10b transmission code. The definition of this transmission code is identical to that specified in ANSI X3.230-1994, clause 11 (and also IEEE 802.3z, 36.2.4). Using this scheme, eight bit Characters and one control bit are treated as three bits and five bits mapped onto a four-bit code group and a six bit code group, respectivley. 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 ten-bit Symbol. As shown in Figure 4-2, ABCDE maps to abcdei and FGH maps to fghj.

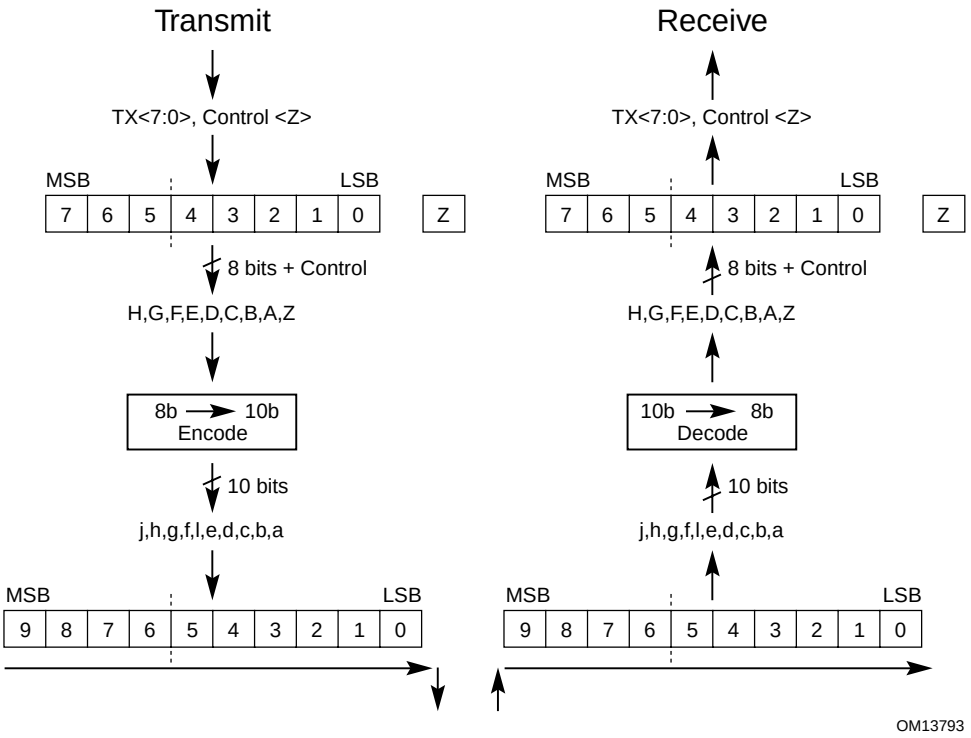

Figure 4-2: Character to Symbol Mapping

4.2.1.1. Serialization and De-serialization of Data

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

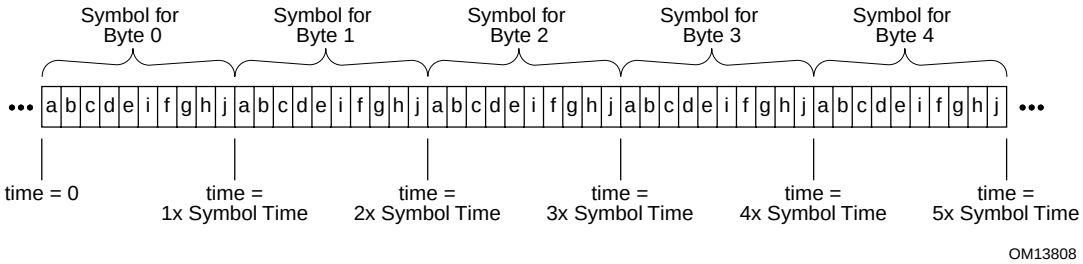

Figure 4-3: Bit Transmission Order on Physical Lanes - x1 Example

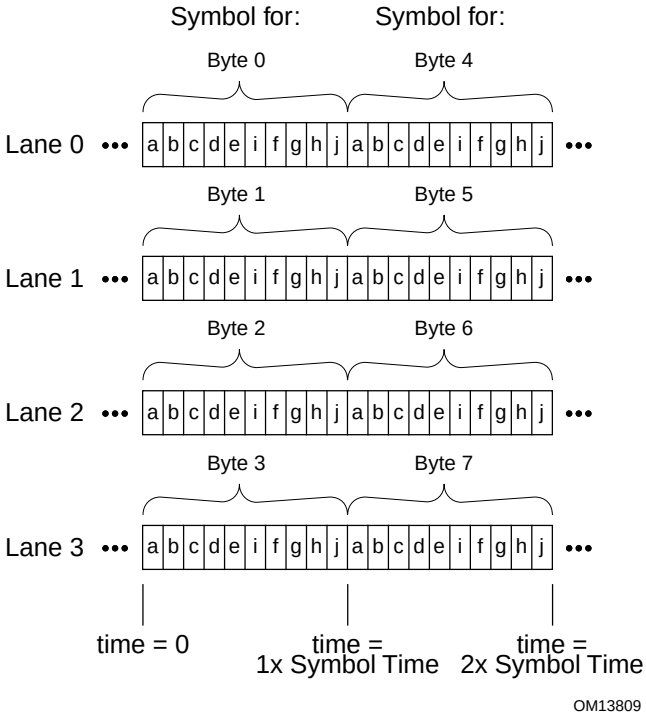

Figure 4-4: Bit Transmission Order on Physical Lanes - x4 Example

4.2.1.2. Special Symbols for Framing and Link Management (K codes)

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

Table 4-1 shows the Special Symbols used for PCI Express and provides a brief description for the use of each. The use of these Symbols will be discussed in greater detail in following sections.

Table 4-1: Special Symbols

Encoding	Symbol	Name	Description
K28.5	COM	Comma	Used for Lane and Link initialization and management
K27.7	STP	Start TLP	Marks the start of a Transaction Layer Packet
K28.2	SDP	Start DLLP	Marks the start of a Data Link Layer Packet
K29.7	END	End	Marks the end of a Transaction Layer Packet or a Data Link Layer Packet
K30.7	EDB	EnD Bad	Marks the end of a nullified TLP
K23.7	PAD	Pad	Used in Framing and Link Width and Lane ordering negotiations
K28.0	SKP	Skip	Used for compensating for different bit rates for two communicating ports
K28.1	FTS	Fast Training Sequence	Used within an ordered-set to exit from L0s to L0
K28.7			Reserved
K28.3	IDL	Idle	Electrical Idle symbol used in the electrical idle ordered-set
K28.4			Reserved
K28.6			Reserved
K28.7			Reserved

4.2.2. Framing and Application of Symbols to Lanes

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 4.2.3 and 8b/10b encoded according to the rules of Section 4.2.1, in the same way that TLP and DLLP data characters are scrambled and encoded. Likewise, when the Receiver is not receiving any packet information or special ordered-sets, the Receiver is in Logical Idle and shall receive idle data as described above. During transmission of the idle data, the skip ordered-set must continue to be transmitted as specified in Section 4.2.7.

4.2.2.1. Framing and Application of Symbols to Lanes – Rules

In this section, "placed" is defined to mean a requirement on the transmitter to put the symbol into the proper Lane of a Link.

- TLPs must be framed by placing an STP Symbol at the start of the TLP and an END Symbol or EDB Symbol at the end of the TLP (see Figure 4-5).
- DLLPs must be framed by placing an SDP Symbol at the start of the DLLP and an END Symbol at the end of the DLLP.
- 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 character (00h) is being transmitted and received.
    - When the Transmitter is in Logical Idle, the Idle data character (00h) shall be transmitted on all Lanes. This is scrambled according to the rules in Section 4.2.3.
    - Receivers must ignore incoming Logical 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 an 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.

Note: For x8 and wider Links, this means that STP and SDP Symbols can be 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 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.
    - Example: on a x8, if END 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 3.5.2.1 for information on the usage of EDB.
- Receivers may optionally check for violations of the rules of this section. If such checking is implemented, violations cause the indication of a Receiver Error to the Data Link Layer. A Receiver Error is a reported error associated with the Port (see Section 7.2).

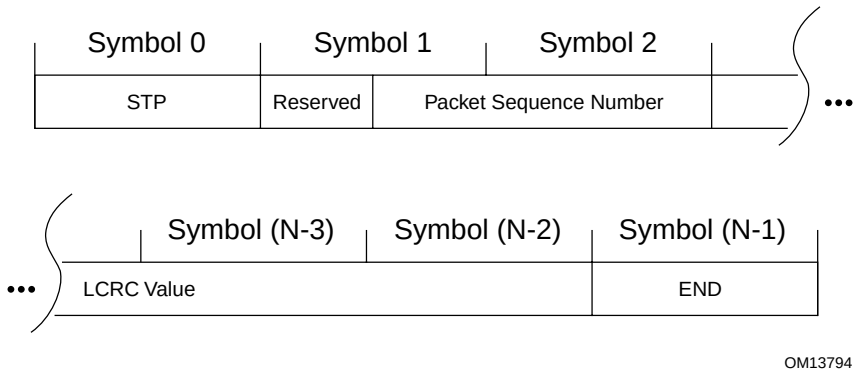

Figure 4-5: TLP with Framing Symbols Applied

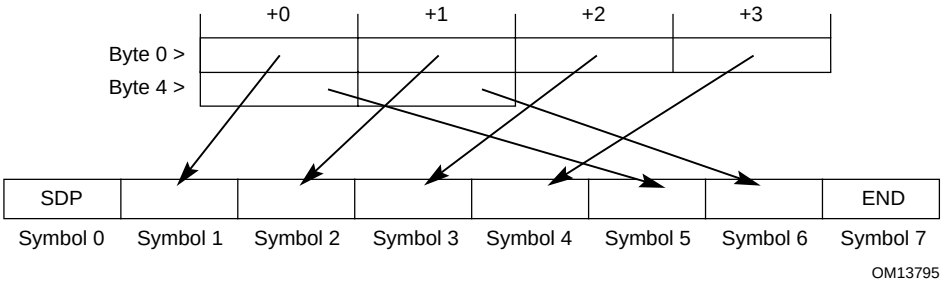

Figure 4-6: DLLP with Framing Symbols Applied

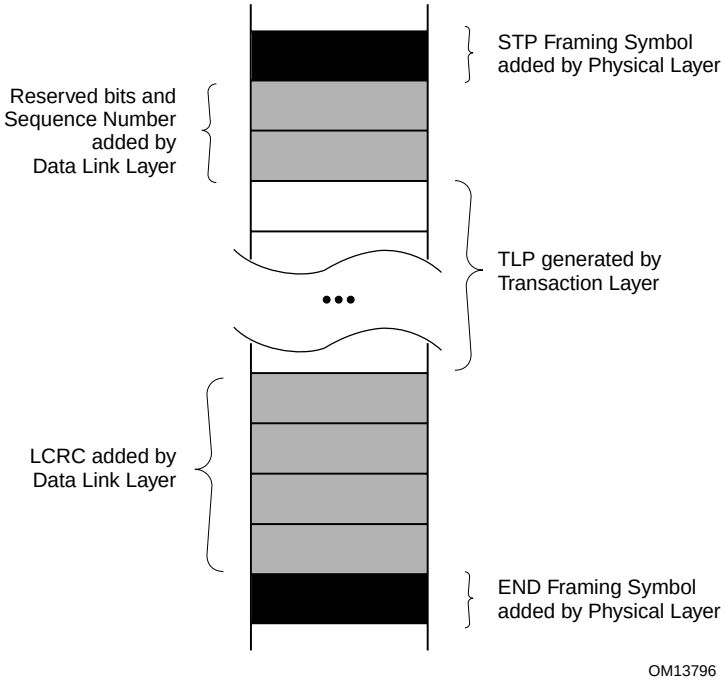

Figure 4-7: Framed TLP on a x1 Link

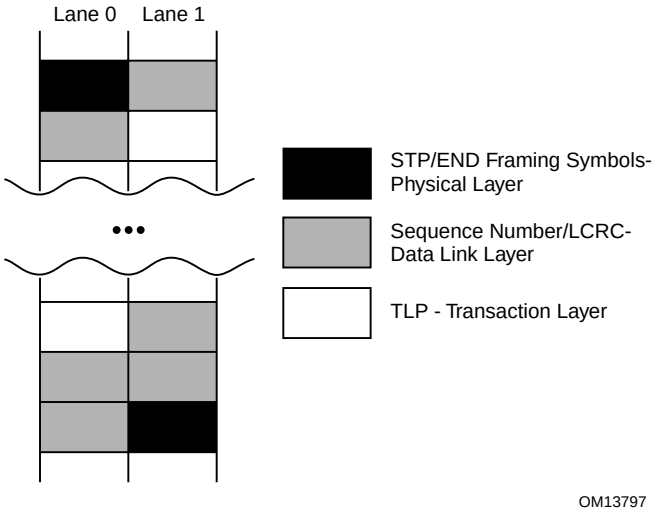

Figure 4-8: Framed TLP on a x2 Link

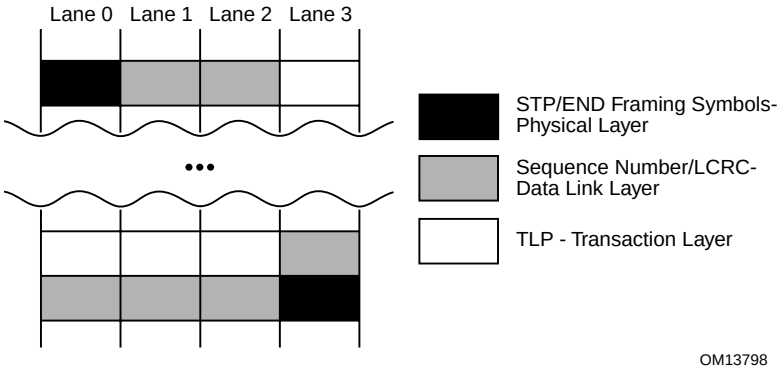

Figure 4-9: Framed TLP on a x4 Link

4.2.3. Data Scrambling

The scrambling function can be implemented with one or many Linear Feedback Shift Register's (LFSR's) on a multi-Lane Link. When there is more than one transmit LFSR per Link, these must operate in concert, maintaining the same simultaneous (see Table 4-4) value in each LFSR. When there is more than one receive LFSR per Link, these must operate in concert, maintaining the same simultaneous (see Table 4-5) value in each LFSR. Regardless of how it's implemented, the LFSRs must interact with data on a Lane-by-Lane basis as if there was a separate LFSR as described here for each Lane within that Link. On the transmit side, scrambling is applied to characters prior to the 8b/10b encoding. On the receive side de-scrambling is applied to characters after 8b/10b decoding.

The LFSR is graphically represented in Figure 4-10. Scrambling or unscrambling is performed by serially XORing the 8-bit (D0-D7) character with the 16-bit (S0-S15) output of the LFSR. An output of the LFSR, S15, 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:

$G(X)=X^{16}+X^{15}+X^{13}+X^{4}+1$

Data scrambling rules:

- The COM character initializes the LFSR.
- The LFSR value is advanced eight serial shifts for each character except the SKP.
- All data characters (D codes) except those within a Training Sequence Ordered-sets (TS1, TS2) and the Compliance Pattern are scrambled.
- All special characters (K codes) are not scrambled.

The initialized value of an LFSR seed (S0-S15) is 0FFFFh. 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 is enabled by default. It can be disabled for diagnostic purposes by setting bit 3 in symbol 5 of the training sequence ordered-sets. If a training sequence ordered-set \is received with this bit set in all Lanes, scrambling must be disabled until the next reset occurs. See Table 4-2 and Table 4-3.

For more information on scrambling, see Appendix C.

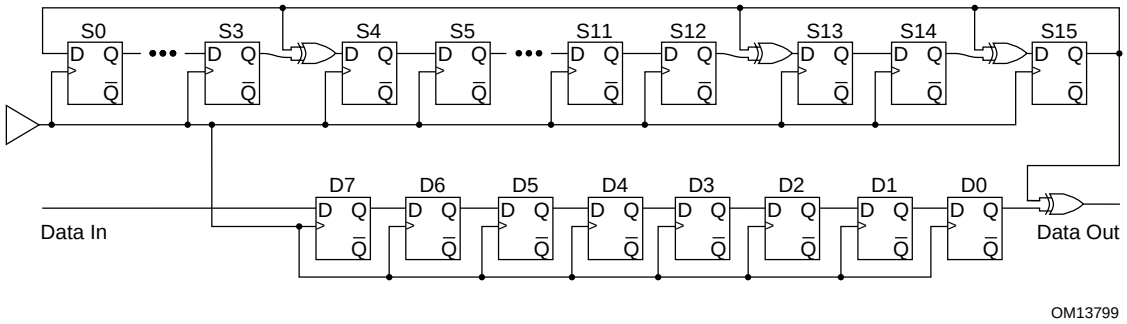

Figure 4-10: LFSR with Scrambling Polynomial

4.2.4. Link Initialization and Training

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

- Configuring and initializing the Link.
- Supporting normal packet transfers.
- Supported state transitions when recovering from Link errors.
- Restarting a Port from low power states.

The following are discovered and determined during the training process:

- Link width.
- Link data rate[11].
- Lane reversal.
- Polarity inversion.

Training does:

- Link data rate[12] negotiation.
- Bit synchronization per Lane.
- Lane polarity.
- Symbol synchronization per Lane.
- Lane ordering within a Link.
- Link width negotiation.
- Lane-to-Lane de-skew within a multi-Lane Link.

Receivers may optionally check for violations of the Link Initialization and Training Protocols. If such checking is implemented, any violation is a Training Error. A Training Error is a reported error associated with the Port (see Section 7.2). A Training Error is considered fatal to the Link.

4.2.4.1. Training Sequence Ordered-sets

Training sequences are composed of ordered-sets used for bit alignment, symbol alignment and to exchange physical layer parameters. Training sequence ordered-sets are never scrambled but are always 8b/10b encoded. SKP ordered-sets may be transmitted during training sequences but never interrupt a TS1 or TS2 ordered-set.

Any reference in the state machine section indicating that 16 ordered-sets are to be transmitted after receiving "n" number of ordered-sets means to send at least 16 additional ordered-sets after the reception of at least "n" ordered-sets. This is in addition to the ordered-sets sent while waiting for "n" ordered-sets to be received.

In order for N_FTS to be valid two or more TSx ordered-sets must be received with the same value.

Anytime two consecutive TS1 or TS2 ordered-sets are received in any state with the reset bit set the Link Control Reset state must be entered directly.

Anytime two consecutive TS1 or TS2 ordered-sets are received in any state with the Loopback Bit set, the Loopback state must be entered directly.

---

[11] This specification only defines one data rate. Future revisions will define additional rates.

[12] This specification defines the mechanism for negotiating the Link operational bit rate to the highest supported operational data rate.

Anytime two consecutive TS1 or TS2 ordered-sets are received in any state with the Disable Bit set, the Disable state must be entered directly.

When desired, Scrambling Disable bit must be set for all TS1 and TS2 sequences to ensure that scrambling will be disabled. If TS1 and TS2 are received with the Scrambling Disable bit set, scrambling is disabled for that entire Lane (both directions). Scrambling remains disabled until the Link is reset.

Skip ordered-sets may be sent between consecutive TS1 or TS2 ordered-sets. Idle data is not allowed between consecutive TS1 or TS2 ordered-sets.

The Link control bits for Scrambling Disable, Reset, Link Disable, and Loopback Enable are mutually exclusive, only one of these bits may be set at a time. If more than one of the Scrambling Disable, Reset, Link Disable or Loopback Enable bits are set the behavior is undefined.

Table 4-2: TS1 Ordered-Set

Symbol Number	Allowed Values	Encoded Values	Description
0		K28.5	COMMA code group for symbol alignment
1	0-255	D0.0 - D31.7, K23.7	Link Number within component
2	0-31	D0.0 - D31.0, K23.7	Lane Number within Port
3	0 – 255	D0.0 - D31.7	N_FTS. This is the number of fast training ordered-sets required by the receiver to obtain reliable bit and symbol lock.
4	1	D1.0	Data Rate Identifier Bit 0 – Reserved, set to 0 Bit 1 = 1, generation 1 (2.5 Gb/s) data rate supported Bit 2:7 – Reserved, set to 0
5	Bit 0 = 0, 1 Bit 1 = 0, 1 Bit 2 = 0, 1 Bit 3 = 0, 1 Bit 4:7 = 0	D0.0, D1.0, D2.0, D4.0, D8.0	Link Control Bit 0 = 0, De-assert Reset Bit 0 = 1, Assert Reset Bit 1 = 0, Enable Link Bit 1 = 1, Disable Link Bit 2 = 0, No Loopback Bit 2 = 1, Enable Loopback Bit 3 = 0, Enable Scrambling Bit 3 = 1, Disable Scrambling Bit 4:7, Reserved
6-15		D10.2	TS1 Identifier

Table 4-3: TS2 Ordered-Set

Symbol Number	Allowed Values	Encoded Values	Description
0		K28.5	COMMA code group for symbol alignment
1	0-255	D0.0 - D31.7, K23.7	Link Number within component
2	0-31	D0.0 - D31.0, K23.7	Lane Number within Port
3	0 – 255	D0.0 - D31.7	N_FTS. This is the number of fast training ordered-sets required by the receiver to obtain reliable bit and symbol lock.
4	1	D1.0	Data Rate Identifier Bit 0 – Reserved, set to 0 Bit 1 = 1, generation 1 (2.5 Gb/s) data rate supported Bit 2:7 – Reserved, set to 0
5	Bit 0 = 0, 1 Bit 1 = 0, 1 Bit 2 = 0, 1 Bit 3 = 0, 1 Bit 4:7 = 0	D0.0, D1.0, D2.0, D4.0, D8.0	Link Control Bit 0 = 0, De-assert Reset Bit 0 = 1, Assert Reset Bit 1 = 0, Enable Link Bit 1 = 1, Disable Link Bit 2 = 0, No Loopback Bit 2 = 1, Enable Loopback Bit 3 = 0, Enable Scrambling Bit 3 = 1, Disable Scrambling Bit 4:7, Reserved
6-15		D5.2	TS2 Identifier

4.2.4.2. Lane Polarity Inversion

During the training sequence, the receiver looks at symbols 6-15 of TS1 and TS2 as the indicator of Lane polarity inversion (D+ and D- are swapped). If Lane polarity inversion occurs, the TS1 symbols 6-15 received will be D21.5 as opposed to the expected D10.2. Similarly, if Lane polarity 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 Lanes.

4.2.4.3. Fast Training Sequence (FTS)

FTS is the mechanism that is used for bit and symbol synchronization when transitioning from L0s to L0. The FTS is used by the receiver to detect the exit from Electrical Idle and align the receiver's bit/symbol receive circuitry to the incoming data. See Section 4.2.5 for a description of L0 and L0s.

A single FTS training sequence is an ordered-set composed of one K28.5 (COM) symbol and three K28.1 symbols. The maximum number of FTS ordered-sets (N_FTS) is 255, providing a bit time synchronization of 4 * 255 * 10 * UI.

After initial power up, the N_FTS value is exchanged in the TS1/TS2 training sequence. N_FTS defines the number of FTS ordered-set that must be transmitted when transitioning from L0s to L0. For the data rate in this specification, this corresponds to a bit lock time of 16 ns to 4 μs.

When transitioning from L0s to L0, the receiver shall observe the period of time from Electrical Idle Exit to the time that the receiver obtains bit and symbol alignment. If the N_FTS period of time expires prior to the receiver obtaining alignment on all lanes of a Link, the receiver must transition to the Recovery state in order to recover the Link alignment. This sequence is detailed in the LTSSM in Section 4.2.5.

4.2.4.4. Link Error Recovery

At any time the Physical Layer can be directed to enter the Recovery state, as described in Section 3.5. Refer to Section 7.2 for more information on behavior when the physical layer reports errors.

4.2.4.5. Link Reset

There are two types of reset, one at the physical layer that is platform specific ("Power Good Reset" or cold/warm reset) and one that is passed in the Link Control Register (bit number 0 of symbol 5) in the TS1 and TS2 ordered-sets. This reset is called the Link Control Reset or hot reset.

4.2.4.5.1. Physical Layer Reset ('Power Good"

A Physical Layer Reset is provided by the system to the logical sub-block and is used to properly initialize the port. This Physical Layer Reset must be asserted when the power to the device does not meet the device specifications. This Physical Layer Reset may also be asserted by other control agents in the device (for instance the Link Layer, the Transaction Layer or a software mechanism) to assert reset to the Physical Layer. The following must be met when this reset is asserted:

- The receiver terminations are disabled.
- The transmitter must hold a constant DC common mode voltage on the differential pair using a high impedance driver. For the definition of high impedance in this context, see Table 4-4.

4.2.4.5.2. Link Control Reset (Hot Reset)

In addition to Physical Layer Reset, a Protocol Reset is defined. This Link Control Reset uses a reset indicator bit defined in the Link Control Register (Table 4-2, Table 4-3) that is sent during the training sequence. An upstream device sets this bit to force a reset of all of the downstream devices and links. Optionally a downstream device may use this reset to reset other logic within the device. The method and mechanisms to do this is implementation specific.

When a bridge receives a training sequence with the reset bit asserted, it must propagate that reset onto all downstream links by transmitting the TS1 ordered-sets with the reset bit asserted. Link Control Reset shall not propagate upstream. All other physical layer information exchanged in those ordered-sets must be accurate and correct.

All Lanes within a multi-Lane Link transmit the TS1, TS2 ordered-sets during Link Control Reset. When Link Control Reset is removed, each transmitter and receiver must enter Detect.

Unless otherwise specified the terms "reset" and "power-on/reset" in this chapter refer to the Physical Layer Reset.

4.2.4.6. Link Disable

A Link can be disabled if directed. When directed to this state the following behavior occurs:

- The Port drives its transmitters to high impedance.
- The receiver terminations must be disabled.
- There should be no response to any received data.

When directed to disable a Link, all Lanes within a multi-Lane Link transmit a minimum of 4 and a maximum of 16 TS1 ordered-sets with the Disable Link bit set. The Link remains disabled until directed or a physical reset occurs.

After a physical reset or after being directed out of Link disable, the next state is Detect.

4.2.4.7. Link Data Rate Negotiation

All devices are required to initialize and configure with generation 1 data rate on each Lane. During initialization, a field is passed in the training sequence (see Section 4.2.4) to indicate the maximum capable data rate for the Lane. This document specifies the data rate of 2.5 Gb/s in each direction on each Lane.

4.2.4.8. Link Width and Lane Sequence Negotiation

PCI Express Links must consist of 1, 2, 4, 8, 12, 16, or 32 Lanes in parallel, referred to as x1, x2, x4, x8, x12, x16, and x32 links respectively. All Lanes within a Link shall transmit data based on the exact same frequency.

The negotiation process is described as a sequence of steps. The negotiation establishes values for Link Number and Lane Number for each Lane that is part of a valid Link; each Lane that is not part of a valid Link exits the negotiation with values of K23.7 (PAD-out of range) for Link Number and Lane number.

During Link width and Lane sequence negotiation, the two communicating ports must accommodate the maximum allowed Lane-Lane skew as specified by $L_{RX\text{-}SKEW}$ in Table 4-5.

Optional behaviors are described to comprehend fixed configuration components and components to be used in the implementation of advanced switching cross-links (Section 1.6). Annex specifications to this specification may impose other rules and restrictions that must be comprehended by components compliant to those annex specifications; it is the intent of this specification to comprehend interoperability for a broad range of component capabilities.

4.2.4.8.1. Required/Optional Port Behavior

The ability for a set of transceivers to become one port and form one Link or become multiple ports and form multiple links is optional.

A xN port must be capable of forming a xN Link as well as a x1 Link (where N can be 32, 16, 12, 8, 4, 2, and 1). All other widths between N and 1 are optional.

Support for Lane reversal at any and all ports is optional.

4.2.4.8.2. Steps to Negotiate the Width and Lane Ordering of Links

While in the configuration state, for Lanes that have successfully completed the bit synchronization, polarity inversion and symbol synchronization training, components negotiate the Link width and sequence of the Lanes within each Link via the steps of:

Step 1:

The upstream component initializes Link numbers:

An upstream component (downstream port) assigns unique Link numbers to groups of Lanes capable of being unique links[13]. Indivisible groups of Lanes (those that can only be configured as a Lane within one Link) must connect to at most one downstream component (upstream port). The initial Link numbers are presented on each Lane to the downstream component(s). Until indicated (step 3), Lane numbers are presented as K23.7 (PAD-out of range). Upstream ports present their Link numbers and Lane numbers as K23.7 (PAD-out of range).

Mechanism: The upstream component shall send out the TS1 ordered-sets with the assigned Link numbers inserted into the Link number field (symbol 1) on the groups of Lanes capable of being unique Links and the Lane number field (symbol 2) set to K23.7.

Example of a set of eight lanes on an upstream component capable of negotiating to become on x8 port when connected to one downstream component or two x4 ports when connected to two different downstream components: The upstream component (downstream port(s)) sends out TS1 ordered-sets with the Link number set to N on four lanes and Link number set to N+1 on the other four lanes. The Lane numbers are all set to K23.7. The resultant number of links that are formed as well as their width(s) is dependant upon the system configuration as well as the capabilities of the downstream components.

Note: From this point on the rules are written to describe how each individual Link is configured. Regardless of the number of links a component supports, each Link is negotiated with the same rules that follow. Lanes within unique (only capable of being configured into one Link) and aggregated (capable of being configured into more than one) links must comply with timing rules in (Section 4.2.4.9). Independent, unique links have independent timing and control of negotiation. Unique and aggregated links are mapped with one and only one PCI-to-PCI bridge structure (Section 1.4).

[13] The most flexible case being all Lanes could be separate x1 Links. The most restrictive case being all Lanes as part of one Link.

Step 2:

The upstream port (downstream component) responds with Link number assignments:

A downstream component (upstream port) assigns the Link number (label) by assigning a common Link number to each of its lanes connected to the upstream component (downstream port), where the assigned Link number is selected from one of the Link numbers the downstream component received from the upstream component. If the downstream component is restricted as to its placement of Lane number 0[14], it must select the Link number received on that Lane. If the downstream component is restricted to Link widths other than what is presented, it must only transition the Link number of the subset of lanes that it can support within the Link.

Mechanism: The upstream port (downstream component) shall send out the TS1 ordered-set with the assigned common Link number inserted into the Link number field (symbol 1) and the Lane number field (symbol 2) set to K23.7 for all lanes within the widths supported by the port. Lanes which cannot be included due to supported width restrictions shall continue sending TS1 ordered-sets with the Link number and Lane number fields both set to K23.7.

Example 1: a x8 port: The upstream port (downstream component) sends out TS1 ordered-sets with the Link number set to one of the Link numbers presented from the upstream component and the Lane number set to K23.7 on all 8 lanes. Per the example under step 1 above, it must choose between N and N+1. If the upstream port (downstream component) did not support Lane reversal, it must choose the Lane number presented on its Lane 0.

Example 2: a x16 port which is not capable of becoming a x8 Link, but is capable of being a x 4 Link: The upstream port (downstream component) sends out TS1 ordered-sets with the Link number set to the Link number presented from the downstream port (upstream component) on the four lanes it can support within the Link; the Lane numbers remain set to K23.7 on those for lanes. It shall send out TS1 ordered-sets with the Link numbers and Lane numbers set to K23.7 on the twelve remaining lanes.

Note per Step 2: There may be times when a upstream port (downstream component) may be connected to another upstream port (downstream component)(cross-link). The rule below defines the behavior in this situation. Support for this behavior is optional.

Upstream port (downstream component) connected to upstream port (downstream component):

[14] A simple example of this is the port does not support Lane reversal.

If after a minimum of 16 TS1 ordered-sets have been received on each Lane within the perspective Link, the upstream port (downstream component) has not received a Data Symbol for its Link number; the port may optionally assume the role of an downstream port and transition this port to Step 1. If this feature is not supported, it must maintain the values of K23.7 for Link and Lane number fields and exit the negotiation.

Step 3:

The downstream port (upstream component) initializes Lane numbers:

The downstream port (upstream component) must acknowledge the assigned Link number received from the upstream port (downstream component) by transitioning the Link number to the assigned Link number on each Lane to the upstream port (downstream component) as well as transitioning the Lane number fields to its preferred Lane numbers, while maintaining Link widths consistent with the width restrictions above. The preferred Lane numbers must be consecutive and one Lane number must be assigned to 0. If the downstream port (upstream component) has not received a Data Symbol for its Link number after receiving an additional 16 (or greater) TS1 ordered-sets on all lanes in the perspective Link, all lanes of the Link must maintain values of K23.7 for Link and Lane number fields and exit the negotiation. If the assigned Link number does not match any of its initial Link numbers, see note below.

Mechanism: The downstream port (upstream component) shall send out the TS1 ordered-set with Link number field (symbol 1) set to the assigned Link number and the Lane number field (symbol 2) set to its preferred number on all lanes which received a Link number from the upstream port (downstream component) that can be accommodated within the Link widths that port can support. It must transition the Link numbers and Lane numbers to K23.7 on the lanes that were previously rejected by the upstream port (downstream component) and any additional lanes the downstream port (upstream component) cannot accommodate within its supported Link widths.

Note per Step 3: There may be times when a downstream port may be connected to a downstream port. The rules below define the behavior in this situation. Support for this behavior is optional.

Downstream port (upstream component) connected to downstream port (upstream component):

If the downstream port (upstream component) receives an assigned Link number that does not match any of its initial Link numbers, it may optionally compare the received Link number to its initial Link number. If the received Link number is less, it must transition these lanes to Step 2, assuming the role of an upstream port of a downstream component. A downstream port (upstream component) must remain in Step 3 if its assigned Link number was greater than the received Link number or it does not support this feature. If after a minimum of 16 TS1 ordered-sets have been received on each Lane within the perspective Link, the downstream port (upstream

component) has not received a Link number that matches any of its initial Link numbers, it must maintain the values of K23.7 for Link and Lane number fields and exit the negotiation.

Step 4:

The upstream port (downstream component) responds with Lane number assignments:

The upstream port (downstream component) must accommodate Link width and Lane numbers presented by the downstream port (upstream component) if it is possible to do so (see system designer rules below in this section). If the upstream port (downstream component) can accept the Link width presented but not the Lane numbers, it must acknowledge with its preferred ordering of Lane numbers at this time. The preferred Lane numbers must be consecutive and one Lane number must be assigned to 0. If the upstream port (downstream component) is restricted to Link widths other than what is presented, it must only transition the Lane numbers on the subset of lanes that can be accommodated within the Link widths that port can support. It must transition the Link numbers and Lane numbers to K23.7 on the lanes that cannot be accommodated within the widths that port supports.

Mechanism: If the upstream port (downstream component) has not received Data Symbols for its Lane numbers after receiving an additional 16 or more TS1 ordered-sets, all perspective lanes of the Link must maintain values of K23.7 for Link and Lane number fields and exit the negotiation. If the upstream port (downstream component) can accommodate the Lane numbers received from the downstream port (upstream component), it shall send out the TS1 ordered-sets with Link number field (symbol 1) set to the assigned Link number and the Lane number field (symbol 2) set to the Lane numbers assigned by the downstream port (upstream component). Otherwise, it shall insert its preferred numbers on all lanes with Lane numbers currently assigned by the downstream port (upstream component) that it can accommodate within the Link widths that port can support. It must transition the Link numbers and Lane numbers to K23.7 on the lanes that were previously rejected by the downstream port (upstream component) and any additional lanes the upstream port (downstream component) cannot accommodate in the Link width.

Step 5:

The downstream port (upstream component) confirms Link number and Lane assignments:

The downstream port (upstream component) must accommodate any Lane numbers which are consistent with all system and component rules that do not match its preferred ordering, completing the Link width and numbering negotiation (see system designer rules below in this section). If the upstream port (downstream component) has assigned Lane numbers to a number of lanes resulting in Link width that port can not support, the downstream port (upstream component) must accommodate upstream port's (downstream component's) Lane 0, establishing a Link of width 1.

Mechanism: If the downstream port (upstream component) has not received Data Symbols for its Lane numbers after receiving an additional 16 or more TS1 ordered-sets, all lanes of the Link must maintain values of K23.7 for Link and Lane number fields and exit the negotiation. If the downstream port (upstream component) is to further reduce the width requested by the upstream port (downstream component) to a width greater than x1, the downstream port (upstream component) must return to step 3[15]. Otherwise, the downstream port (upstream component) shall transition to sending the TS2 ordered-sets with the Link number fields (symbol 1) and Lane number fields (symbol 2) set to negotiated values. It must transition the Link numbers and Lane numbers to K23.7 on the lanes that were previously rejected by the upstream port (downstream component) and any additional lanes the downstream port (upstream component) cannot accommodate in the Link width.

Step 6:

The upstream port (downstream component) confirms Link number and Lane assignments:

Mechanism: After receiving at least one TS2 ordered-set, the upstream port (downstream component) shall transition to sending the TS2 ordered-sets with the Link number fields (symbol 1) and Lane number fields (symbol 2) set to negotiated values. It must transition the Link numbers and Lane numbers to K23.7 on the lanes that were previously rejected by the downstream port (upstream component).

Step 7:

The downstream and upstream ports (upstream and downstream components, respectively) settle on Link number and Lane assignments:

Both ports continue sending TS2 ordered-sets. If after completing the negotiation (steps 1 – 6), either port again transitions the Lane numbers (should only occur if this Link is an advanced switching cross-link (refer to Section 1.6) where both ports have been implemented as downstream ports of upstream components), the port may optionally silently accept the received Lane numbers as the correct labeling of the other port's transmitters and therefore its own receivers; otherwise, all lanes of the Link must maintain values of K23.7 for Link and Lane number fields and exit the negotiation. No further changes to Link number and Lane number are allowed at this point without a complete re-training and re-configuration of the ports and associated Link(s). Label numbers are to be retained, skipping the Link width and Lane sequence negotiation steps unless transitioned to Link state Polling.Quiet. When these steps are skipped, the previously negotiated Link and Lane numbers are retained and inserted into the appropriate fields of the TS1 and TS2 ordered-sets.

---

[15] It is only possible to return to step 3 one time due to the limited number of Link widths allowed (x1, x2, x4, x8, x12, x16, x32). The longest sequence of width negotiation consists of an upstream component (downstream port), which supports x16, x8, x2, x1 connected to a downstream component (upstream port), which supports x32, x12, x4, x1. Step 1 would attempt to create a x16 Link, step 2 a x12 Link, step 3 (first pass) a x8 Link, step 4 (first pass) a x4 Link, step 3 (second pass) a x2 Link, step 4 (second pass) a x1 Link.

Mechanism: At least 16 TS2 ordered-sets are sent after receiving one TS2 ordered-set. If the received TS2 ordered-sets have Lane numbers that do not match those transmitted in the transmitted TS2 ordered-set, the port may internally disassociate the transmitter and receiver from the same Lane number label, associating the receiver with the Lane number received. The transmitter association to Lane number must not change once TS2 ordered-sets have been sent. If the receiver cannot be associated with the Lane number received, all lanes of the Link must maintain values of K23.7 for Link and Lane number fields and exit the negotiation. The port returns to Config.Idle after at least 8 TS2 ordered-sets are received.

All lanes that are connected to the other port but not included in the negotiated Link must maintain values of K23.7 for Link and Lane number fields assigned by upstream and downstream ports.

Any lanes that fail to establish Data Symbol values for Link and Lane number fields are inactive, and will not exchange information with the Data Link Layer. All data and control to the Data Link Layer from active lanes shall be consistent with the agreed Lane numbering. When negotiating Link width and Lane sequence; each downstream port (upstream component) must transition between steps in unison across all of its lanes and each upstream port (downstream component) must transition between steps in unison across all of its lanes.

The following graphical flow diagrams demonstrate interoperability of components with simplified negotiation machines. Components/ports with complex negotiation machines are added to facilitate clarity.

169

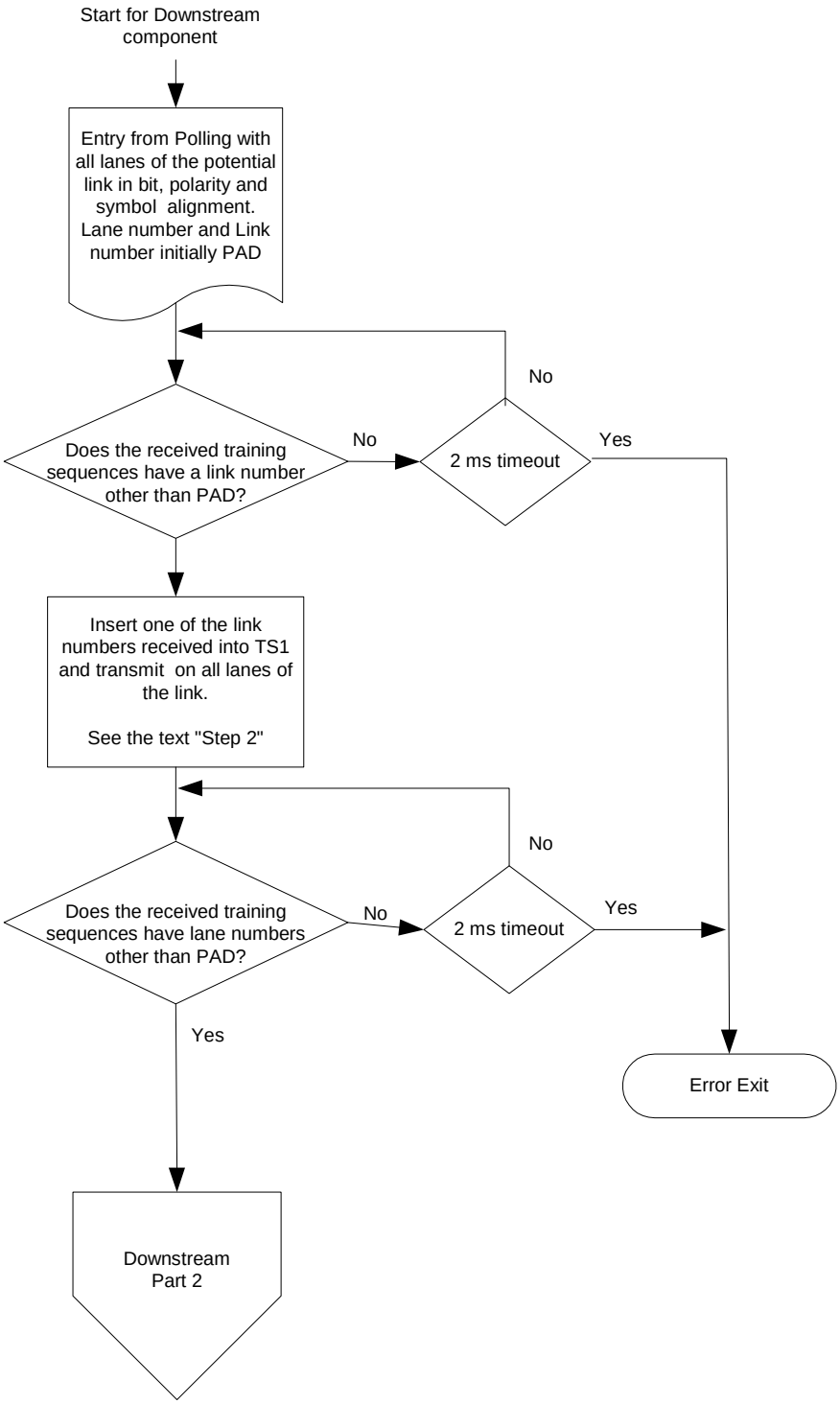

Figure 4-11: Width Negotiation, Simplified State Machine, Downstream Component (Part 1)

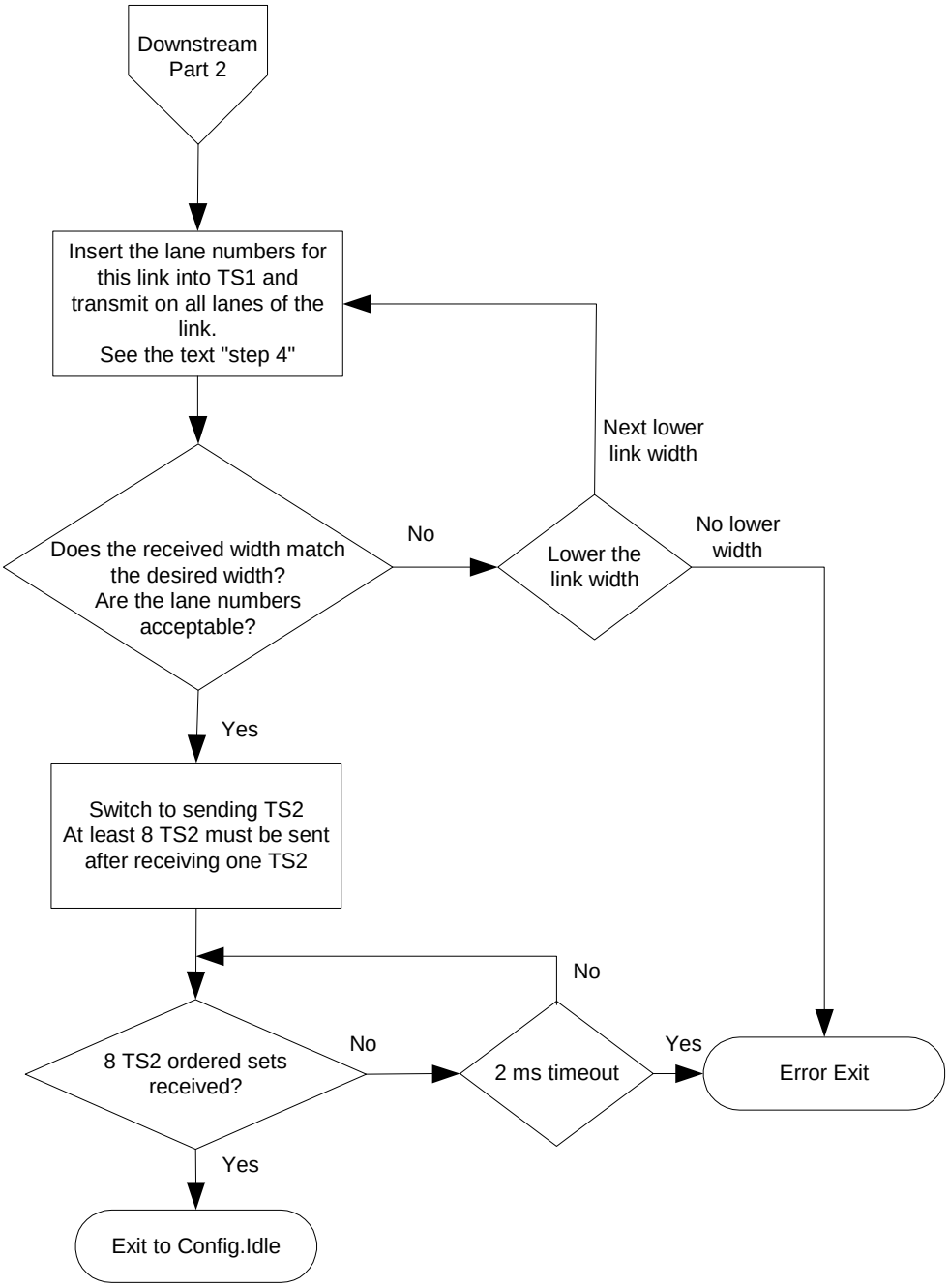

Figure 4-12: Width Negotiation, Simplified State Machine, Downstream Component (Part 2)

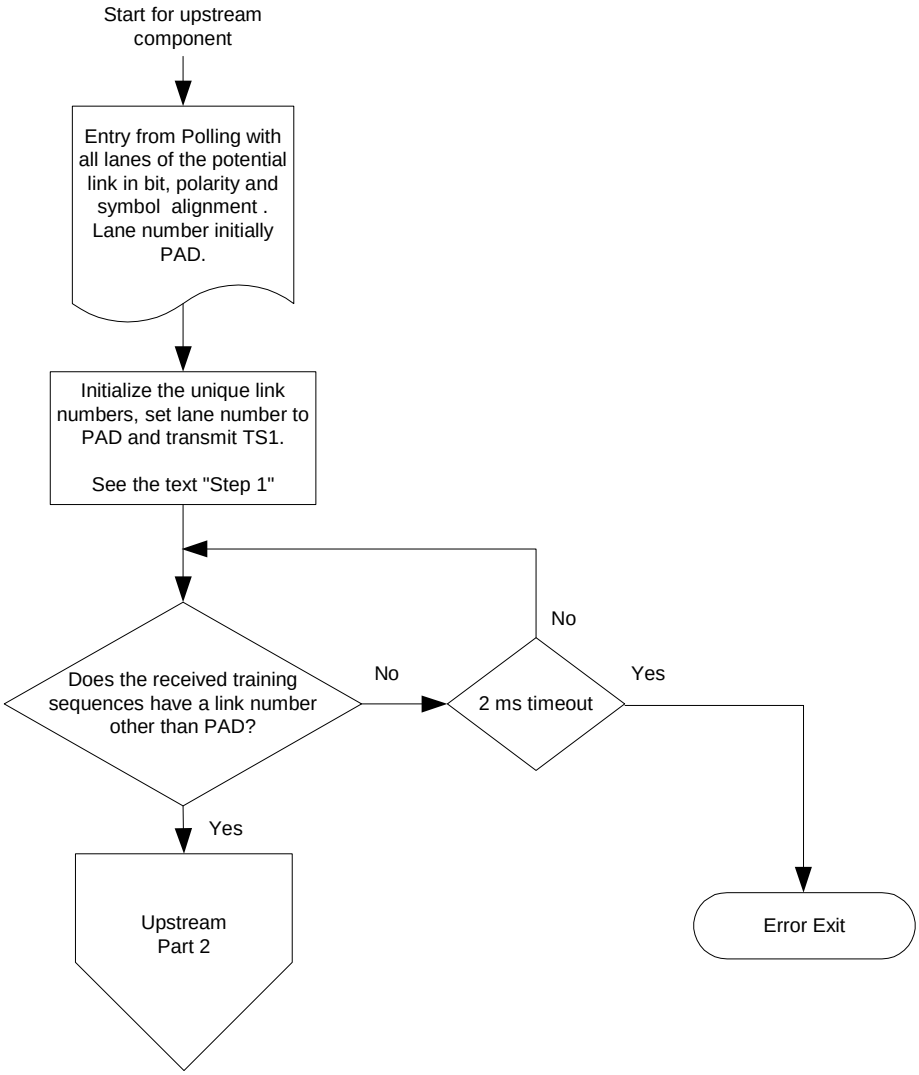

Figure 4-13: Width Negotiation, Simplified State Machine, Upstream Component (Part 1)

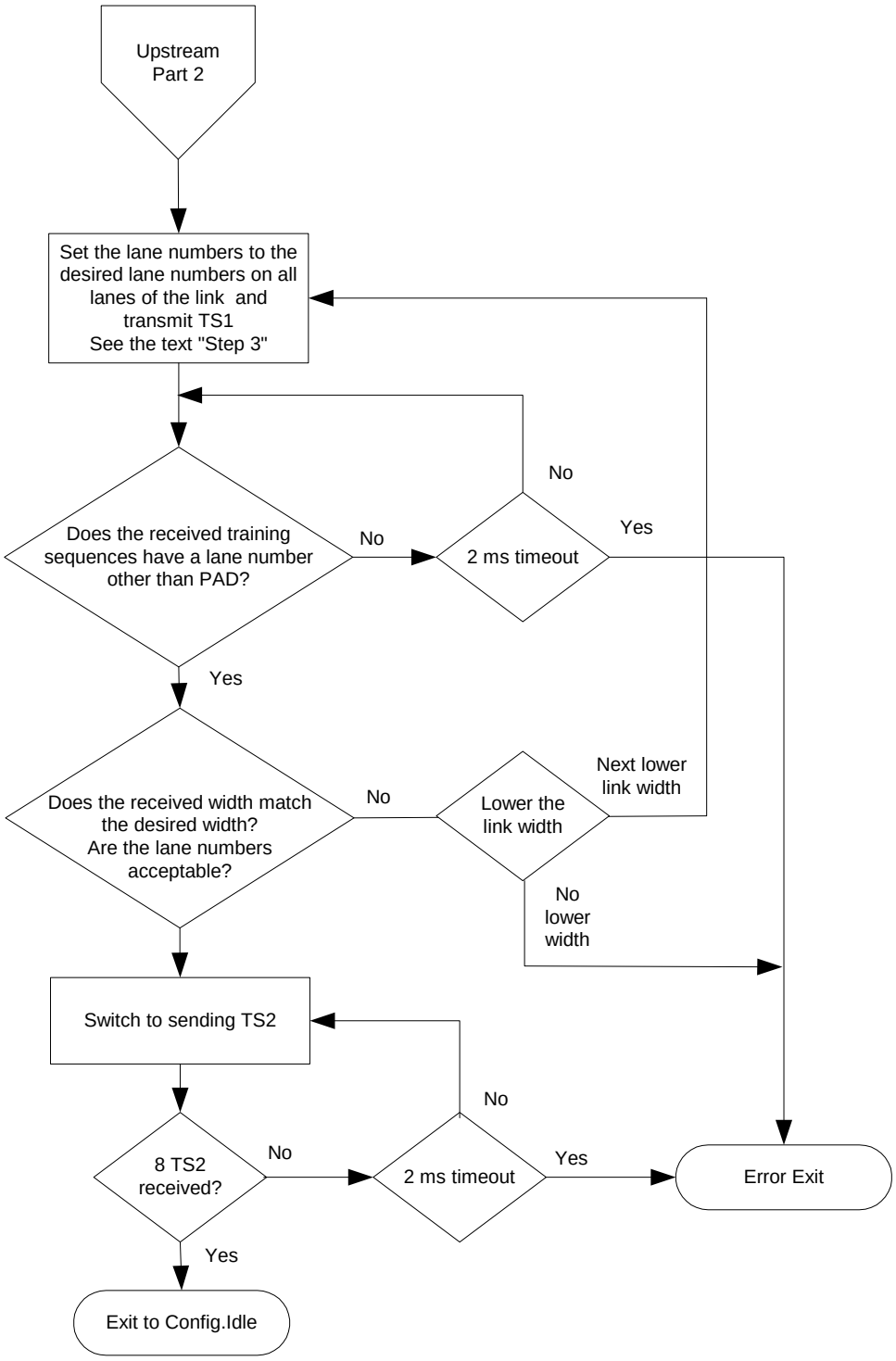

Figure 4-14: Width Negotiation, Simplified State Machine, Upstream Component (Part 2)

System Designer Rules:

System designers must connect Lanes within a Link interconnected through a connector or other suitable reference such that components are capable of labeling Lanes consistent with consecutive Lanes of the reference, inclusive of reference Lane 0. Components with fixed Lane ordering will interoperate with other compliant components flexible enough to also support other labelings. A simple example of increased flexibility would be to accommodate Lanes connected in reverse order. It is straightforward for a component to reverse Lane order upon receiving a Lane number of 0 on its most significant Lane or failure to detect an active in-bound Lane on its own Lane 0.

Example of Simple, Compliant Components:

Illustrated in Figure 4-15 are the transitions in the training exchange of Link and Lane numbers between an upstream component's 5th downstream Port that supports Link widths of x32, x12, x4, or x1 only and a downstream component's upstream Port that supports Link widths of x16, x8, x2, or x1 only. This is a worst-case scenario, with negotiation occurring at each step and the only common width is x1; at each step, the next largest Link width supported is implicit in the next exchange in the sequence. Link number is first (above), and Lane number second (below), with K representing the K23.7 symbol; transitions trigger the next step in the negotiation. The upstream component's (downstream port's) Lane transitions are shown in white and downstream component's (upstream port's) Lane transitions are shown in gray. Only the 16 Lanes that have successfully completed the bit synchronization, polarity reversal and symbol synchronization training are shown.

t	Port Lanes															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	K K	K K	K K	K K	K K	K K	K K	K K	K K	K K	K K	K K	K K	K K	K K	K K
1					5 K	5 K	5 K	5 K	5 K	5 K	5 K	5 K	5 K	5 K	5 K	5 K
2	K K	K K	K K	K K	K K	K K	K K	K K	5 K	5 K	5 K	5 K	5 K	5 K	5 K	5 K
3					K K	K K	K K	K K	K K	K K	K K	K K	5 3	5 2	5 1	5 0
4	K K	K K	K K	K K	K K	K K	K K	K K	K K	K K	K K	K K	K K	K K	5 1	5 0
5					K K	K K	K K	K K	K K	K K	K K	K K	K K	K K	K K	5 0

Figure 4-15: Width Negotiation Example

4.2.4.8.3. Port to Port Width Negotiation Example

The following text and diagrams show additional examples of two ports negotiating the width and Lane ordering of a Link.

Configuring groups of Lanes to form single logical links is done via a negotiation process that modifies the values of TS1 and TS2 symbols representing Link number and Lane number for each Lane. The process can be viewed as presentation of proposals and counter-proposals in the form of transitioning symbol values in discrete steps. Ordered-sets are symbol synchronized across Lanes that potentially make up a single logical Link, allowing the transitions of symbol values across Lanes to be examined together. The ordered-sets containing each proposal are repeated until a new counter-proposal is detected via the receipt of appropriate symbol transitions. The association of a Lane to a particular logical Link is indicated by its final Link number symbol and its position (ordering) within the Link is indicated by its final Lane number symbol. Lanes that have not been included in any logical Link will have final symbol values identical to the pre-negotiation value of PAD.

Steps 1 and 2 establish the number of links an upstream component (downstream port(s)) is attached to.

Steps 1 and 2 also begin (but not necessarily) complete establishing the width of the resultant Link(s).

175

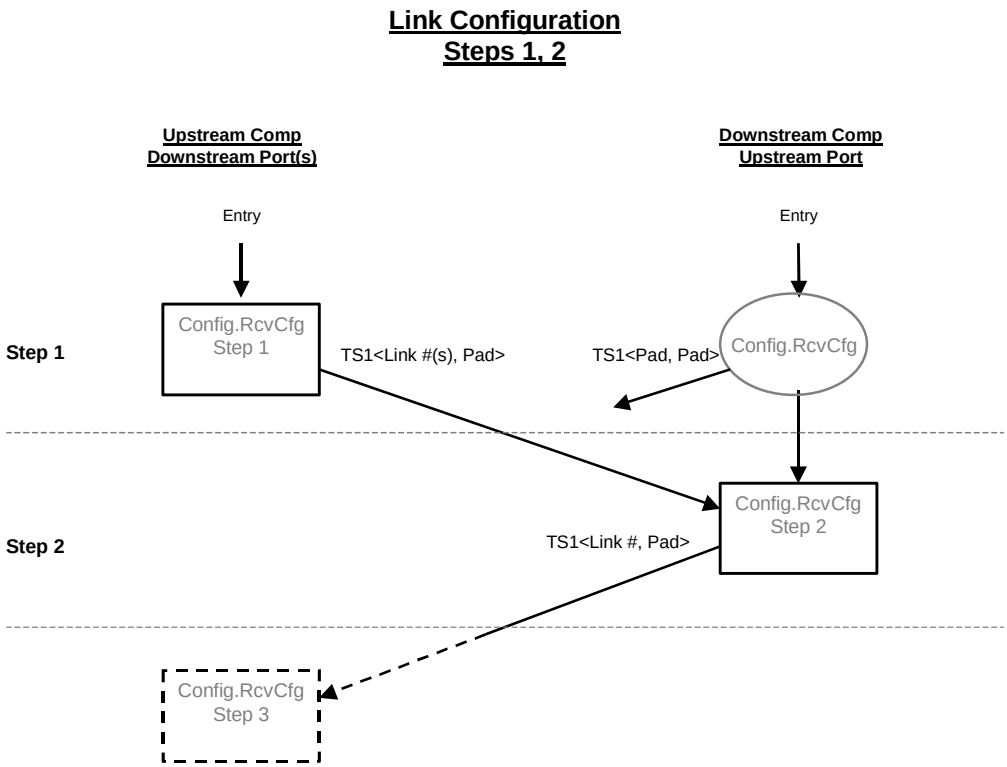

Figure 4-16: Link Width Negotiation; Steps 1,2

In order to enter the configuration state, Lanes within a perspective Link have already exchanged TS1 ordered-sets and completed the bit synchronization, polarity inversion (if needed) and symbol synchronization functions. Prior to entering the configuration state, the Link number and Lane number fields have been set to PAD (K23.7) and TS1 ordered-sets are sent repeatedly.

Step 1:

Upon entering Config.RcvrCfg, the downstream port(s) starts the Link width and Lane ordering negotiations by sending out the TS1 ordered-set with a unique Link number on sets of Lanes, which that component could support as unique links; the Lane numbers continue to be set to PAD.

Step 2:

Upon receipt of the TS1 ordered-set with Link numbers (non-PADs) present in the Link number field, the upstream port shall respond by choosing one of the Link numbers it received. This step of returning the one Link number determines the downstream port(s) the number of links that are to be negotiated.

The upstream port responds with a Link number only on the Lanes in which it received a Link number and Lanes that it can support in one Link. A simple example: a port may

be designed to support a x32 Link. Only 16 of those Lanes may have been attached, and therefore TS1s received only on 16 Lanes. The port may not support a x16 Link, but may support a x12 Link. In that case, the upstream port returns TS1 ordered-sets with a Link number only on the 12 Lanes that it is capable of supporting in a x12, and with the Link number set to PAD on the 4 remaining Lanes. This is the first counter-proposal towards establishing the final Link width.

Additional notes on steps 1 and 2:

One method to create a cross-link Section 1.6 is to connect a downstream port to another downstream port. One of two scenarios can occur; a.) The two ports choose different Link numbers to begin negotiations, or b.) The two ports choose the same Link number to begin negotiation. If a.) occurs, the rules are described as part of step 3. If b.) occurs, the two ports will not yet be able to differentiate the Step 1 behavior of a cross-link condition from the Step 2 behavior of a normal upstream port.

Note: If a system designer connects two (or more) downstream ports on one upstream component that is capable of being aggregated into one Link (Link aggregation) in a cross-link to two downstream ports on a different upstream component, the configuration results are undefined.

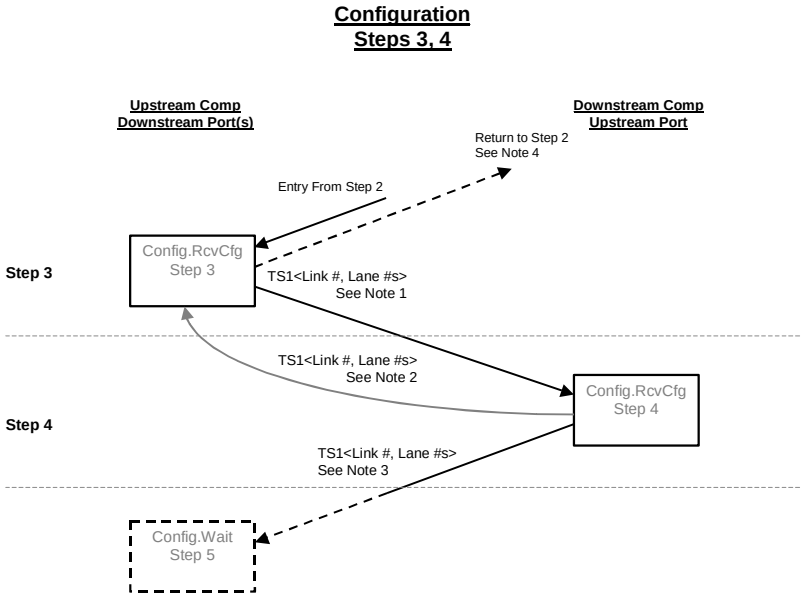

Figure 4-17: Link Width Negotiation; Steps 3, 4

Steps 3 and 4 establish Lane ordering within each Link established in Steps 1 and 2. To find a supported Link width common to both components, Steps 3 and 4 continue to reduce the Link width by removing select Lanes from the negotiation; Lanes never join a Link negotiation through these steps. Returning a Lane's Link number to the value of PAD indicates removal; otherwise Link numbers persist with the value assigned in Step 2.

177

Step 3:

Upon receipt of TS1 ordered-sets with a Link number inserted in that field on each Lane, the downstream port transitions to Step 3, making its first proposal for Lane numbers within each group of Lanes with common received Link numbers.

Note 1 (Figure 4-17): In the event that a set of ports were connected to a single upstream port, those ports would all see the same Link number returned. This is the mechanism that allows those ports to be aggregated into one Link. If those ports are not capable of being aggregated into one Link, the upstream component must continue negotiation with only one of those ports and transition the Link number to PAD on Lanes of the remaining ports, removing them from the negotiation process.

Note 2 (Figure 4-17): Components in a cross-link as described in scenario a.) above ,start negotiations by presenting different Link numbers to each other. Components designed to comprehend the cross-link condition implement the optional compare of the two Link numbers. The component that receives a Link number smaller than the Link number it presented on its port assumes the role of an upstream port and transitions to Step 2. The other component receives a Link number greater than the Link number presented on its port remains in Step 3 to await a further Link number transition matching its own.

Step 4:

Upon receipt of the TS1 ordered-set with Lane numbers presented in the Lane number fields and a common Link number present in the Link number fields, the upstream port transitions to Step 4, asserting an appropriate set of Lane numbers. The upstream port should only counter-propose Lane numbers if it has a fixed ordering of its Lanes and the downstream port is connected in a reversed Lane fashion; otherwise, Step 4 acknowledges the downstream port proposal.

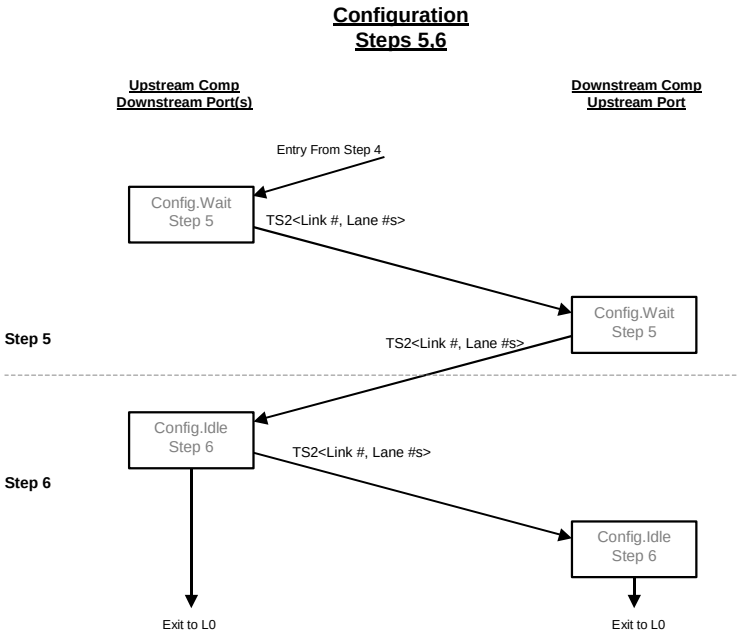

Figure 4-18: Link Width Negotiation; Steps 5, 6

Steps 5 and 6 acknowledge the completed Link width and Lane sequence negotiation.

Step 5:

The downstream port transitions to sending TS2 ordered-sets with the Link number and the Lane numbers inserted in the defined fields. At this time, the downstream port also removes Lanes that have been removed from the negotiation by the upstream port.

Note 3 (Figure 4-18): If additional Lanes are removed from the negotiation as may occur in the extreme mismatch of supported Link widths described in Step 4 of the rules, the downstream port can only transition to TS2 if a x1 Link is to be formed. Fewer additional Lanes removed resulting in a x2 Link require the downstream port to return to Step 3, remaining in TS1.

Step 6:

The upstream port transitions to sending TS2 ordered-sets when it receives TS2 ordered-sets with the agreed upon Link number and the Lane ordering it last presented to the downstream port. At least Lane 0 is retained in this step; the upstream port removes Lanes that have been removed from the negotiation by the downstream port.

Step 7: (not shown in Figure 4-18)

As noted earlier, there is a case where two downstream ports have been negotiating with each other and not realizing it until this step. This can occur in a cross-link when the two downstream ports both choose the same Link number to begin negotiation; both

ports present their Link number in their Step 1, receiving that same Link number in Step 3. In Step 3, they both presented their preferred Lane numbers; if connected such that these align, Step 5 will not modify Lane numbers and simply transition to sending TS2 ordered-sets. However, if one component is connected in reverse Lane fashion and both ports support Lane reversal, Step 5 will cause both to accommodate the other. The mechanism to observe this is when a TS2 ordered-sets arrives with Lane numbers that do not match the Lane numbers being transmitted. The optional behavior to resolve this conflict is to continue sending TS2 ordered-set with the agreed upon Link number and conflicting Lane numbers. However, the Lane numbers now represent the transmitter Lane number. The port must then disassociate its transmitter with a receiver and reverse the ordering of the receivers to match the Lane ordering of the other port.

4.2.4.9. Lane-to-Lane De-skew

Lane-to-Lane de-skew shall be done across all Lanes within multi-Lane links. An unambiguous de-skew mechanism is the COM symbol transmitted during training sequence or skip ordered-sets across all Lanes within the Link (at what the transmitter believes is) simultaneously. Other de-skew mechanisms may also be employed. The receiver must compensate for the allowable skew between Lanes within a multi-Lane Link before delivering the data and control to the Data Link Layer.

4.2.4.10. Lane vs. Link Training

The initialization Link training process builds unassociated Lanes on a device into associated Lanes that form a Link. This occurs during the first state of the configuration state machine Config.RcvrCfg where the links are configured (e.g. width negotiation and optional Lane reversal). State machines prior to the Config.RcvrCfg operate on a per Lane basis, after Config.RcvrCfg the operations are on a Link basis.

For example, transmitted data prior to Config.RcvrCfg sends the specified data on all Lanes of the device; after the Config.RcvrCfg state the transmitter sends the specified data on all Lanes of the configured Link.

4.2.5. Link Training and Status State Machine (LTSSM)

All timeout values specified in the Link training and status state machine (LTSSM) timeout values are minus 0 seconds and plus 50% unless explicitly stated otherwise. All timeout values must be set to the specified values after power-on/reset. All counter values must be set to the specified values after power-on/reset.

The LTSSM states are illustrated in Figure 4-19. These states are described in following sections.

4.2.5.1. Detect

The purpose of this state is to detect when a far end receiver is powered on in order to avoid transferring common mode between the transmitter and receiver.

4.2.5.2. Polling

The Port transmits training ordered-sets and responds to the received training ordered-sets. In this state bit lock and symbol lock are established, Lane polarity is configured, and Lane data rate is established.

4.2.5.3. Configuration

In Configuration both the transmitter and receiver are sending and receiving data at the negotiated data rate. The Link configures width and Lane reversal and manages Lane-to-Lane skew within the Link.

4.2.5.4. Recovery

In Recovery both the transmitter and receiver are sending and receiving data at the previously negotiated data rate. The Port transmits training ordered-sets and responds to the received training ordered-sets. In this state bit lock and symbol lock are re-established.

4.2.5.5. L0

L0 the normal operational state where data and control packets can be transmitted and received.

4.2.5.6. L0s

L0s is intended as a power savings state.

L0s allows a Link to quickly enter and recover from a power conservation state without going through the Configuration or Recovery states.

The entry to L0s occurs after receiving an Electrical Idle ordered-set.

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 receiver must be able to recover from this state within 64 μs, including reacquiring bit and symbol synchronization.

The entry to L1 occurs after being directed by the Data Link Layer and receiving an Electrical Idle ordered-set.

4.2.5.8. L2

Power can be aggressively conserved in L2. Most of the Transmitter and Receiver may be disabled[16]. Main power and clocks are not guaranteed, but aux[17] power is available.

An upstream port must be able to send and a downstream port must be able to receive a wakeup signal referred to as a Beacon.[18]

The entry to L2 occurs after being directed by the Data Link Layer and receiving an Electrical Idle ordered-set.

4.2.5.9. External Loopback

Loopback is intended strictly for testing and validation purposes. When a Link is in loopback, the symbols received are "looped back" to the transmitter on the same Lane.

A Loopback master is the component requesting loopback.

A Loopback slave is the component looping back the data.

Loopback is entered whenever two consecutive TS1 or TS2 ordered-sets are received with the loopback bit set.

Loopback is exited by the sending of an Electrical Idle ordered-set followed by Electrical Idle.

4.2.5.10. Disabled

In Disabled the receiver terminators must remain enabled and the transmitter is in a high impedance Electrical Idle.

The Receiver Detection sequence (see Section 4.3.1.8) is allowed while in the disabled state if desired.

Disabled is entered when directed by the Data Link Layer.

4.2.5.11. Link Control Reset

Link Control Reset is entered when directed or when two consecutive TS1 or TS2 ordered-sets are received with the Reset bit set.

---

[16] The exception is the receiver termination, which must remain in a low impedance state.

[17] In this context, "aux" power means a power source which can be used to drive the Beacon and Receiver Detection circuitry.

[18] A device generates beacons in order to wake a system that is in D3cold. See Section 4.3.2.4 for information on the electrical requirements of the beacon. Refer to Chapter 6 for more information on how a device may use the beacon as the wake mechanism.

4.2.6. Link Training and Status State Descriptions

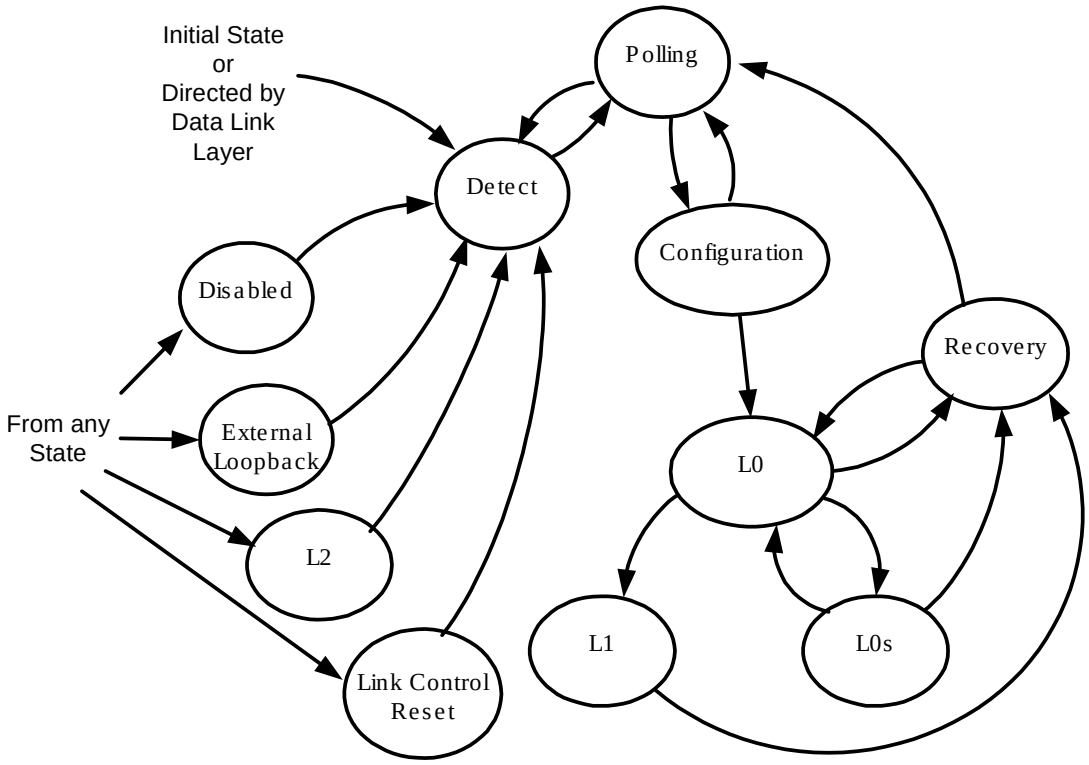

Figure 4-19: Main State Diagram for Link Training and Status State Machine

4.2.6.1. Detect

4.2.6.1.1. Detect.Quiet

- Transmitter is in a high impedance Electrical Idle state.
- Lane number and Link Number are initialized to K23.7.
- Generation 1 data rate is selected.
- LinkUp = 0 (status is cleared).
- Next state is Detect.Active after a 64 ms timeout.

4.2.6.1.2. Detect.Active

- The transmitter performs a high impedance Receiver Detection sequence(see Section 4.3.1.8 for more information).
- Next state is Detect.Charge if a receiver is detected.
- Next state is Detect.Quiet if a receiver is not detected.

4.2.6.1.3. Detect.Charge

- Transmitter is in a high impedance Electrical Idle state.
- Next state is Polling after 64 ms timeout or when the operating DC common mode voltage is stable and within specification.[19]

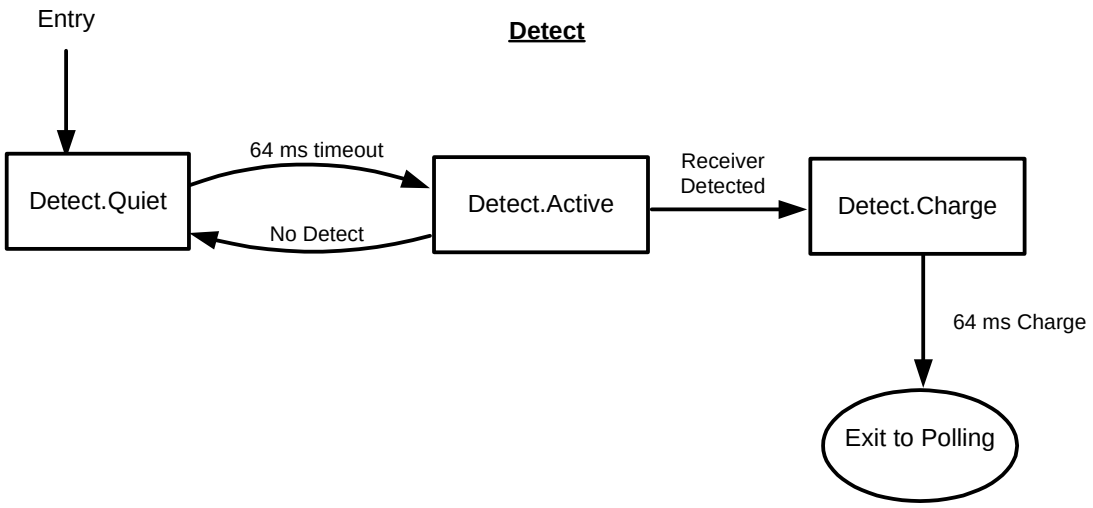

Figure 4-20: Detect Sub-State Machine

[19] The common mode being driven must meet the Absolute Delta Between DC Common Mode During L0 and Electrical Idle ($V_{TX\text{-}CM\text{-}DC\text{-}ACTIVE\text{-}IDLE\text{-}DELTA}$) specification (see Table 4-4).

4.2.6.2. Polling

4.2.6.2.1. Polling.Quiet

- Transmitter is in Electrical Idle.
- A Receiver Detection sequence (see Section 4.3.1.8 for more information) is performed.
    - If no receiver is present, next state is Detect
- LinkUp = 0 (status is cleared).
- Next state is Polling.Configuration if a single TS1 or TS2 ordered-set or their complement is received.
- Next state is Polling.Active after a minimum of 64 ms.

4.2.6.2.2. Polling.Active

- Transmitter sends a minimum of 1024 consecutive TS1 ordered-sets on all Lanes.
    - Note: This guarantees a minimum of 64 µs for the bit lock time at generation 1 data rates.
- Next state is Polling.Configuration if a single TS1 or TS2 ordered-set or their complement is received.
- Next state is Polling.Compliance if the transmitter has entered Polling.Active 32 consecutive times without receiving a single TS1 or TS2 ordered set and the receiver has never detected an exit from Electrical Idle after the first time entering Polling.
    - Note: The compliance mode is entered only if no signal was detected at any receiver on a Link since the time of reset.
- Next state is Polling.Quiet if the transmitter sends 1024 TS1 ordered-sets without receiving a single TS1 or TS2 ordered-set.

4.2.6.2.3. Polling.Compliance

- 8b/10b encoder is set to positive disparity
- Transmitter sends out the compliance pattern (see Section 4.2.8)
- Next state is Polling.Active if Electrical Idle is no longer detected at the receiver.

185

4.2.6.2.4. Polling.Configuration

- Receiver inverts polarity if necessary (see Section 4.2.4.2).
- Transmitter sends TS1 ordered-sets on the Port. At least 16 TS1 ordered-sets are sent after receiving one TS1 or TS2 ordered-set.
- Next state is Configuration if eight consecutive TS1 or TS2 ordered-sets are received and no higher data rate is supported
    - Otherwise, next state is Polling.Speed if eight consecutive TS1 or TS2 ordered-sets are received.
- Otherwise, next state is Polling.Active after a 2 ms timeout.

4.2.6.2.5. Polling.Speed

- The transmitter enters Electrical Idle for a minimum of $T_{TX\text{-}IDLE\text{-}MIN}$ (see Table 4-4).
- Data rate is changed to highest common data rate supported in the training sequence (see Section 4.2.4.1).
- Transmitter sends a minimum of 1024 consecutive TS1 ordered-sets on all lanes.
    - Note: This guarantees a minimum bit lock time.
- Next state is Configuration.

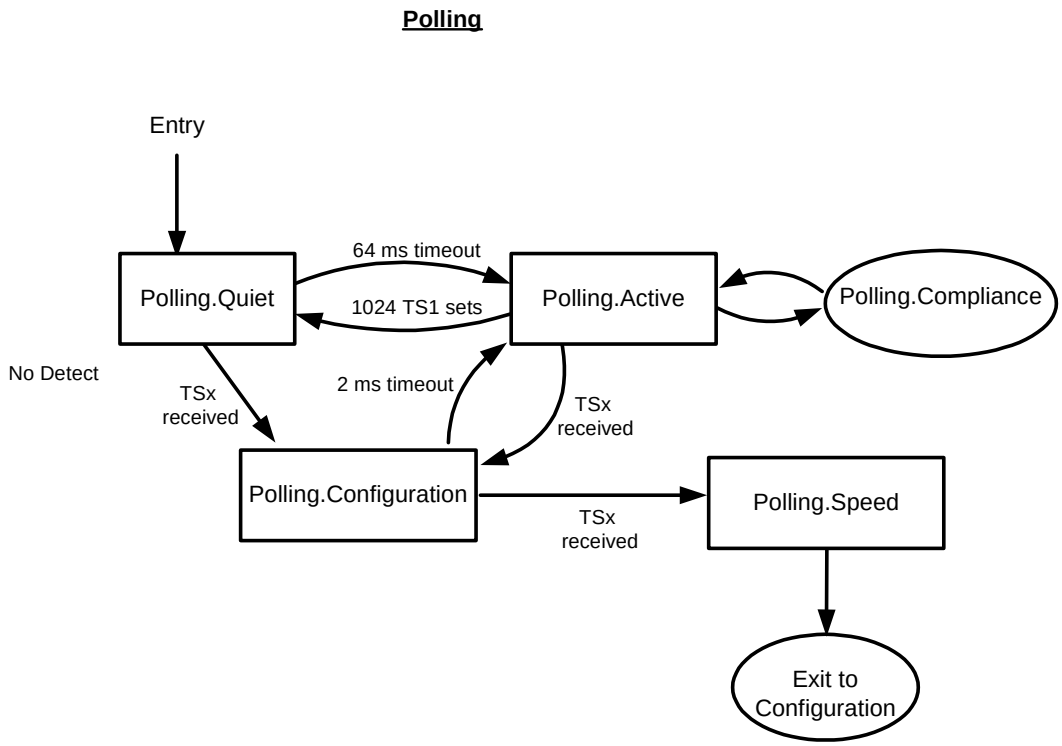

Figure 4-21: Polling Sub-State Machine

4.2.6.3. Configuration

4.2.6.3.1. Config.RcvrCfg

- Transmitter sends TS1 ordered-sets on the Lanes. At least 16 TS1 ordered-sets are sent after receiving one TS1 or TS2 ordered-set.
    - Note: All lanes must have achieved bit and symbol lock by this state as ensured by Polling.
- Link width and Lane reversal is performed as described in Section 4.2.4.8.
- Note: If some Lanes do not configure successfully they may be disabled or may be returned to Polling.
    - Note: Disabled Lanes should be re-enabled if any active Lanes within the same Link enter Detect.
    - Note: All Lanes on a configured Link must operate at the same data rate.
- Next state is Config.Idle if a receiver negotiates a valid configuration and receives eight consecutive TS1 or TS2 ordered-sets on all configured Lanes.

Otherwise, the data rate that the Port indicates it supports is dropped down to the next lower data rate and the next state is Polling. See Section 4.2.4.7 for information on data rate negotiation.

4.2.6.3.2. Config.Idle

- Transmitter sends Idle data symbols on all configured Lanes. At least 16 idle data symbols are sent after receiving one Idle data symbol.
- Receiver waits for Idle data.
- LinkUp = 1 (status is set true).
- Next state is L0 if eight consecutive symbol times of Idle data received on all configured Lanes.
- Otherwise, after a minimum 2 ms timeout the data rate that the Port indicates it supports is dropped down to the next lower data rate and the next state is Polling. See Section 4.2.4.7 for information on data rate negotiation.

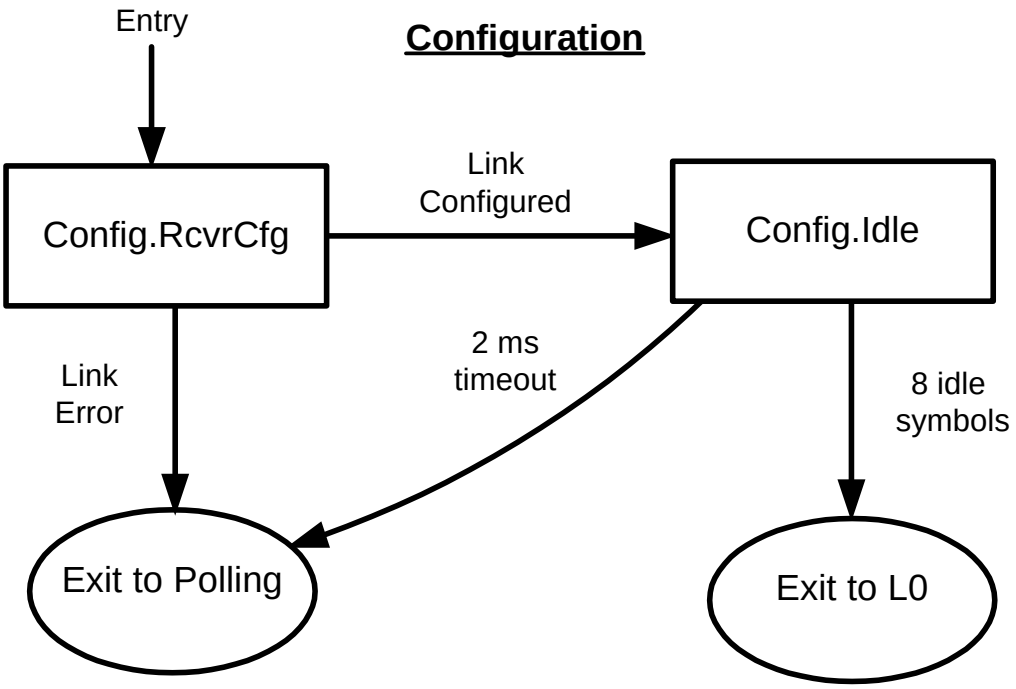

Figure 4-22: Configuration Sub-State Machine

4.2.6.4. Recovery

4.2.6.4.1. Recovery.RcvrCfg

- Transmitter sends TS2 ordered-sets on all configured Lanes. At least 16 TS2 ordered-sets are sent after receiving one TS2 ordered-set.
- Next state is Recovery.Idle if 8 consecutive TS2 ordered-sets are received on all configured Lanes.
- Otherwise, after 2 ms an error is reported to the Data Link Layer and the next state is Polling.

4.2.6.4.2. Recovery.Idle

- Transmitter sends Idle data (minimum of 16 symbol times) on configured Lanes.
- Receiver waits for Idle data.
- Next state is L0 if eight consecutive symbol times of Idle data received on all configured Lanes
- Otherwise, after 2 ms, an error is reported to the Data Link Layer and the next state is Polling.

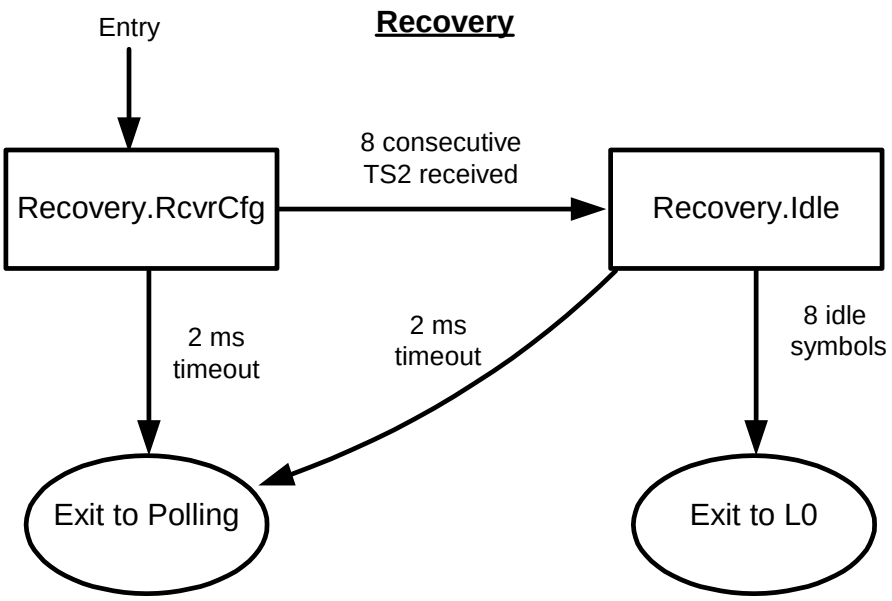

Figure 4-23: Recovery Sub-State Machine

4.2.6.5. L0

This is the normal operational state.

- Transmitter and receiver are enabled in a low impedance state.
- Next state is Recovery if TS1 or TS2 received.
- Next state is Recovery if directed to this state.
- Next state is Polling if directed to this state.
- Next state is Detect if directed to this state.
- Next state is L0s if receiver detects Electrical Idle ordered-set.
- Next state of transmitter is L0s if directed to this state.
- Next state is L1 if receiver detects Electrical Idle ordered-set and is directed to this state.
- Next state is L2 if receiver detects Electrical Idle ordered-set and is directed to this state.
- Next state is Link Control Reset if directed to this state.
- Next state is the Disabled if directed to this state.
- Next state is External Loopback if directed to this state.

4.2.6.6. L0s

4.2.6.6.1. Receiver L0s

4.2.6.6.1.1. Rx_L0s.Entry

- Next state is Rx_L0s.Idle after a $T_{TX\text{-}IDLE\text{-}MIN}$ (Table 4-4) timeout

4.2.6.6.1.2. Rx_L0s.Idle

- Next state is Rx_L0s.FTS if receiver detects an exit from Electrical Idle

4.2.6.6.1.3. Rx_L0s.FTS

- Receiver locks to incoming bit stream and acquires symbol alignment.
- Next state is Recovery if the receiver does not detect bit and symbol alignment within the N_FTS duration on all Lanes of the Link.
- Otherwise, if bit and symbol lock is obtained the next state is L0.

4.2.6.6.2. Transmitter L0s

4.2.6.6.2.1. Tx_L0s.Entry

- Transmitter is in Electrical Idle.
- Next state is Tx_L0s.Idle after a $T_{TX\text{-}IDLE\text{-}MIN}$ (Table 4-4) timeout.

4.2.6.6.2.2. Tx_L0s.Idle

- Next state is Tx_L0s.FTS if directed.

4.2.6.6.2.3. Tx_L0s.FTS

- Transmitter sends N_FTS Fast Training Sequences.
- Transmitter sends a single SKP ordered set on all configured Lanes.
- Next state is L0.

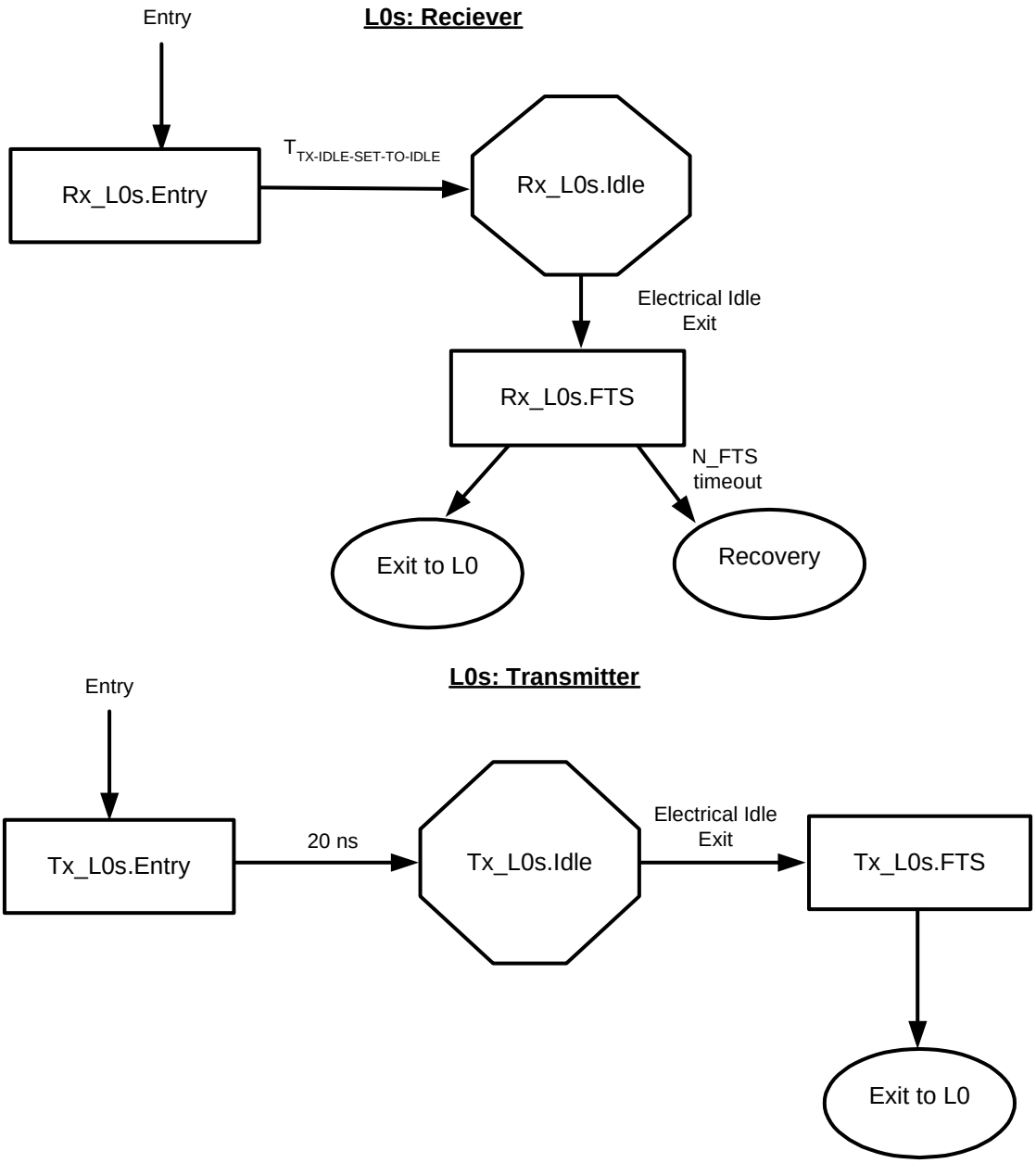

Figure 4-24: L0s Sub-State Machine

4.2.6.7. L1

4.2.6.7.1. L1.Entry

- Transmitter is in Electrical Idle.
- Receiver waits for at least Electrical Idle $T_{TX\text{-}IDLE\text{-}SET\text{-}TO\text{-}IDLE}$ time given in Table 4-4.
- The next state is L1.Idle after a $T_{TX\text{-}IDLE\text{-}MIN}$ (Table 4-4) timeout.
- Next state is L1.Quiet.

4.2.6.7.2. L1.Idle

- Transmitter is in Electrical Idle.
- Next state is Recovery if directed or if the receiver detects exit from Electrical Idle.

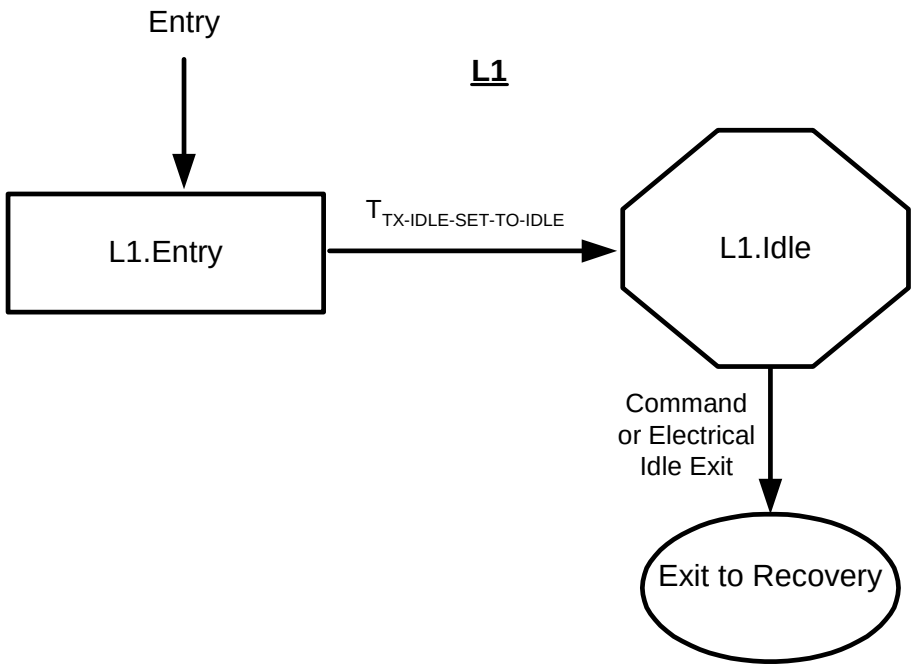

Figure 4-25: L1 Sub-State Machine

4.2.6.8. L2

4.2.6.8.1. L2.Idle

- Transmitter is in a high impedance Electrical Idle state for a minimum of 64 ms.
- Next state is Polling if detection of a Beacon occurs on Lane 0.
- Next state is L2.Detect if directed to transmit a Beacon.

4.2.6.8.2. L2.Detect

- Transmitter is in a high impedance Electrical Idle state for a minimum of 64 ms. .
- A high impedance Receiver Detection sequence is performed (see Section 4.3.1.8 for more information)
- Next state is L2.TransmitWake if a receiver is detected.
- Next state is Detect if a receiver is not detected.

4.2.6.8.3. L2.TransmitWake

- Transmitter is in a high impedance Electrical Idle state for a minimum of 64 ms.
- The transmitter transmits the Beacon on at least Lane 0 of the Link (Refer to Section 4.3.2.4).
- Next state is Polling if Electrical Idle is exited on any incoming receiver Lane.

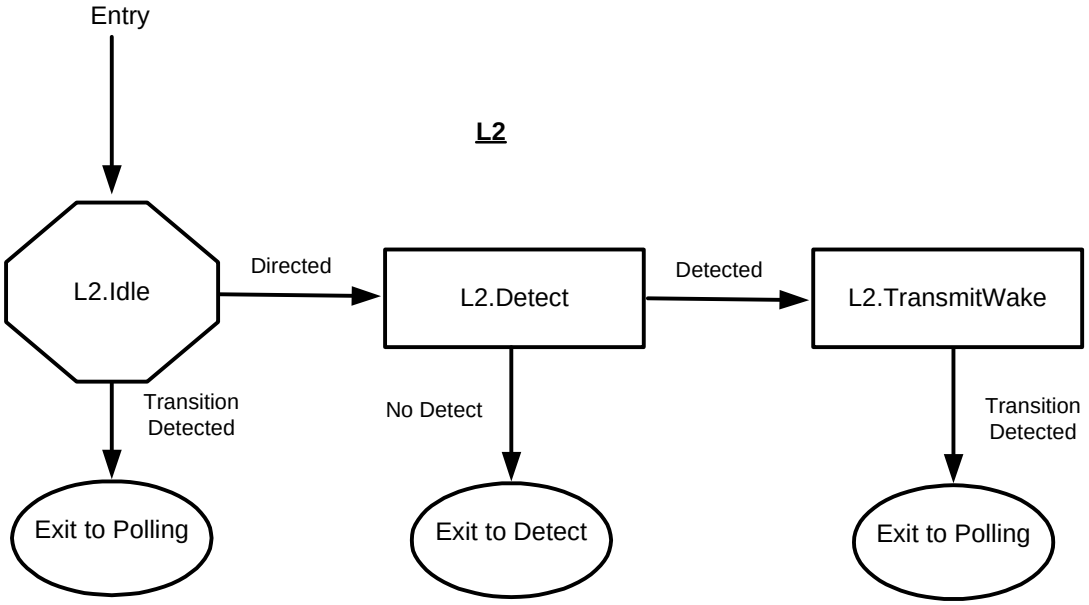

Figure 4-26: L2 Sub-State Machine

4.2.6.9. Disabled

- Entrance to and exit from this state only when directed.
- Transmitter sends between 4 and 16 TS1 ordered-sets with the Disable bit set.
- Transmitter then goes into a high impedance Electrical Idle state.
- Next state is Detect when directed.

4.2.6.10. Loopback

This mode is intended for test and fault isolation use only, and is not a normal operational mode. Only the entry and exit behavior is specified. All other details are implementation specific.

4.2.6.10.1. Loopback.Active

- The Loopback Slave must receive valid 8b/10b data. If SKP ordered-sets are received they are also looped back to the Loopback Master. SKP symbols may be added or removed by the Loopback Slave as needed.
- The Loopback Slave re-transmitter is sending the 10 bit data as received. If the received data was not 8b/10b valid, the transmitter sends back the special symbol EDB control character in place of the invalid character.
    - Note: The Loopback Slave must transmit the data with the same disparity as was received.
- Next state is Loopback.Exit when an Electrical Idle ordered-set is received by the Loopback Slave after the Electical.Idle ordered-set is transmitted back to the Loopback Master.
- ·Next state is Loopback.Exit if an Electrical Idle is detected continuously for a minimum of 1 UI at the Loopback Slave receiver.

4.2.6.10.2. Loopback.Exit

- Transmitter is in Electrical Idle
- Next state is Detect

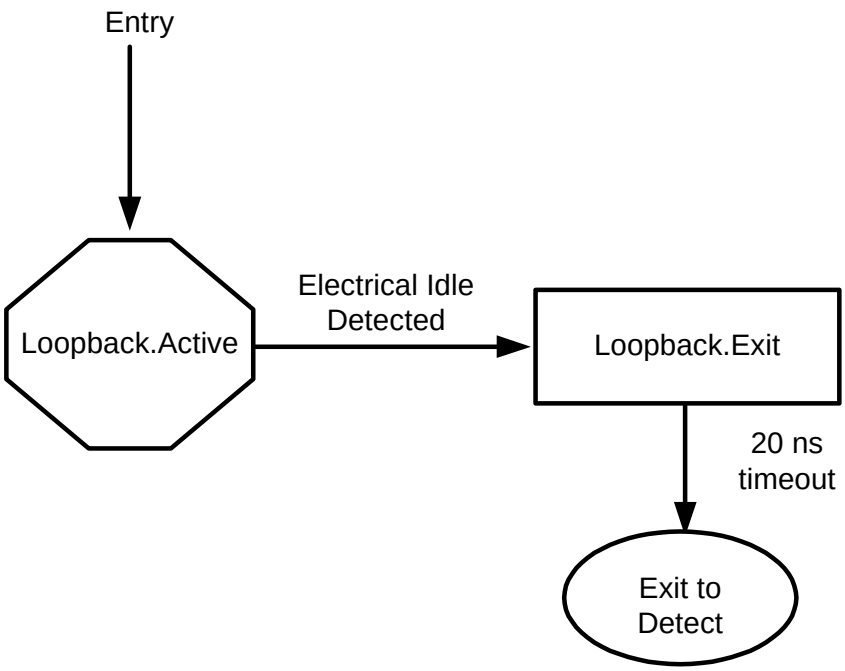

Figure 4-27: Loopback State Machine

4.2.6.11. Link Control Reset

4.2.6.11.1. Link Control Reset Active

- Link enters reset state and transmits a minimum of 1024 TS1 ordered-sets with the reset bit set on all downstream ports.
- All transmitters on upstream ports transmit one Electrical Idle ordered-set, then enter Electrical Idle.
- Next state is Polling.

4.2.7. Clock Tolerance Compensation

Skip 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 skip ordered-set transmissions is derived from the absolute value of the Transmit and Receive clock frequency difference specified in Table 4-4. Having worse case clock frequencies at the limits of the tolerance specified will result in a 600 ppm difference between the transmit and receive clocks of a Link. As a result, the transmit and receive clocks can shift one clock every 1666 clocks.

Rules for Transmitters

- All Lanes shall transmit Symbols at the same frequency (the difference between bit rates is 0 ppm within all multi-Lane links).
- When transmitted, the skip ordered-set shall be transmitted simultaneously on all Lanes of a multi-Lane Link (See Section 4.2.4.9 and Table 4-4 for the definition of simultaneous in this context).
- The transmitted skip ordered-set is: one COM Symbol followed by three consecutive SKP Symbols
- The skip ordered-set shall be scheduled for insertion at an interval between 1180 and 1538 Symbol Times.
- Scheduled SKIP 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.

Rules for Receivers

- Receivers shall recognize received skip ordered-set consisting of one COM Symbol followed consecutively by one to five SKP Symbols.
- Receivers shall be tolerant to receive and process SKIP ordered-sets at an average interval between 1180 to 1538 symbol times.
- Receivers shall be tolerant to receive and process consecutive SKIP ordered-sets.
    - Receivers shall be tolerant to receive and process SKIP ordered-sets separated from each other at most 5664 symbol times – measured as the distance between the leading COM symbols.

4.2.8. Compliance Pattern

During polling the compliance substate of the polling state machine may be entered (see Section 4.2.5.3). The compliance pattern consists of the sequence of 8b/10b symbols K28.5, D21.5, K28.5, and D10.2 repeating. Current running disparity must be set to negative before sending the first symbol.

The compliance pattern is not entered if the receiver has previously detected an exit from Electrical Idle.

The compliance sequence is:

Symbol	K28.5	D21.5	K28.5	D10.2
Current Disparity	0	1	1	0
Pattern	0011111010	1010101010	1100000101	0101010101

For any given device that has multiple lanes, every fourth Lane is delayed by a total of 4 symbols. A 2 symbol delay occurs at both the beginning and end of the 4 symbol sequence, for a total of 8 symbols.

This delay sequence on every fourth Lane is then:

Symbol:	D	D	K28.5	D21.5	K28.5	D10.2	D	D

Where D is 2 symbols that are the same such that disparity is preserved after sending the 2 D symbols. Example D symbols are the K28.5 and the D10.2.

After the 8 symbols are sent, the delay symbols are advanced to the next Lane and the process is repeated. This looks like:

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	D	D	K28.5-	D21.5	K28.5+	D10.2	D	D	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	D	D	K28.5-	D21.5

Key:

K28.5-	Comma when disparity is negative, specifically: "0011111010"
K28.5+	Comma when disparity is positive, specifically: "1100000101"
D21.5	Out of phase data character, specifically: "1010101010"
D10.2	Out of phase data character, specifically: "0101010101"
D	Delay Character

This sequence of delays ensures that the maximum possible interference effects of adjacent lanes occur for when measuring the compliance pattern.

The compliance pattern is only exited if an Electrical Idle Exit condition is detected at the receiver or if a physical reset occurs.

4.3. Electrical Sub-Block

The Electrical sub-block contains Transmitter and a Receiver. The Transmitter is supplied by the Logical sub-block with Symbols which it serializes and transmits onto a Lane. The Receiver is supplied with serialized Symbols from the Lane. It transforms the electrical signals into a bit stream which is de-serialized and supplied to the Logical sub-block along with a Link clock recovered from the incoming serial stream.

4.3.1. Electrical Sub-Block Requirements

4.3.1.1. Clocking Dependencies

The ports on the two ends of a Link must transmit data at a rate that is within 600 parts per million (ppm) of each other at all times. This is specified to allow bit rate clock sources with a +/- 300 ppm tolerance.

4.3.1.1.1. Spread Spectrum Clock (SSC) Sources

The data rate can be modulated from +0% to -0.5% of the nominal data rate frequency, at a modulation rate in the range not exceeding 30 kHz – 33 kHz. The +/- 300 ppm requirement still holds, which requires the two communicating ports be modulated such that they never exceed a total of 600 ppm difference. For most implementations this places the requirement that both ports require the same bit rate clock source when the data is modulated with an SSC.

4.3.1.2. AC Coupling

Each Lane of a Link must be AC coupled. The minimum and maximum value for the capacitance is given in Table 4-4. The requirement for the inclusion of AC coupling capacitors on the interconnect media is associated with the transmitter.

4.3.1.3. Interconnect

In the context of this spec, the interconnect consists of everything between the pins at a transmitter package and the pins of a receiver package. Often, this will consist of traces on a printed circuit board of other suitable medium, AC coupling capacitors and perhaps connectors. Regardless of what physically makes up the interconnect, the total capacitance of the interconnect seen by the receiver detection circuit (see Section 4.3.1.8) may not exceed 3 nF.

4.3.1.4. Termination

Low and high impedance states are defined for both the transmitter and the receiver and are listed in Table 4-4 and Table 4-5.

The only time the transmitter high impedance state is required is to initialize and maintain Electrical Idle during times when a hot plug/removal or asynchronous power up event could occur.[20]

The transmitter low impedance state is required any time differential data is to be sent.

The only time the receiver must be in a high impedance state is when the receiver does not have power. Otherwise, the receiver must always be in a low impedance state.

4.3.1.5. DC Common Mode

The receiver DC common mode is always 0 V during all states.

The transmitter DC common mode is initially established during Detect and is held at the same value during all subsequent states.

4.3.1.6. ESD

All signal and power pins must withstand (2000 V) of ESD using the human body model and 800 V using the charged device model without damage. Class 2 per JEDEC JESE22-A114-A.

This ESD protection mechanism also protects the powered down receiver from potential common mode transients during some possible reset or surprise insertion situations.

4.3.1.7. Short Circuit Requirements

All Transmitters and Receivers must support surprise hot insertion/removal without damage to the component. The transmitter and receiver must be capable of withstanding sustained short circuit to ground of D+ and D-.

4.3.1.8. Receiver Detection

The receiver detection sequence is used to avoid unwanted common mode transfers between the receiver and transmitter.

The receiver detection can be performed in either a low or high impedance state unless explicitly specified.

---

[20] Any time high impedance is required it is explicitly stated in the Link Training and Status State Machine (LTSSM).

The behavior of the receiver detection sequence is described below.

Step 1. Transmitter is in a stable Electrical Idle state.

Step 2. The transmitter changes the common mode voltage on both D+ and D- lines[21] to a different value.

a. A receiver is detected based on the rate[22] that the lines change to the new voltage.

i. The receiver is not present if the voltage at the transmitter charges at a rate dictated by the transmitter impedance and capacitance of the interconnect.

ii. 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.

The AC capacitance of the worst-case transmission line must not exceed 3 nF total. The minimum and maximum AC capacitance for the AC coupling capacitors is given in Table 4-4.

4.3.1.9. Disable/Surprise Removal Detection

Two separate events may signal that one end of a Link has either been disabled or disconnected.

1. During L0 if Electrical Idle is detected without receiving the Electrical Idle ordered-set. The Link immediately enters Detect.
2. During Electrical Idle and certain specified times the transmitter must poll for the presence of a powered receiver as described in Section 4.3.1.8. If a receiver is no longer present, the Link immediately enters Detect.

4.3.1.10. Electrical Idle

Electrical idle is a steady state condition where the Transmitter and Receive voltages are held constant. Electrical idle is primarily used in power saving and common mode initialization.

Before a transmitter enters Electrical Idle, it must send the Electrical Idle ordered-set, a K28.5 (COM) followed by three K28.3 (IDL)(see Table 4-4). After sending the last symbol of the Electrical Idle ordered-set the transmitter must be in a valid Electrical Idle state as specified by $T_{TX\text{-}IDLE\text{-}SET\text{-}TO\text{-}IDLE}$ (see Table 4-4). The receiver shall use this ordered-set to enter electrical idle.

The Receiver terminations must remain enabled in Electrical Idle. The transmitter must meet the DC common mode specification while transitioning into and out of Electrical Idle, which can be done in a low or high impedance state unless specifically specified.

---

[21] The maximum change in common mode voltage can be no more than $V_{TX\text{-}CM\text{-}RCV\text{-}DETECT}$ in Table 4-4.

[22] The rate of change should be at least 40x different between a receiver present and not present.

Any time a transmitter enters Electrical Idle it must remain in electrical idle for a minimum of $T_{TX\text{-}IDLE\text{-}MIN}$(see Table 4-4). The receiver should expect the Electrical Idle ordered-set followed by a minimum amount of time in Electrical Idle ($T_{TX\text{-}IDLE\text{-}SET\text{-}TO\text{-}IDLE}$) to arm its Electrical Idle Exit detector.

See Section 4.3.1.9 for additional notes related to Electrical Idle.

4.3.2. Electrical Signal Specifications

A Differential Signal is defined by taking the voltage difference between two conductors. In this specification, a differential signal or differential pair is comprised of a voltage on a positive conductor, V_{D+}, and a negative conductor, V_{D-}. The differential voltage (V_{DIFF}) is defined as the difference of the positive conductor voltage and the negative conductor voltage ($V_{DIFF} = V_{D+} - V_{D-}$). The Common Mode Voltage (V_{CM}) is defined as the average or mean voltage present on the same differential pair ($V_{CM} = [V_{D+} + V_{D-}]/2$). This document's electrical specifications often refer to peak-to-peak measurements or peak measurements, which are defined by the following equations.

- $V_{DIFFp\text{-}p} = (2*\max|V_{D+} - V_{D-}|)$ (This applies to a symmetric differential swing)
- $V_{DIFFp\text{-}p} = (\max|V_{D+} - V_{D-}|\ \{V_{D+} > V_{D-}\} + \max|V_{D+} - V_{D-}|\ \{V_{D+} < V_{D-}\})$ (This applies to an asymmetric differential swing.)
- $V_{DIFFp} = (\max|V_{D+} - V_{D-}|)$ (This applies to a symmetric differential swing)
- $V_{DIFFp\text{-}p} = (\max|V_{D+} - V_{D-}|\ \{V_{D+} > V_{D-}\})$ or $(\max|V_{D+} - V_{D-}|\ \{V_{D+} < V_{D-}\})$ which ever is greater (This applies to an asymmetric differential swing.)
- $V_{CMp} = (\max|V_{D+} + V_{D-}|/2)$

Note: The maximum value is calculated on a per unit interval evaluation. The maximum function as described is implicit for all peak-to-peak and peak equations throughout the rest of this chapter, and thus a max function will not appear in any following representations of these equations.

In this section, DC is defined as all frequency components below F_{dc} = 30 kHz. AC is defined as all frequency components above F_{dc} = 30 kHz. These definitions pertain to all voltage and current specifications.

An example waveform is shown in Figure 4-28. In this waveform the differential peak-peak signal is approximately 0.6 V, the differential peak signal is approximately 0.3 V and the common mode is approximately 0.25 V.

201

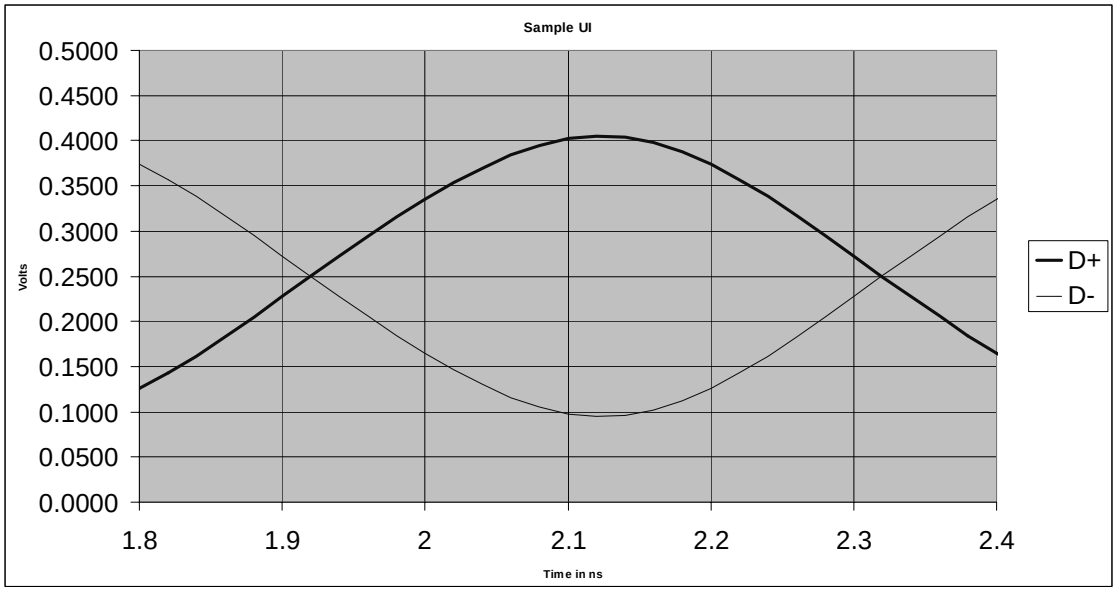

Figure 4-28: Sample Differential Signal

4.3.2.1. Loss

Loss (attenuation of the differential voltage swing) in this system is a critical parameter that must be properly considered and managed in order to ensure proper system functioning. Failure to do so may result in a differential signal swing arriving at the Receiver that does not meet specifications. The interconnect loss is specified in terms of the amount of attenuation or loss it can tolerate between the Transmitter (Tx) and Receiver (Rx). The Tx is responsible for producing the specified differential eye height at the pins of its package. Together, the Tx and the interconnect are responsible for producing the specified differential eye height at the Rx pins (see Figure 4-34).

The worst-case operational loss budget is calculated by taking the minimum output voltage ($V_{TX\text{-}DIFFp\text{-}p}$ = 800 mV) divided by the minimum input voltage to the receiver ($V_{RX\text{-}DIFFp\text{-}p}$ = 175 mV), which results in 13.2 dB. Additional headroom in loss budget can be achieved by driving a larger differential output voltage at the transmitter.

4.3.2.2. Jitter

The jitter budget is derived assuming a maximum bit error rate (BER) of 10^{-12}. The allocation anticipates both data dependent and random jitter contributions. The total jitter budget is the sum of the deterministic jitter and 14 times the standard deviation RMS value of the random jitter distribution. Total jitter is the combined peak-to-peak measured jitter from all sources.

4.3.2.3. De-emphasis

De-emphasis is included to minimize Inter-symbol interference (ISI) due to delta in loss versus the primary fundamental transient frequencies (i.e., Generation 1 fundamental band = 250 MHz to 1.25 GHz).

De-emphasis must be implemented when multiple bits of the same polarity are output in succession. Subsequent bits are driven at a differential voltage level 3.5 dB (+/-.5 dB) below the first bit. Individual bits are always driven at the full voltage level.

The only exception pertains to transmitting the Beacon (see Section 4.3.2.4).

Note: The specified amount of de-emphasis was chosen to optimize maximum inter-operability while minimizing complexity of managing configurable de-emphasis values. Thus, the de-emphasis was targeted to work for the worst-case loss budget of 11-13.2 dB, which tends to make it less optimal for the lower loss systems. The fact that is less optimal for lower loss systems is more than offset by the fact that there is inherently more voltage margin in lower loss systems.

4.3.2.3.1. De-emphasis Example

An example waveform representing the 10-bit symbol 243H is shown in Figure 4-29.

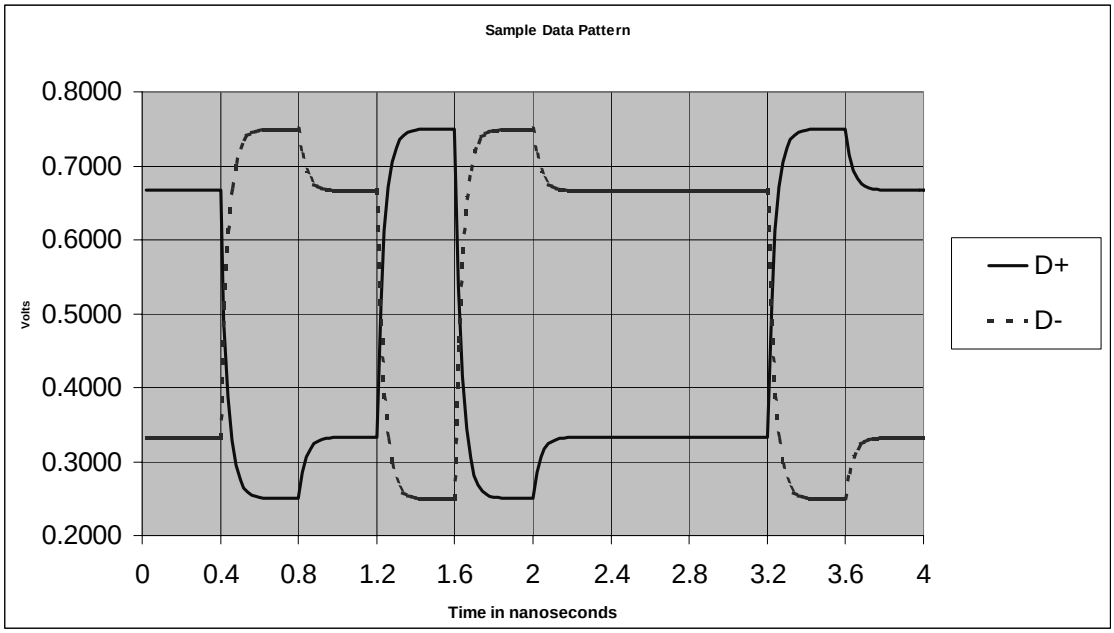

Figure 4-29: Sample Transmitted Waveform Showing -3.5 dB De-emphasis Around a 0.5 V Common Mode

4.3.2.4. Beacon

All transmitter electrical specifications must be met while sending a Beacon with the following exceptions and clarifications.

- The period of the Beacon must be no greater than 33.333 µs maximum.
- All Beacons must be transmitted on at least Lane 0 of multi-lane links[23].
- The Beacon signal must contain pulses that are 2 ns minimum.
- The Beacon must be DC Balanced (i.e., any Beacon must contain an equal number of 1's and 0's).
- The output Beacon voltage level must be at a -6 dB de-emphasis level for Beacon pulses with a width greater than 500 ns.
- The output Beacon voltage level can range between the pre-emphasized and corresponding -3.5 dB de-emphasized voltage levels for Beacon pulses smaller than 500 ns.
- The output Beacon voltage level must be at the de-emphasis level for Beacon pulses with a width greater than 500 ns. Otherwise, the Beacon output voltage can range between the pre-emphasized and corresponding de-emphasized voltage levels.
- A Receiver Detection sequence (Section 4.3.1.8) must occur every 100 ms, and if no receiver is found then the Link returns to Detect.
- The Lane-to-Lane Output Skew and Skip Symbol Output specifications do not apply.

When any bridge and/or switch receives a Beacon, that component must propagate a Beacon upstream.

4.3.2.4.1. Beacon Example

An example receiver waveform driven at the -6 dB level for a 30 kHz Beacon is shown in Figure 4-30. An example receiver waveform using the COM character at full speed signaling is shown in Figure 4-31 It should be noted that other waveforms and signaling are possible other than the two examples shown below (i.e., Polling is another valid Beacon signal).

[23] Lane 0 as defined after Link Width and Lane reversal negotiations are complete.

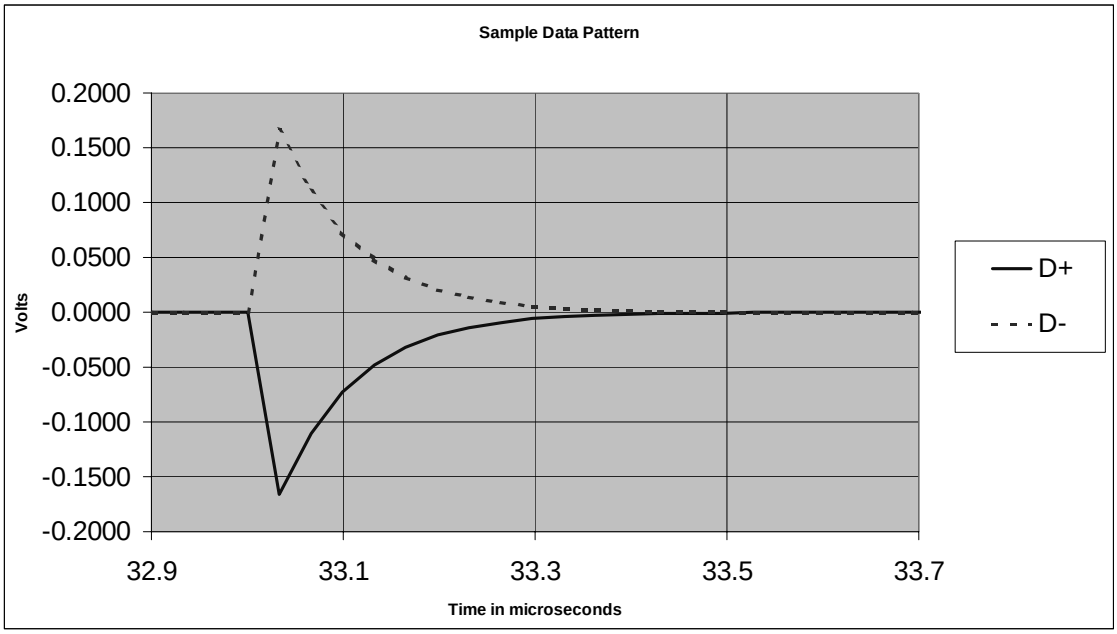

Figure 4-30: A 30 kHz BEACON Signaling Through a 75 nF Capacitor

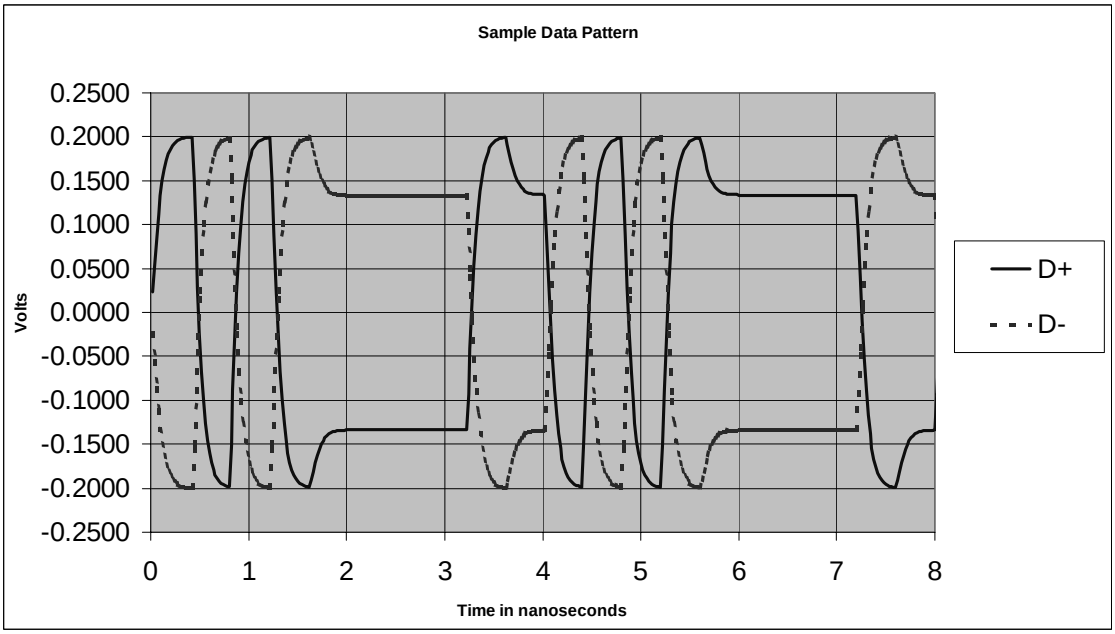

Figure 4-31: BEACON, Which Includes a 2 ns Pulse Through a 75 nF Capacitor

4.3.3. Differential Transmitter (Tx) Output Specifications

The following table defines the specification of parameters for the differential output at all Transmitters (Txs). The parameters are specified at the component pins.

Table 4-4: Differential Transmitter (Tx) Output Specifications

Symbol	Parameter	Min	Nom	Max	Units	Comments
UI	Unit Interval	399.88	400	400.12	ps	Each UI is 400 ps +/-300 ppm. UI does not account for SSC dictated variations. See Note 1.
$V_{TX\text{-}DIFFp\text{-}p}$	Differential Peak to Peak Output Voltage	0.800		1.2	V	$V_{TX\text{-}DIFFp\text{-}p} = 2*\lvert V_{TX\text{-}D+} - V_{TX\text{-}D-}\rvert$ Measured at the package pins of the transmitter. See Note 2.
$V_{TX\text{-}DE\text{-}Ratio}$	De-Emphasized Differential Output Voltage (Ratio)	-3.0	-3.5	-4.0	dB	This is the ratio of the $V_{tx\text{-}Diffp\text{-}p}$ of the second and following bits after a transition divided by the $V_{tx\text{-}Diffp\text{-}p}$ of the first bit after a transition. See Note 2.
$T_{TX\text{-}EYE}$	Minimum TX Eye Width	0.70			UI	Minimum transmitter eye width measured in relation to a fixed UI at the package pins of the transmitter. The maximum transmitter jitter can be derived as $T_{TX\text{-}MAX\text{-}JITTER} = 1 - T_{TX\text{-}EYE} = .3$ UI See Notes 2 and 3.
$T_{TX\text{-}EYE\text{-}MEDIAN\text{-}to\text{-}MAX\text{-}JITTER}$	Maximum time between the jitter median and maximum deviation from the median.			< 0.15	UI	Jitter is defined as the measurement variation of the crossing points ($V_{TX\text{-}DIFFp\text{-}p.}$ = 0 V) in relation to an ideal TX UI. Median jitter and the maximum jitter deviation from the median are measured at the package pins of the transmitter. ees Note 2 and 3.
$T_{TX\text{-}RISE}$, $T_{TX\text{-}FALL}$	D+/D- TX Output Rise/Fall Time	0.125		0.4	UI	See Notes 2 and 5.
$V_{TX\text{-}CM\text{-}Acp}$	AC Peak Common Mode Output Voltage			20	mV	$V_{TX\text{-}CM\text{-}AC} = \lvert V_{TX\text{-}D+} + V_{TX\text{-}D-}\rvert/2 - V_{TX\text{-}CM\text{-}DC}$ $V_{TX\text{-}CM\text{-}DC} = DC_{(avg)}$ of $\lvert V_{TX\text{-}D+} + V_{TX\text{-}D-}\rvert/2$ during L0 See Note 2.

Symbol	Parameter	Min	Nom	Max	Units	Comments
$V_{TX\text{-}CM\text{-}DC\text{-}ACTIVE\text{-}IDLE\text{-}DELTA}$	Absolute Delta Between DC Common Mode During L0 and Electrical Idle.	0		100	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 $\lvert V_{TX\text{-}D+} + V_{TX\text{-}D-}\rvert/2$ [L0] $V_{TX\text{-}CM\text{-}Idle\text{-}DC} = DC_{(avg)}$ of $\lvert V_{TX\text{-}D+} + V_{TX\text{-}D-}\rvert/2$ [electrical idle] See Note 2.
$V_{TX\text{-}CM\text{-}DC\text{-}LINE\text{-}DELTA}$	Absolute Delta Between DC Common Mode between D+ and D-.	0		25	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] See Note 2.
$V_{TX\text{-}IDLE\text{-}DIFFp}$	Electrical Idle Differential Peak Output Voltage	0		20	mV	$V_{TX\text{-}Idle\text{-}DIFFp} = \lvert V_{TX\text{-}Idle\text{-}D+} - V_{Tx\text{-}Idle\text{-}D-}\rvert$ <= 20 mV See Note 2.
$V_{TX\text{-}CM\text{-}RCV\text{-}DETECT}$	The amount of common mode voltage change allowed during Receiver Detection.			600	mV	The total amount of common mode voltage change that a transmitter can apply to sense whether a low impedance receiver is present. See Section 4.3.1.8.
$T_{TX\text{-}IDLE\text{-}MIN}$	Minimum time spent in Electrical Idle	50			UI	Minimum time a transmitter must be in electrical idle after exiting.
$T_{TX\text{-}IDLE\text{-}SET\text{-}TO\text{-}IDLE}$	Maximum time to transition to a valid Electrical Idle after sending an Electrical Idle ordered-set			20	UI	After sending an electrical idle ordered-set, the transmitter must meet all electrical idle specifications within this time.
$T_{TX\text{-}IDLE\text{-}RCV\text{-}DETECT\text{-}MAX}$	Maximum time spent in Electrical Idle before initiating a receiver detect sequence.			100	ms	Maximum time spent in Electrical Idle before initiating a receiver detect sequence. See Section 4.3.1.8
$RL_{TX\text{-}DIFF}$	Differential Return Loss	12			dB	Measured over 50 MHz to 1.25 GHz See Note 4.

207

Symbol	Parameter	Min	Nom	Max	Units	Comments
RL_{TX-CM}	Common Mode Return Loss	6			dB	Measured over 50 MHz to 1.25 GHz See Note 4.
$Z_{TX-DIFF-DC}$	DC Differential TX Impedance	90	100	110	Ω	TX DC Differential Mode Low impedance
$Z_{TX-Match-DC}$	D+/D- TX Impedance Matching	-5		+5	Ω	TX DC impedance matching between D+ and D- on a given Lane.
$Z_{TX-COM-High-IMP-DC}$	Transmitter Common Mode High Impedance State (DC)	5 k		20 k	Ω	Tx DC High Impedance.
$L_{TX-SKEW}$	Lane-to-Lane Output Skew			500	ps	Between any two Lanes within a single Transmitter.
C_{TX}	AC Coupling Capacitor	75		500	nF	All transmitters shall be AC coupled to the media.

Notes:

1. No test load is necessarily associated with this value.
2. Specified at the package pins into a timing and voltage compliance test load as shown in Figure 4-33 and measured over at least 250 Tx UIs. (also refer to the Transmitter Compliance Eye Diagram as shown in Figure 4-32).
3. A T_{TX-EYE} = 0.70 UI provides for a total sum of deterministic and random jitter budget of $T_{TX-JITTER-MAX}$ = 0.30 UI for the transmitter collected over at least 250 TX UIs. The $T_{TX-EYE-MEDIAN-to-MAX-JITTER}$ specification ensures a jitter distribution in which the median and the maximum deviation from the median is less than half of the total TX jitter budget collected over at least 250 TX UIs. It should be noted that the median is not the same as the mean. The jitter median describes the point in time where the number of jitter points on either side is approximately equal as opposed to the averaged time value.
4. The transmitter input impedance shall result in a differential return loss greater than or equal to 12 dB and a common mode return loss greater than or equal to 6 dB over a frequency range of 50 MHz to 1.25 GHz. This input impedance requirement applies to all valid input levels. The reference impedance for return loss measurements for is 50 ohms to ground for both the D+ and D- line (i.e., as measured by a Vector Network Analyzer with 50 ohm probes - see Figure 4-33). Note: that the series capacitors C_{TX} is optional for the return loss measurement.
5. Measured between 20-80% at Transmitter package pins into a test load as shown in Figure 4-33 for both V_{TX-D+} and V_{TX-D-}. The maximum rise/fall time of the D+ and D- signals in Table 4-4 is considered relative bounds in that absolute boundaries for the maximum rise/fall time is dictated by the Transmitter Compliance Eye Diagram as shown in Figure 4-32.

4.3.3.1. Transmitter Compliance Eye Diagram

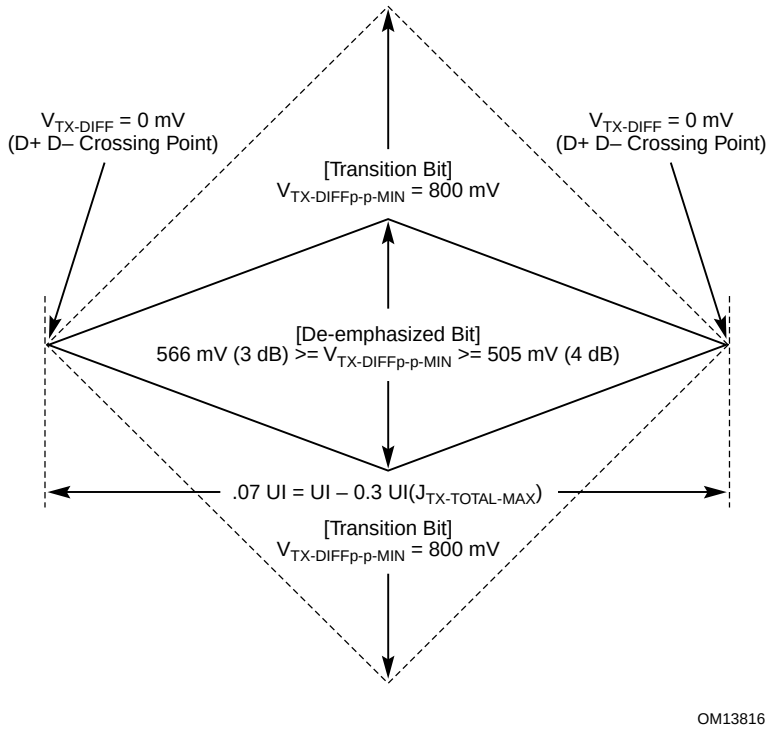

Figure 4-32: Minimum Transmitter Timing and Voltage Output Compliance Specification

There are two eye diagrams that must be met for the transmitter. Both eye diagrams must aligned in time and meet the minimum 0.7 UI requirement. The different eye diagrams will differ in voltage depending whether it is a transition bit or a de-emphasized bit. The eye diagram must be valid for at least 250 UIs. The Tx UI must be used as a trigger for the eye diagram.

4.3.3.2. Compliance Test and Measurement Load

The AC timing and voltage parameters should be verified at the package pins into a test/measurement load shown in Figure 4-33.

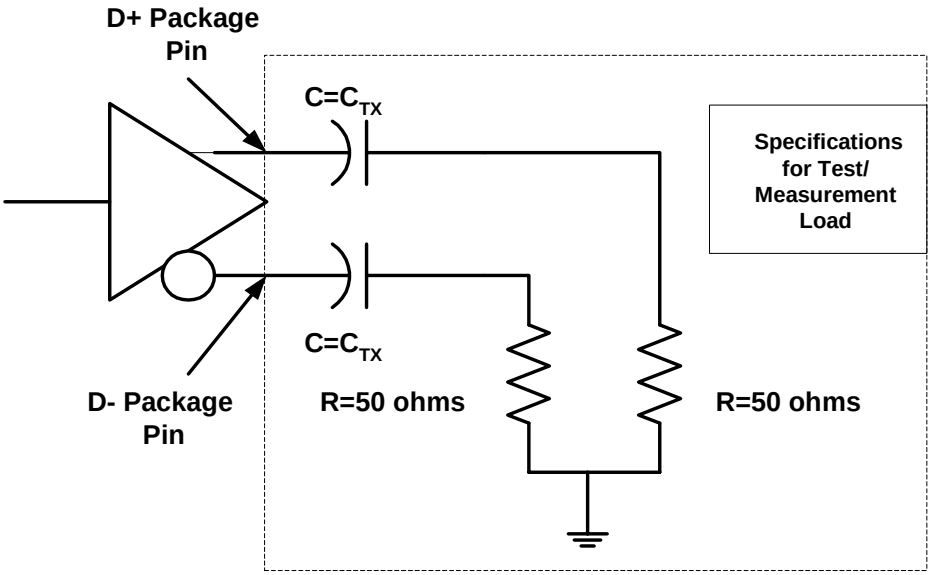

Figure 4-33: Compliance Test/Measurement Load

The test load is shown at the transmitter package reference plane, but the same Test/Measurement load is applicable to the receiver package reference plane.

Return Loss measurements do not require that C_{TX} be part of the measurement test load.

4.3.4. Differential Receiver (Rx) Input Specifications

The following table defines the specification of parameters for all differential Receivers (Rxs). The parameters are specified at the component pins.

Table 4-5: Differential Receiver (Rx) Input Specifications

Symbol	Parameter	Min	Nom	Max	Units	Comments
UI	Unit Interval	399.88	400	400.12	ps	The UI is 400 ps +/-300 ppm. UI does not account for SSC dictated variations. See Note 6.
$V_{RX\text{-}DIFFp\text{-}p}$	Differential Input Peak to Peak Voltage	0.175		1.200	V	$V_{RX\text{-}DIFFp\text{-}p} = 2*\lvert V_{RX\text{-}D+} - V_{RX\text{-}D-}\rvert$ Measured at the package pins of the Receiver. See Note 7.
$T_{RX\text{-}EYE}$	Minimum Receiver Eye Width	0.4			UI	Minimum receiver eye width measured in relation to a fixed UI at the package pins of the receiver. The maximum transmitter jitter can be derived as $T_{RX\text{-}MAX\text{-}JITTER} = 1 - T_{RX\text{-}EYE} = .6$ UI See Notes 7 and 8.
$T_{RX\text{-}EYE\text{-}MEDIAN\text{-}to\text{-}MAX\text{-}JITTER}$	Maximum time between the jitter median and maximum deviation from the median.			< 0.3	UI	Jitter is defined as the measurement variation of the crossing points ($V_{RX\text{-}DIFFp\text{-}p.}$ = 0 V) in relation to an ideal TX UI. Median jitter and the maximum jitter deviation from the median are measured at the package pins of the transmitter. See Notes 7 and 8.
$V_{RX\text{-}CM\text{-}ACp}$	AC Peak Common Mode Input Voltage			100	mV	$V_{RX\text{-}CM\text{-}AC} = \lvert V_{RX\text{-}D+} + V_{RX\text{-}D-}\rvert/2 - V_{RX\text{-}CM\text{-}DC}$ $V_{RX\text{-}CM\text{-}DC} = DC_{(avg)}$ of $\lvert V_{RX\text{-}D+} + V_{RX\text{-}D-}\rvert/2$ during L0 See Note 7,
$RL_{RX\text{-}DIFF}$	Differential Return Loss	15			dB	Measured over 50 MHz to 1.25 GHz See Note 9
$RL_{RX\text{-}CM}$	Common Mode Return Loss	6			dB	Measured over 50 MHz to 1.25 GHz See Note 9
$Z_{RX\text{-}DIFF\text{-}DC}$	DC Differential Input Impedance	90	100	110	Ω	RX DC Differential Mode impedance. See Note 10

Symbol	Parameter	Min	Nom	Max	Units	Comments
$Z_{RX\text{-}COM\text{-}DC}$	DC Input Common Mode Input Impedance	45	50	55	Ω	RX DC Common Mode impedance 50 Ω +/-10% tolerance. See Note 10. See Note 7.
$Z_{RX\text{-}Match\text{-}DC}$	Differential Pair Impedance Match	-5		+5	Ω	RX DC impedance matching between D+ and D- on a given Lane. See Note 10
$Z_{RX\text{-}COM\text{-}Inital\text{-}DC}$	Initial DC Input Common Mode Input Impedance	5	50	55	Ω	RX DC Common Mode impedance allowed when the receiver terminations are first power on. See Note 11.
$Z_{RX\text{-}COM\text{-}HIGH\text{-}IMP\text{-}DC}$	Powered Down DC Input Common Mode Input Impedance	200 k			Ω	RX DC Common Mode impedance when the receiver terminations are not powered (i.e. no power). See Note 12
$V_{RX\text{-}IDLE\text{-}DET\text{-}DIFFp\text{-}p}$	Electrical Idle Detect Threshold	65		175	mV	$V_{RX\text{-}IDLE\text{-}DET\text{-}DIFFp\text{-}p} = 2*\lvert V_{RX\text{-}D+} - V_{RX\text{-}D-}\rvert$ Measured at the package pins of the Receiver.
$T_{RX\text{-}IDLE\text{-}DET\text{-}DIFF\text{-}ENTERTIME}$	Unexpected Electrical Idle Enter Detect Threshold Integration Time			10	ms	$V_{RX\text{-}DIFFp\text{-}p} < V_{RX\text{-}IDLE\text{-}DET\text{-}DIFFp\text{-}p}$ must be recognized no longer than $V_{RX\text{-}IDLE\text{-}DET\text{-}DIFF\text{-}ENTERTIME}$ to signal an unexpected idle condition. See Note 13.
$L_{RX\text{-}SKEW}$	Total Skew			20	ns	Across all Lanes on a port. This includes variation in the length of a skip ordered-set (e.g., COM and 1 to 5 SKP symbols) at the rx as well as any delay differences arising from the interconnect itself.

Notes:

6. No test load is necessarily associated with this value.
7. Specified at the interface to the RX package pins and measured over at least 250 UIs. The test load in Figure 4-33 should be used as the RX device when taking measurements (also refer to the Receiver Compliance Eye Diagram as shown in Figure 4-34). If the clocks to the RX and TX are not derived from the same clock chip the TX UI must be used as a trigger for the eye diagram.
8. A $T_{RX\text{-}EYE}$ = 0.40 UI provides for a total sum of 0.60 UI deterministic and random jitter budget for the transmitter and interconnect collected over at least 250 TX UIs. The $T_{RX\text{-}EYE\text{-}MEDIAN\text{-}to\text{-}MAX\text{-}JITTER}$ specification ensures a jitter distribution in which the median and the maximum deviation from the median is less than half of the total .6 UI jitter budget collected over at least

250 TX UIs. It should be noted that the median is not the same as the mean. The jitter median describes the point in time where the number of jitter points on either side is approximately equal as opposed to the averaged time value.

9. The receiver input impedance shall result in a differential return loss greater than or equal to 15 dB and a common mode return loss greater than or equal to 6 dB over a frequency range of 50 MHz to 1.25 GHz. This input impedance requirement applies to all valid input levels. The reference impedance for return loss measurements for is 50 ohms to ground for both the D+ and D- line (i.e., as measured by a Vector Network Analyzer with 50 ohm probes - see Figure 4-33). Note: that the series capacitors C_{TX} is optional for the return loss measurement.
10. Impedance during all operating conditions except when in disable.
11. The Rx DC common mode impedance that must be present when the receiver terminations are first enabled to ensure that the Receiver Detect occurs properly. Compensation of this impedance can start immediately and the ($Z_{RX\text{-}COM\text{-}DC}$) Rx DC Common Mode Impedance must be with in the 45 ohms to 55 ohms range by the time Detect is entered.
12. The Rx DC common mode impedance that exists when the receiver terminations are disabled or when no power is present. This helps ensure that the Receiver Detect circuit will not falsely assume a receiver is enabled when it is not.
13. If a receiver is not in Electrical Idle or directed to go into Electrical Idle, and a peak-to-peak differential signal remains below the Electrical Idle threshold for $V_{RX\text{-}IDLE\text{-}DET\text{-}DIFF\text{-}ENTERTIME}$, a surprise removal or disable has occurred.

4.3.4.1. Receiver Compliance Eye Diagram

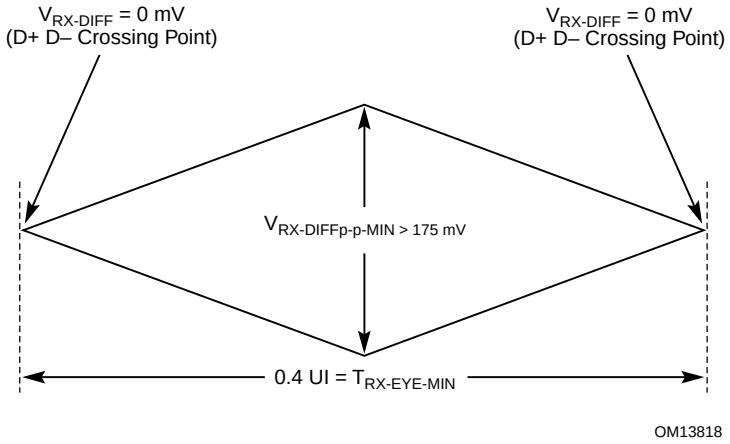

Figure 4-34: Minimum Receiver Eye Timing and Voltage Compliance Specification

5

5. Software Initialization and Configuration

The PCI Express Configuration model supports two configuration space access mechanisms:

- PCI compatible configuration mechanism
- PCI Express enhanced configuration mechanism

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

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

5.1. Configuration Topology

To maintain compatibility with PCI software configuration mechanisms, all PCI Express elements have a PCI-compatible configuration space representation. Each PCI Express Link originates from a logical PCI-PCI Bridge and is mapped into configuration space as the secondary bus of this bridge. The Root Port is a PCI-PCI Bridge structure that originates a PCI Express Link from a PCI Express Root Complex.

A PCI Express Switch is represented by multiple PCI-PCI Bridge structures connecting PCI Express Links to an internal logical PCI bus. 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.

A PCI Express endpoint is mapped into configuration space as a single logical device (Device 0) with one or more logical functions.

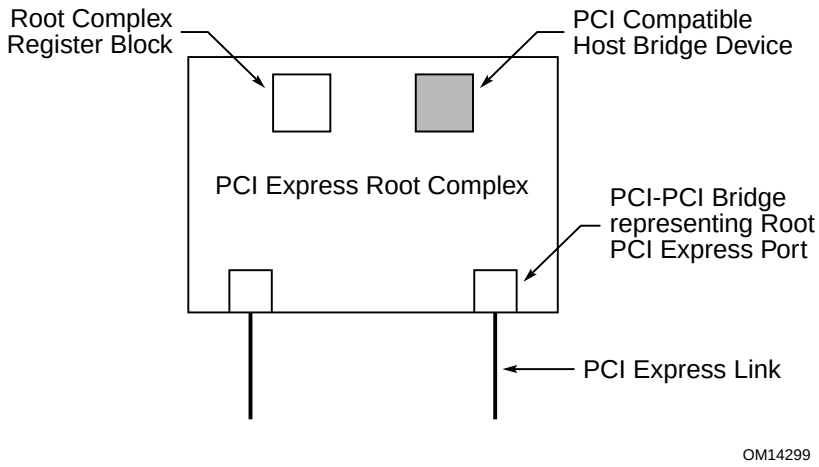

Figure 5-1: PCI Express Root Complex Device Mapping

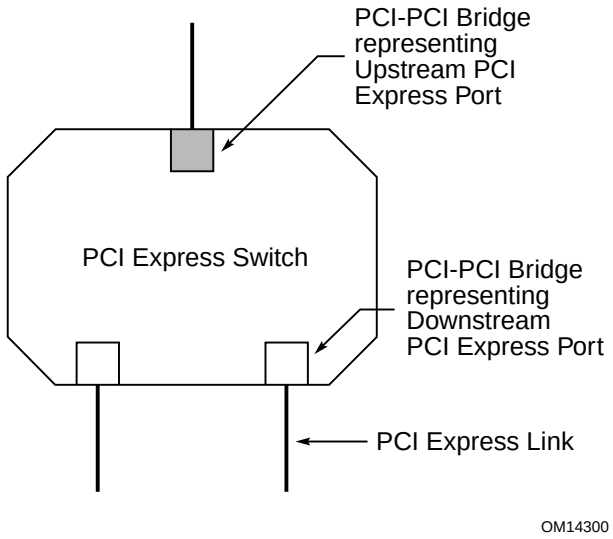

Figure 5-2: PCI Express Switch Device Mapping [24]

5.2. PCI Express Configuration Mechanisms

PCI Express extends the configuration space to 4096 bytes per device function as compared to 256 bytes allowed by PCI Specification Revision 2.3. PCI Express configuration space is divided into a PCI 2.3 compatible region, which consists of the first 256 bytes of a logical device's configuration space and an extended PCI Express configuration space region which consists of the remaining configuration space. The PCI 2.3 compatible region can be accessed using either the mechanism defined in the PCI 2.3 specification or the enhanced PCI Express configuration access mechanism described later in this section. All changes

[24] 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.

made using either access mechanism are equivalent; however, software is not allowed to simultaneously use (interleave) both PCI Express and PCI access mechanisms to access the configuration registers of devices. The extended PCI Express region can only be accessed using the enhanced PCI Express configuration access mechanism.[25]

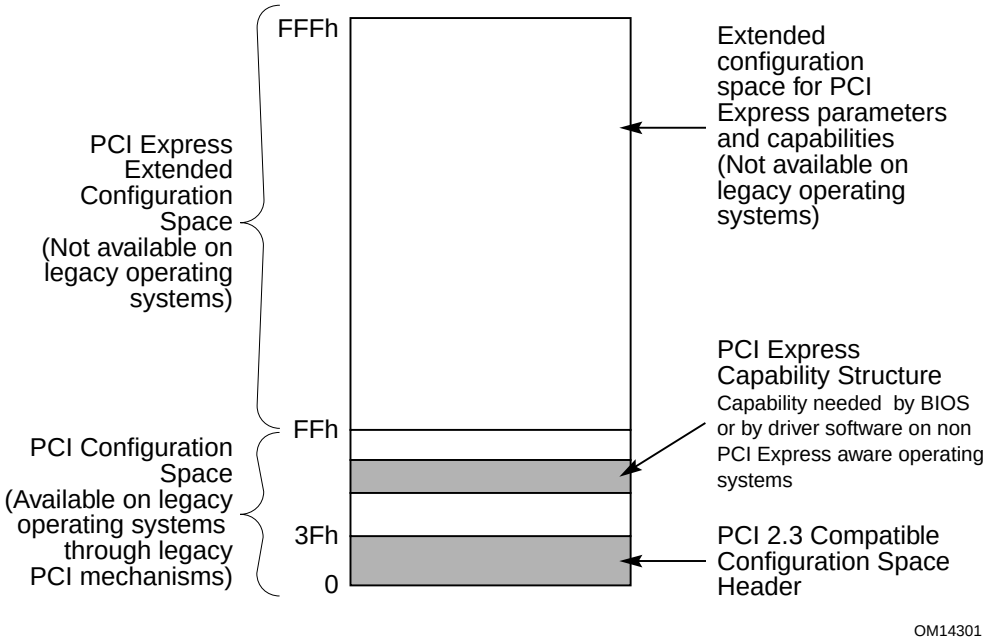

Figure 5-3: PCI Express Configuration Space Layout

5.2.1. PCI 2.3 Compatible Configuration Mechanism

The PCI 2.3 compatible PCI Express configuration mechanism supports the PCI configuration space programming model defined in the *PCI Local Bus Specification, Rev. 2.3*. By adhering to this model, systems incorporating PCI Express interfaces remain compliant with conventional PCI bus enumeration and configuration software.

In the same manner as PCI 2.3 devices, PCI Express devices are required to provide a configuration register space for software-driven initialization and configuration. Except for the differences described in this chapter, the PCI Express configuration header space registers are organized to correspond with the format and behavior defined in the PCI 2.3 Specification (Section 6.1).

The PCI 2.3 compatible configuration access mechanism uses the same Request format as the enhanced PCI Express mechanism. For PCI compatible Configuration Requests, the Extended Register Address field must be all zeros.

[25] Accesses strictly to PCI Express extended configuration space using the enhanced PCI Express configuration access mechanism are allowed to be interleaved with PCI 2.3 configuration access mechanism accesses.

5.2.2. PCI Express Enhanced Configuration Mechanism

The enhanced PCI Express configuration access mechanism utilizes a flat memory-mapped address space to access device configuration registers. In this case, the memory address determines the configuration register accessed and the memory data returns the contents of the addressed register. The mapping from memory address A[27:0] to PCI Express configuration space address is defined in Table 5-1. The base address A[63:28] is allocated in an implementation specific manner and reported by the system firmware to the operating system.

Table 5-1: Configuration Address Mapping

Memory Address	PCI Express Configuration Space
A[27:20]	Bus[7:0]
A[19:15]	Device[4:0]
A[14:12]	Function[2:0]
A[11:8]	Extended Register [3:0]
A[7:0]	Register[7:0]

5.2.2.1. Host Bridge Requirements

The PCI Express Host Bridge is required to translate the memory-mapped PCI Express configuration space accesses from the host processor to PCI Express configuration transactions. The use of Host Bridge PCI class code is reserved for backwards compatibility; host bridge configuration space is opaque to standard PCI Express software and may be implemented in an implementation specific manner that is compatible with PCI Host Bridge Type 0 configuration space.

5.2.2.2. 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.

5.2.3. Root Complex Register Block

Each root port is associated with a 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 5.9.2.

System firmware communicates the base address of the RCRB for each Root Port to the operating system. Multiple Root Ports may be associated with the same RCRB. The RCRB memory-mapped registers must not reside in the same address space as the memory-mapped configuration space.

5.3. Configuration Transaction Rules

5.3.1. Device Number

As in conventional PCI and PCI-X, all PCI Express components are restricted to implementing a single device number on their primary interface (Upstream Port), but may implement up to eight independent functions within that device number. Each internal function is selected based on decoded address information that is provided as part of the address portion of Configuration Request packets.

Switches and Root Complexes must associate only Device 0 on the logical bus from a Downstream Port or a Root Port. Configuration Requests targeting the Bus Number associated with a Port specifying Device Number 0 are delivered to that Port; Configuration Requests specifying all other Device Numbers (1-31) must be terminated with an Unsupported Request Completion Status (equivalent to Master Abort in PCI).[26]

Switches, and components wishing to incorporate more than eight functions at their upstream Port, may implement one or more Type 1 (PCI-to-PCI Bridge) configuration space headers. This allows them to introduce an "internal bus" on which all the device numbers may be utilized, but in this case all address information fields (bus, device and function numbers) must be completely decoded to access the correct register. Any configuration access targeting an unimplemented bus, device or function must return a Completion with Unsupported Request Completion Status.

The following section provides details of the Configuration Space addressing mechanism.

5.3.2. Configuration Transaction Addressing

PCI Express Configuration Requests use the following addressing fields:

- Bus Number – PCI Express maps logical PCI Bus Numbers onto PCI Express Links such that PCI 2.3 compatible configuration software views the configuration space of a PCI Express Hierarchy as a PCI Hierarchy including multiple bus segments.
- Device Number – Device Number association is discussed in Section 5.3.1.
- Function Number – PCI Express also supports multi-function devices using the same discovery mechanism as PCI 2.3.
- Extended Register Number and Register Number – Specify the configuration space address of the register being accessed.

[26] Future switch components that are implemented as a single switch device (without the PCI-PCI Bridges) that is not limited by legacy compatibility requirements may not have this restriction. To accommodate such future implementations, devices may not assume that device 0 is associated with their upstream port.

5.3.3. Configuration Request Routing Rules

For PCI Express Endpoint devices, the following rules apply:

- If Configuration Request Type is 1,
    - Follow the rules for handling Unsupported Requests
- If Configuration Request Type is 0,
    - Determine if the Request addresses a valid local configuration space
        - If so, process the Request
        - If not, follow rules for handling Unsupported Requests

For Switches and PCI Express-PCI Bridges, the following rules apply:

- Propagation of Configuration Requests from Downstream to Upstream as well as peer-to-peer are not supported
    - Configuration Requests are initiated only by the Host Bridge
- If Configuration Request Type is 0,
    - Determine if the Request addresses a valid local configuration space
        - If so, process the Request
        - If not, follow rules for handling Unsupported Requests
- If Configuration Request Type is 1, apply the following tests, in sequence, to the Bus Number field:
    - If in the case of a PCI Express-PCI Bridge, equal to the bus number assigned to secondary PCI bus or, in the case of a Switch or Root Complex, equal to the bus number and decoded device numbers assigned to one of the Root (Root Complex) or Downstream Ports (Switch),
        - Transform the Request to Type 0
        - Forward the Request to that Downstream Port (or PCI bus, in the case of a PCI Express-PCI Bridge)
    - If not equal to the bus number of any of Downstream Ports or secondary PCI bus, but in the range of bus numbers assigned to one of a Downstream Port or secondary PCI bus,
        - Forward the Request to that Downstream Port interface without modification
    - Else (none of the above) –
        - The Request is invalid - follow the rules for handling Unsupported Requests

- PCI Express-PCI Bridges must terminate as Unsupported Requests any Configuration Requests directed towards the PCI bus for which the Extended Register Address field is non-zero

  Note: This type of access is a consequence of a programming error.

For Root Complexes:

- Configuration Requests addressing Bus 0 are processed by the Root Complex.
- Configuration Requests addressing other buses are processed according to the rules for Switches (above)

For all types of devices:

All other configuration space addressing fields are decoded according to the PCI Local Bus Specification.

5.3.4. Generating PCI Special Cycles using PCI Configuration Mechanism #1

Generating PCI Special Cycles using PCI Configuration Mechanism Number One (see the *PCI Local Bus Specification, Rev. 2.3* for details), and handling of such Requests, is not required.

5.4. Configuration Register Types

Configuration register fields are assigned one of the attributes described in Table 5-2.

Table 5-2: Register (and Register Bit-Field) Types

Register Attribute	Description
RO	Read-only register: Register bits are read-only and cannot be altered by software.
RW	Read-Write register: Register bits are read-write and may be either set or cleared by software to the desired state.
RW1C	Read-only status, Write-1-to-clear status register: Register bits indicate status when read, a set bit indicating a status event may be cleared by writing a 1. Writing a 0 to RW1C bits has no effect.
ROS	Sticky bit - Read-only register: Register bits are read-only and cannot be altered by software. Bits are not cleared by reset and can only be reset with “Power Good Reset” (see Section 7.6). Devices that consume AUX power are not allowed to reset sticky bits on “Power Good Reset” when AUX power consumption (either via AUX power or PME Enable) is enabled.

Register Attribute	Description
RWS	Sticky bit - Read-Write register: Register bits are read-write and may be either set or cleared by software to the desired state. Bits are not cleared by reset and can only be reset with "Power Good Reset" (see Section 7.6). Devices that consume AUX power are not allowed to reset sticky bits on "Power Good Reset" when AUX power consumption (either via AUX power or PME Enable) is enabled.
RW1CS	Sticky bit - Read-only status, Write-1-to-clear status register: Register bits indicate status when read, a set bit indicating a status event may be cleared by writing a 1. Writing a 0 to RW1CS bits has no effect. Bits are not cleared by reset and can only be reset with "Power Good Reset" (see Section 7.6). Devices that consume AUX power are not allowed to reset sticky bits on "Power Good Reset" when AUX power consumption (either via AUX power or PME Enable) is enabled.
HwInit	Hardware Initialized: Register bits are initialized by firmware or hardware mechanisms such as pin strapping or serial EEPROM. Bits are read-only after initialization and can only be reset (for write-once by firmware) with "Power Good Reset" (see Section 7.6).
RsvdP	Reserved and Preserved: Reserved for future RW implementations; software must preserve value read for writes to bits.
RsvdZ	Reserved and Zero: Reserved for future RW1C implementations; software must use 0 for writes to bits.

5.5. PCI-Compatible Configuration Registers

The first 256 bytes of the PCI Express configuration space form the PCI 2.3 compatibility region. This region completely aliases the PCI 2.3 configuration space of the device/function. Legacy PCI devices may also be accessed via enhanced PCI Express configuration access mechanism without requiring any modifications to the device hardware or device driver software. This section establishes the mapping between PCI 2.3 and PCI Express for format and behavior of PCI 2.3 compatible registers.

All registers and fields not described in this section are assumed to have the exact same definition as in PCI 2.3.

5.5.1. Type 0/1 Common Configuration Space

Figure 5-4 details allocation for common register fields of PCI 2.3 Type 0 and Type 1 Configuration Space Headers for PCI Express devices.

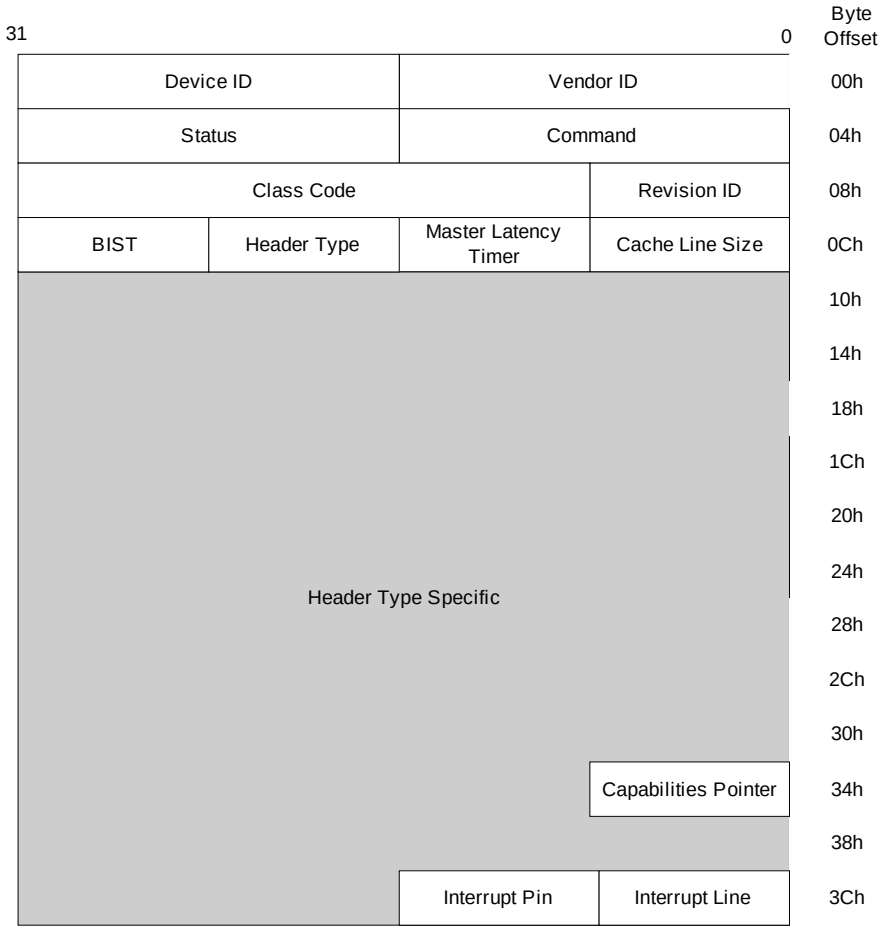

Figure 5-4: Common Configuration Space Header

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.

5.5.1.1. Command Register (Offset 04h)

Table 5-3 establishes the mapping between PCI 2.3 and PCI Express for PCI 2.3 configuration space Command register.

Table 5-3: Command Register

Bit Location	Register Description	Attributes
2	**Bus Master Enable** – Controls the ability of a PCI Express agent to issue memory and I/O read/write requests. Disabling this bit prevents a PCI Express agent from issuing any memory or I/O read/write requests. Note that as MSI interrupt messages are in-band memory writes, disabling the bus master enable bit disables MSI interrupt messages as well. Default value of this field is 0.	RW
3	**Special Cycle Enable** – Does not apply to PCI Express. Must be hardwired to 0.	RO
4	**Memory Write and Invalidate –** Does not apply to PCI Express. Must be hardwired to 0.	RO
5	**VGA Palette Snoop –** Does not apply to PCI Express. Must be hardwired to 0.	RO
6	**Parity Error Enable –** See Section 5.5.1.7. Default value of this field is 0.	RW
7	**IDSEL Stepping / Wait Cycle Control –** Does not apply to PCI Express. Must be hardwired to 0.	RO
8	**SERR Enable –** See Section 5.5.1.7. This bit when set enables reporting of non-fatal and fatal errors to the Root Complex. Note that PCI Express-specific error register bits take precedence over this bit. Default value of this field is 0.	RW
9	**Fast Back-to-Back Transactions Enable –** Does not apply to PCI Express. Must be hardwired to 0.	RO
10	**Interrupt Disable -** Controls the ability of a PCI Express device to generate INTx interrupt messages. When set, devices are prevented from generating INTx interrupt messages. Any INTx emulation interrupts already asserted must be deasserted when this bit is set. Default value of this field is 0.	RW

5.5.1.2. Status Register (Offset 06h)

Table 5-4 establishes the mapping between PCI 2.3 and PCI Express for PCI 2.3 configuration space Status register.

Table 5-4: Status Register

Bit Location	Register Description	Attributes
3	**Interrupt Status -** Indicates that an INTx interrupt message is pending internally to the device. Default value of this field is 0.	RO
4	**Capabilities List** – Indicates the presence of an extended capability list item. Since all PCI Express devices are required to implement the PCI Express capability structure, this bit must be set to 1.	RO
5	**66 MHz Capable –** Does not apply to PCI Express. Must be hardwired to 0.	RO
7	**Fast Back-to-Back Transactions Capable –** Does not apply to PCI Express. Must be hardwired to 0.	RO
8	**Master Data Parity Error –** See Section 5.5.1.7. This bit is set by Requestor (Primary Side for Type 1 Configuration Space Header Device) if its Parity Error Enable bit is set and either of the following two conditions occurs: • Requestor receives a Completion marked poisoned • Requestor poisons a write Request If the Parity Error Enable bit is cleared, this bit is never set. Default value of this field is 0.	RW1C
10:9	**DEVSEL Timing –** Does not apply to PCI Express. Must be hardwired to 0.	RO
11	**Signaled Target Abort –** See Section 5.5.1.7. This bit is set when a device (Primary Side for Type 1 Configuration Space Header device for requests completed by the Type 1 Header device itself) completes a Request using Completer Abort Completion Status. Default value of this field is 0.	RW1C
12	**Received Target Abort –** See Section 5.5.1.7. This bit is set when a Requestor (Primary Side for Type 1 Configuration Space Header device for requests initiated by the Type 1 Header device itself) receives a Completion with Completer Abort Completion Status. Default value of this field is 0.	RW1C

Bit Location	Register Description	Attributes
13	**Received Master Abort –** See Section 5.5.1.7. This bit is set when a Requestor (Primary Side for Type 1 Header Configuration Space Header device for requests initiated by the Type 1 Header device itself) receives a Completion with Unsupported Request Completion Status. Default value of this field is 0.	RW1C
14	**Signaled System Error –** See Section 5.5.1.7. This bit is set when a device sends a ERR_FATAL or ERR_NONFATAL message. Default value of this field is 0.	RW1C
15	**Detected Parity Error –** See Section 5.5.1.7. This bit is set by a device (Primary Side for Type 1 Configuration Space Header device) whenever it receives a poisoned TLP, regardless of the state the Parity Error Enable bit. Default value of this field is 0.	RW1C

5.5.1.3. Cache Line Size Register (Offset 0Ch)

The cache line size register is set by the system firmware and the operating system to system cache line size. However, note that legacy PCI 2.3 software may not always be able to program this field correctly especially in case of hot-plug devices. This field is implemented by PCI Express devices as a read-write field for legacy compatibility purposes but has no impact on any PCI Express device functionality.

5.5.1.4. Master Latency Timer Register (Offset 0Dh)

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

5.5.1.5. Interrupt Line Register (Offset 3Ch)

As in PCI 2.3, the Interrupt Line register communicates interrupt line routing information. The register is read/write and must be implemented by any device (or device function) that uses an interrupt pin (see following description). Values in this register are programmed by system software and are system architecture specific. The device itself does not use this value; rather the value in this register is used by device drivers and operating systems.

5.5.1.6. Interrupt Pin Register (Offset 3Dh)

The Interrupt Pin is a read-only register that identifies the legacy interrupt message(s) the device (or device function) uses; refer to Section 7.1 for further details. Valid values are 1, 2,

3, and 4 that map to legacy interrupt messages for INTA, INTB, INTC, and INTD respectively; a value of 0 indicates that the device uses no legacy interrupt message(s).

5.5.1.7. Error Registers

The error control/status register bits in the Command and Status registers (see Section 5.5.1.1 and Section 5.5.1.2 respectively) control PCI compatible error reporting for both PCI and PCI Express devices. Mapping of PCI Express errors onto PCI errors is also discussed in Section 7.2.5.1. In addition to the PCI compatible error control and status, PCI Express error reporting may be controlled separately from PCI devices through the PCI Express Capability Structure described in Section 5.8. The PCI compatible error control and status register fields do not have any effect on PCI Express error reporting enabled through the PCI Express Capability Structure. PCI Express devices may also implement optional advanced error reporting as described in Section 5.10.

227

5.5.2. Type 0 Configuration Space Header

Figure 5-5 details allocation for register fields of PCI 2.3 Type 0 Configuration Space Header for PCI Express devices.

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

Figure 5-5: Type 0 Configuration Space Header

Section 5.5.1 details the PCI Express-specific registers that are valid for all Configuration Space Header types. The PCI Express-specific interpretation of registers specific to Type 0 PCI 2.3 Configuration Space Header is defined in this section.

5.5.2.1. Min_Gnt/Max_Lat Registers (Offset 3Eh/3Fh)

These registers do not apply to PCI Express. They must be read-only and hardwired to 0.

5.5.3. Type 1 Configuration Space Header

Figure 5-6 details allocation for register fields of PCI 2.3 Type 1 Configuration Space Header for PCI Express devices.

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

Figure 5-6: Type 1 Configuration Space Header

Section 5.5.1 details the PCI Express-specific registers that are valid for all Configuration Space Header types. The PCI Express-specific interpretation of registers specific to Type 1 PCI 2.3 Configuration Space Header is defined in this section.

5.5.3.1. Secondary Latency Timer (Offset 1Bh)

This register does not apply to PCI Express. It must be read-only and hardwired to 0.

5.5.3.2. Secondary Status Register (Offset 1Eh)

Table 5-5 establishes the mapping between PCI 2.3 and PCI Express for PCI 2.3 configuration space Secondary Status register.

Table 5-5: Secondary Status Register

Bit Location	Register Description	Attributes
5	**66 MHz Capable –** Does not apply to PCI Express. Must be hardwired to 0.	RO
7	**Fast Back-to-Back Transactions Capable –** Does not apply to PCI Express. Must be hardwired to 0.	RO
8	**Master Data Parity Error –** See Section 5.5.1.7. This bit is set by the Secondary side Requestor if the Parity Error Response bit is set and either of the following two conditions occurs: • Requestor receives Completion marked poisoned • Requestor poisons a write Request If the Parity Error Response bit is cleared, this bit is never set. Default value of this field is 0.	RW1C
10:9	**DEVSEL Timing –** Does not apply to PCI Express. Must be hardwired to 0.	RO
11	**Signaled Target Abort –** See Section 5.5.1.7. This bit is set when the Secondary Side for Type 1 Configuration Space Header device (for requests completed by the Type 1 Header device itself) completes a Request using Completer Abort Completion Status. Default value of this field is 0.	RW1C
12	**Received Target Abort –** See Section 5.5.1.7. This bit is set when the Secondary Side for Type 1 Configuration Space Header device (for requests initiated by the Type 1 Header device itself) receives a Completion with Completer Abort Completion Status. Default value of this field is 0.	RW1C
13	**Received Master Abort –** See Section 5.5.1.7. This bit is set when the Secondary Side for Type 1 Configuration Space Header device (for requests initiated by the Type 1 Header device itself) receives a Completion with Unsupported Request Completion Status. Default value of this field is 0.	RW1C

Bit Location	Register Description	Attributes
14	**Received System Error –** See Section 5.5.1.7. This bit is sent when a device sends a ERR_FATAL or ERR_NONFATAL message. Default value of this field is 0.	RW1C
15	**Detected Parity Error –** See Section 5.5.1.7. This bit is set by the Secondary Side for a Type 1 Configuration Space Header device whenever it receives a poisoned TLP, regardless of the state the Parity Error Response bit. Default value of this field is 0.	RW1C

5.5.3.3. Bridge Control Register (Offset 3Eh)

Table 5-6 establishes the mapping between PCI 2.3 and PCI Express for PCI 2.3 configuration space Bridge Control register.

Table 5-6: Bridge Control Register

Bit Location	Register Description	Attributes
0	**Parity Error Response Enable** – See Section 5.5.1.7. This bit controls the response to poisoned TLPs. Default value of this field is 0.	RW
1	**SERR Enable** – See Section 5.5.1.7. This bit controls forwarding of ERR_COR, ERR_NONFATAL and ERR_FATAL from secondary to primary. Default value of this field is 0.	RW
5	**Master Abort Mode** – Does not apply to PCI Express. Must be hardwired to 0.	RO
6	**Secondary Bus Reset** –Setting this bit triggers a warm reset on the corresponding PCI Express Port and the PCI Express hierarchy domain subordinate to the Port. Default value of this field is 0.	RW
7	**Fast Back-to-Back Transactions Enable –** Does not apply to PCI Express. Must be hardwired to 0.	RO
8	**Primary Discard Timer** – Does not apply to PCI Express. Must be hardwired to 0.	RO
9	**Secondary Discard Timer** – Does not apply to PCI Express. Must be hardwired to 0.	RO
10	**Discard Timer Status –** Does not apply to PCI Express. Must be hardwired to 0.	RO
11	**Discard Timer SERR Enable –** Does not apply to PCI Express. Must be hardwired to 0.	RO

5.6. PCI Power Management Capability Structure

This structure is required for all PCI Express devices. Figure 5-7 details allocation of the PCI PM Capability Structure register fields in a PCI Express Context. PCI Express devices are required to support D0 and D3 device states (refer to Section 6.1.1); PCI-PCI bridge structures representing PCI Express ports as described in Section 5.1 are required to indicate PME wake 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 but the PCI-PCI bridge function itself is not internally asserting PME.

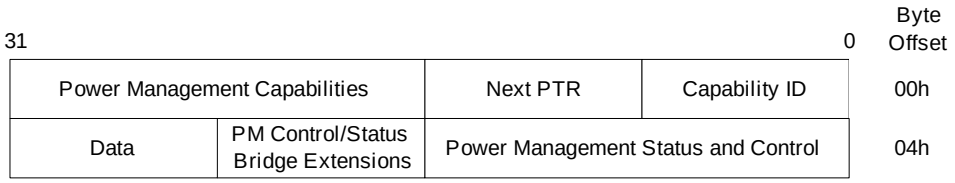

Figure 5-7: PCI Power Management Capability Structure

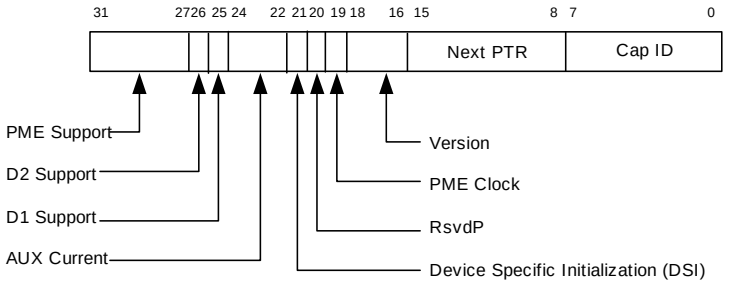

Figure 5-8: Power Management Capabilities

Figure 5-8 details allocation of register fields for Power Management Capabilities register; Table 5-7 establishes the mapping between PCI 2.3 and PCI Express for this register.

Table 5-7: Power Management Capabilities

Bit Location	Register Description	Attributes
7:0	**Capability ID –** Must be set to 01h	RO
15:8	**Next Capability Pointer**	RO
18:16	**Version –** Set to 02h for this version of the specification.	RO
19	**PME Clock –** Does not apply to PCI Express. Must be hardwired to 0.	RO
21	**Device Specific Initialization**	RO
24:22	**AUX Current**	RO

Bit Location	Register Description	Attributes
25	**D1 Support**	RO
26	**D2 Support**	RO
31:27	**PME Support –** Must be set for PCI-PCI bridge structures representing ports on root complexes/switches.	RO

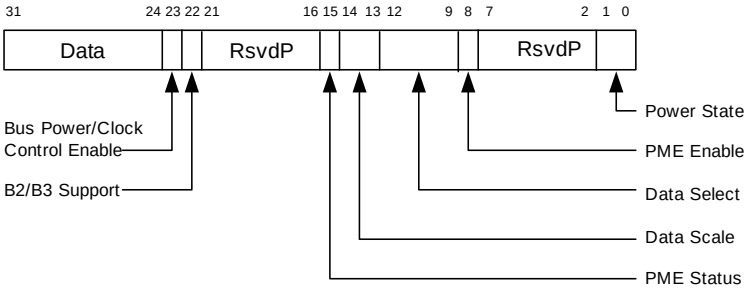

Figure 5-9: Power Management Status/Control

Figure 5-9 details allocation of register fields for Power Management Status and Control register; Table 5-8 establishes the mapping between PCI 2.3 and PCI Express for this register.

Table 5-8: Power Management Status/Control

Bit Location	Register Description	Attributes
1:0	**Power State**	RW
8	**PME Enable**	RWS
12:9	**Data Select**	RW
14:13	**Data Scale**	RO
15	**PME Status**	RW1CS
22	**B2/B3 Support –** Does not apply to PCI Express. Must be hardwired to 0.	RO
23	**Bus Power/Clock Control Enable –** Does not apply to PCI Express. Must be hardwired to 0.	RO
31:24	**Data**	RO

5.7. MSI Capability Structure

This structure is required for all PCI Express devices that are capable of generating interrupts. Definition of register structure associated with MSI is compatible with PCI 2.3 specification.

5.8. PCI Express Capability Structure

PCI Express defines a capability structure in PCI 2.3 compatible configuration space (first 256 bytes) as shown in Figure 5-3 for identification of a PCI Express device and indicates support for new PCI Express features. The PCI Express Capability Structure is required for PCI Express devices. The capability structure is a mechanism for enabling PCI software transparent features requiring support on legacy operating systems. In addition to identifying a PCI Express device, the PCI Express Capability Structure is used to provide access to PCI Express specific Control/Status registers and related Power Management enhancements.

Figure 5-10 details allocation of register fields in the PCI Express Capability Structure. The PCI Express Capabilities, Device Capabilities, Device Status/Control, Link Capabilities and Link Status/Control registers are required for all PCI Express devices. Endpoints are not required to implement registers other than those listed above and terminate the capability structure.

Slot Capabilities and Slot Status/Control registers are required for Switch Downstream and Root Ports if a slot is implemented on the port. Root Control/Status registers are required for root ports. Root ports must implement the entire PCI Express Capability Structure.

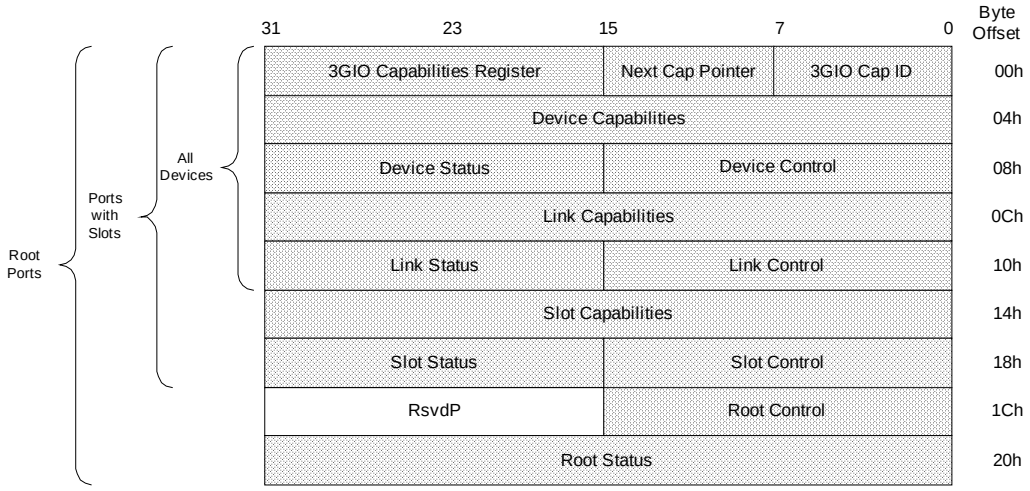

Figure 5-10: PCI Express Capability Structure

5.8.1. PCI Express Capability List Register (Offset 00h)

The PCI Express Capability List register enumerates the PCI Express Capability Structure in the PCI 2.3 configuration space capability list. Figure 5-11 details allocation of register fields in the PCI Express Capability List register; Table 5-9 provides the respective bit definitions.

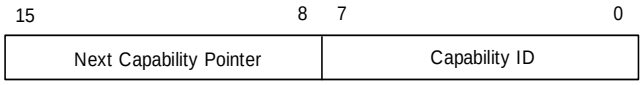

Figure 5-11: PCI Express Capability List Register

Table 5-9: PCI Express Capability List Register

Bit Location	Register Description	Attributes
7:0	**Capability ID** – Indicates PCI Express Capability Structure. This field must return a Capability ID of (value to be assigned by PCI-SIG) indicating that this is a PCI Express Capability Structure.	RO
15:8	**Next Capability Pointer** – The offset to the next PCI capability structure or 00h if no other items exist in the linked list of capabilities.	RO

5.8.2. PCI Express Capabilities Register (Offset 02h)

The PCI Express Capabilities register identifies PCI Express device type and associated capabilities. Figure 5-12 details allocation of register fields in the PCI Express Capabilities register; Table 5-10 provides the respective bit definitions.

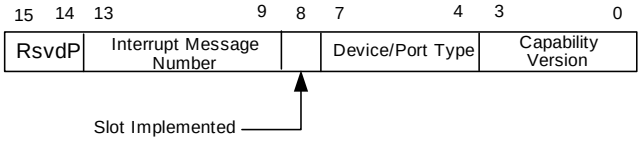

Figure 5-12: PCI Express Capabilities Register

Table 5-10: PCI Express Capabilities Register

<table>
<tr><th>Bit Location</th><th>Register Description</th><th>Attributes</th></tr>
<tr><td>3:0</td><td>Capability Version – Indicates PCI-SIG defined PCI Express capability structure version number.

Must be 1h for this specification.</td><td>RO</td></tr>
<tr><td>7:4</td><td>Device/Port Type – Indicates the type of PCI Express device. Defined encodings are:
<table>
<tr><td>0000b</td><td>PCI Express Endpoint device</td></tr>
<tr><td>0001b</td><td>Legacy PCI Express Endpoint device</td></tr>
<tr><td>0100b</td><td>Root Port of PCI Express Root Complex*</td></tr>
<tr><td>0101b</td><td>Upstream Port of PCI Express Switch*</td></tr>
<tr><td>0110b</td><td>Downstream Port of PCI Express Switch*</td></tr>
<tr><td>0111b</td><td>PCI Express-to-PCI/PCI-X Bridge*</td></tr>
</table>
All other encodings are reserved.

*This value is only valid for devices/functions that implement a Type 01h PCI Configuration Space Header.

Native PCI Express Endpoint devices that do not require I/O resources for correct operation indicate a device Type of 0000b; such devices may request I/O resources (through BARs) for legacy boot support but system software is allowed to close requested I/O resources once appropriate services are made available to device specific software for access to device specific resources claimed through memory BARs.

Legacy PCI Express Endpoint devices that require I/O resources claimed through BARs for correct operation indicate a Device Type of 0001b.</td><td>RO</td></tr>
<tr><td>8</td><td>Slot Implemented – This bit when set indicates that the PCI Express Link associated with this port is connected to a slot (as compared to being connected to an integrated component or being disabled).

This field is valid for the following PCI Express device/Port Types:
<table>
<tr><td>0100b</td><td>Root Port of PCI Express Root Complex</td></tr>
<tr><td>0110b</td><td>Downstream Port of PCI Express Switch</td></tr>
</table></td><td>HwInit</td></tr>
<tr><td>13:9</td><td>Interrupt Message Number – If this function is allocated more than one MSI interrupt number, this register is required to contain the offset between the base Message Data and the MSI Message that is generated when any of status bits in either the Slot Status register or the Root Port Status register of this capability structure are set.

Hardware is required to update this field so that it is correct if the number of MSI Messages assigned to the device changes.</td><td>RO</td></tr>
</table>

5.8.3. Device Capabilities Register (Offset 04h)

The Device Capabilities register identifies PCI Express device specific capabilities. Figure 5-13 details allocation of register fields in the Device Capabilities register; Table 5-11 provides the respective bit definitions.

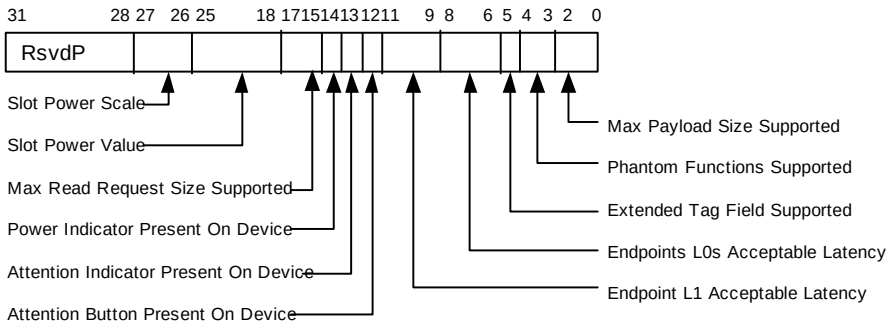

Figure 5-13: Device Capabilities Register

Table 5-11: Device Capabilities Register

Bit Location	Register Description	Attributes
2:0	**Max_Payload_Size Supported** – This field indicates the maximum payload size that the device can support for TLPs. Defined encodings are: 000b 128B max payload size 001b 256B max payload size 010b 512B max payload size 011b 1024B max payload size 100b 2048B max payload size 101b 4096B max payload size 110b Reserved 111b Reserved	RO

<table>
<tr><th>Bit Location</th><th>Register Description</th><th>Attributes</th></tr>
<tr><td>4:3</td><td>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 2.4.2 for description of Tag Extensions.

This field indicates the number of most significant bits of the function number portion of Requester ID that are logically combined with the Tag identifier. Defined encodings are:
<table>
<tr><td>00b</td><td>No function number bits used for Phantom Functions; device may implement all function numbers.</td></tr>
<tr><td>01b</td><td>First most significant bit of function number in Requestor ID used for Phantom Functions; device may implement functions 0-3. Functions 0, 1, 2, and 3 may claim functions 4, 5, 6, and 7 as Phantom Functions respectively.</td></tr>
<tr><td>10b</td><td>First two most significant bits of function number in Requestor ID used for Phantom Functions; device may implement functions 0-1. Function 0 may claim functions 2, 4, and 6 as Phantom Functions, function 1 may claim functions 3, 5, and 7 as Phantom Functions.</td></tr>
<tr><td>11b</td><td>All three bits of function number in Requestor ID used for Phantom Functions; device must be a single function 0 device that may claim all other functions as Phantom Functions.</td></tr>
</table>
Note that Phantom Function support for the Device must be enabled by the corresponding control field in the Device Control register.

A Root Port must always return 0b in this field.</td><td>RO</td></tr>
<tr><td>5</td><td>Extended Tag Field Supported – This field indicates the maximum supported size of the Tag field. Defined encodings are:
<table>
<tr><td>0b</td><td>5-bit Tag field supported</td></tr>
<tr><td>1b</td><td>8-bit Tag field supported</td></tr>
</table>
Note that 8-bit Tag field support must be enabled by the corresponding control field in the Device Control register.

A Root Port must always return 0b in this field.</td><td>RO</td></tr>
</table>

<table>
<tr><th>Bit Location</th><th>Register Description</th><th>Attributes</th></tr>
<tr><td>8:6</td><td>Endpoint L0s Acceptable Latency – This field indicates the acceptable 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 Active State Link PM L0s entry can be used with no loss of performance. Defined encodings are:
<table><tr><td>000b</td><td>Less than 64 ns</td></tr><tr><td>001b</td><td>64 ns-128 ns</td></tr><tr><td>010b</td><td>128 ns-256 ns</td></tr><tr><td>011b</td><td>256 ns-512 ns</td></tr><tr><td>100b</td><td>512 ns-1 µs</td></tr><tr><td>101b</td><td>1 µs-2 µs</td></tr><tr><td>110b</td><td>2 µs-4 µs</td></tr><tr><td>111b</td><td>More than 4 µs</td></tr></table></td><td>RO</td></tr>
<tr><td>11:9</td><td>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 Active State Link PM L1 entry can be used with no loss of performance. Defined encodings are:
<table><tr><td>000b</td><td>Less than 1µs</td></tr><tr><td>001b</td><td>1 µs-2 µs</td></tr><tr><td>010b</td><td>2 µs-4 µs</td></tr><tr><td>011b</td><td>4 µs-8 µs</td></tr><tr><td>100b</td><td>8 µs-16 µs</td></tr><tr><td>101b</td><td>16 µs-32 µs</td></tr><tr><td>110b</td><td>32 µs-64 µs</td></tr><tr><td>111b</td><td>More than 64 µs</td></tr></table></td><td>RO</td></tr>
<tr><td>12</td><td>Attention Button Present – This bit when set indicates that an Attention Button is implemented on the card or module.

This bit is valid for the following PCI Express device Types:
<table><tr><td>0000b</td><td>PCI Express Endpoint device</td></tr><tr><td>0001b</td><td>Legacy PCI Express Endpoint device</td></tr></table></td><td>RO</td></tr>
</table>

<table>
<tr><th>Bit Location</th><th>Register Description</th><th>Attributes</th></tr>
<tr><td>13</td><td>Attention Indicator Present – This bit when set indicates that an Attention Indicator is implemented on the card or module.
This bit is valid for the following PCI Express device Types:<table><tr><td>0000b</td><td>PCI Express Endpoint device</td></tr><tr><td>0001b</td><td>Legacy PCI Express Endpoint device</td></tr></table></td><td>RO</td></tr>
<tr><td>14</td><td>Power Indicator Present – This bit indicates when set indicates that a Power Indicator is implemented on the card or module.
This bit is valid for the following PCI Express device Types:<table><tr><td>0000b</td><td>PCI Express Endpoint device</td></tr><tr><td>0001b</td><td>Legacy PCI Express Endpoint device</td></tr></table></td><td>RO</td></tr>
<tr><td>17:15</td><td>Max_Read_Request_Size Supported (Root Complex only) - This field indicates the maximum Read Request size for the Device as a Completer. Defined encodings are:<table><tr><td>000b</td><td>Reserved</td></tr><tr><td>001b</td><td>Reserved</td></tr><tr><td>010b</td><td>512B max read request size</td></tr><tr><td>011b</td><td>1024B max read request size</td></tr><tr><td>100b</td><td>2048B max read request size</td></tr><tr><td>101b</td><td>4096B max read request size</td></tr><tr><td>110b</td><td>Reserved</td></tr><tr><td>111b</td><td>Reserved</td></tr></table></td><td>RO</td></tr>
<tr><td>25:18</td><td>Slot Power Limit Value (Upstream Ports only) – In combination with the Slot Power Limit Scale value, specifies the upper limit on power supplied by slot.
Power limit (in Watts) calculated by multiplying the value in this field by the value in the Slot Power Limit Scale field.
This value is set by the Set_Slot_Power_Limit message or hardwired to 0000 0000b (see Section 7.9). The default value is 0000 0000b.</td><td>RO</td></tr>
<tr><td>27:26</td><td>Slot Power Limit Scale (Upstream Ports only) – Specifies the scale used for the Slot Power Limit Value.
Range of Values
00b = 1.0x (25.5-255)
01b = 0.1x (2.55-25.5)
10b = 0.01x (0.255-2.55)
11b = 0.001x (0.0-0.255)
This value is set by the •Set_Slot_Power_Limit message or hardwired to 00b (see Section 7.9). The default value is all 00b.</td><td>RO</td></tr>
</table>

5.8.4. Device Control Register (Offset 08h)

The Device Control register controls PCI Express device specific parameters. Figure 5-14 details allocation of register fields in the Device Control register; Table 5-12 provides the respective bit definitions.

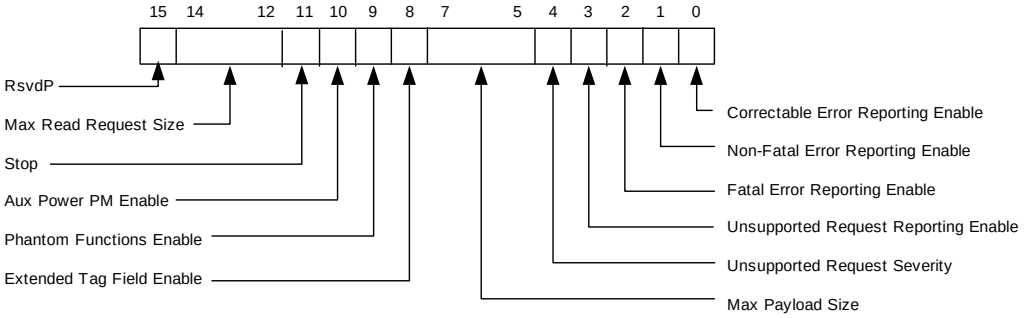

Figure 5-14: Device Control Register

Table 5-12: Device Control Register

Bit Location	Register Description	Attributes
0	**Correctable Error Reporting Enable** – This bit controls reporting of correctable errors. Refer to Section 7.2 for further details. For a multi-function device, this bit controls error reporting for each function from point-of-view of the respective function. For a Root Port, the reporting of correctable errors is internal to the root. No external ERR_CORR message is generated. Default value of this field is 0.	RW
1	**Non-Fatal Error Reporting Enable** - This bit controls reporting of non-fatal errors. Refer to Section 7.2 for further details. For a multi-function device, this bit controls error reporting for each function from point-of-view of the respective function. For a Root Port, the reporting of non-fatal errors is internal to the root. No external ERR_NONFATAL message is generated. Default value of this field is 0.	RW
2	**Fatal Error Reporting Enable** - This bit controls reporting of fatal errors. Refer to Section 7.2 for further details. For a multi-function device, this bit controls error reporting for each function from point-of-view of the respective function. For a Root Port, the reporting of fatal errors is internal to the root. No external ERR_FATAL message is generated. Default value of this field is 0.	RW

241

<table>
<tr><th>Bit Location</th><th>Register Description</th><th>Attributes</th></tr>
<tr><td>3</td><td>Unsupported Request Reporting Enable – This bit enables reporting of Unsupported Requests when set. Refer to Section 7.2 for further details. For a multi-function device, this bit controls error reporting for each function from point-of-view of the respective function. Note that the reporting of error messages (ERR_CORR, ERR_NONFATAL, ERR_FATAL) received by Root Port is controlled exclusively by Root Port Command Register described in Section 5.8.12.

Default value of this field is 0.</td><td>RW</td></tr>
<tr><td>4</td><td>Unsupported Request Severity – This bit controls whether ERR_NONFATAL (0) or ERR_FATAL (1) is used for reporting Unsupported Request errors.

Default value of this field is 0.</td><td>RW</td></tr>
<tr><td>7:5</td><td>Max_Payload_Size - This field sets maximum TLP payload size for the device. As a receiver, the device must handle TLPs as large as the set value; as transmitter, the device must not generate TLPs exceeding the set value. Permissible values that can be programmed are indicated by the Max_Payload_Size Supported in the Device Capabilities register (refer to Section 5.8.3). Defined encodings for this field are:
<table>
<tr><td>000b</td><td>128B max payload size</td></tr>
<tr><td>001b</td><td>256B max payload size</td></tr>
<tr><td>010b</td><td>512B max payload size</td></tr>
<tr><td>011b</td><td>1024B max payload size</td></tr>
<tr><td>100b</td><td>2048B max payload size</td></tr>
<tr><td>101b</td><td>4096B max payload size</td></tr>
<tr><td>110b</td><td>Reserved</td></tr>
<tr><td>111b</td><td>Reserved</td></tr>
</table>
Default value of this field is 001b.</td><td>RW</td></tr>
<tr><td>8</td><td>Extended Tag Field Enable – When set, this bit enables a device to use an 8-bit Tag field as a requester. If the bit is cleared, the device is restricted to a 5-bit Tag field. See Section 2.4.2 for description of Tag extensions.

Default value of this field is 0.

A Root Port does not implement this field.</td><td>RW</td></tr>
<tr><td>9</td><td>Phantom Functions Enable – When set, this bit enables a device to use unclaimed functions as Phantom Functions to extend the number of outstanding transaction identifiers. If the bit is cleared, the device is not allowed to use Phantom Functions. See Section 2.4.2 for description of Tag extensions.

Default value of this field is 0.

A Root Port does not implement this field.</td><td>RW</td></tr>
</table>

<table>
<tr><th>Bit Location</th><th>Register Description</th><th>Attributes</th></tr>
<tr><td>10</td><td>Auxiliary (AUX) Power PM Enable - This bit when set enables a device to draw AUX power independent of PME AUX power. devices that require AUX power on legacy operating systems should continue to indicate PME AUX power requirements. AUX power is allocated as requested in the AUX_Current field of the Power Management Capabilities Register (PMC), independent of the PME_En bit in the Power Management Control/Status Register (PMCSR) (see Chapter 6). For multi-function devices, a component is allowed to draw AUX power if at least one of the functions has this bit set.

Default value of this field is 0.</td><td>RW</td></tr>
<tr><td>11</td><td>Stop – Writing 1 to this bit signals the device to complete pending transactions. Refer to Section 7.4 for device/function stop synchronization mechanism.

This bit always returns 0 when read.</td><td>RW</td></tr>
<tr><td>14:12</td><td>Max_Read_Request_Size - This field sets maximum Read Request size for the Device as a Requester. The Device must not generate read requests with size exceeding the set value. Permissible values that can be programmed are indicated by the Max_Read_Request_Size Supported in the Device Capabilities register (refer to Section 5.8.3). Defined encodings for this field are:
<table>
<tr><td>000b</td><td>128B max read request size</td></tr>
<tr><td>001b</td><td>256B max read request size</td></tr>
<tr><td>010b</td><td>512B max read request size</td></tr>
<tr><td>011b</td><td>1024B max read request size</td></tr>
<tr><td>100b</td><td>2048B max read request size</td></tr>
<tr><td>101b</td><td>4096B max read request size</td></tr>
<tr><td>110b</td><td>Reserved</td></tr>
<tr><td>111b</td><td>Reserved</td></tr>
</table>
Default value of this field is 010b.</td><td>RW</td></tr>
</table>

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 poor 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.

The mechanism provides a way to simplify a Root Complex implementation by limiting the size of the read requests which the Root Complex, as a Completer must handle.

PCI Express aware operating systems may use the Max_Read_Request_Size mechanism to help enable correct operation of a device whose Max_Payload_Size capability is smaller than the Max_Payload_Size configured for other devices within the same Hierarchy Domain. For such a device, its Max_Read_Request_Size can be configured to equal its Max_Payload_Size. Thus, read completion packets destined for that device are guaranteed never to exceed its Max_Payload_Size even when the Completer's Max_Payload_Size is configured to a higher value. Otherwise, the Max_Payload_Size of the other devices would have to be reduced to the "lowest common denominator" of the devices they send read completions to.

Use of the Max_Read_Request_Size mechanism as described above does not address the issue of devices sending large posted writes to a device whose Max_Payload_Size capability is smaller than their configured Max_Payload_Size. However, for many devices, their programming model doesn't require them to receive posted writes of a size exceeding their Max_Payload_Size capability anyway, making the posted writes issue irrelevant.

5.8.5. Device Status Register (Offset 0Ah)

The Device Status register provides information about PCI Express device specific parameters. Figure 5-15 details allocation of register fields in the Device Status register; Table 5-13 provides the respective bit definitions.

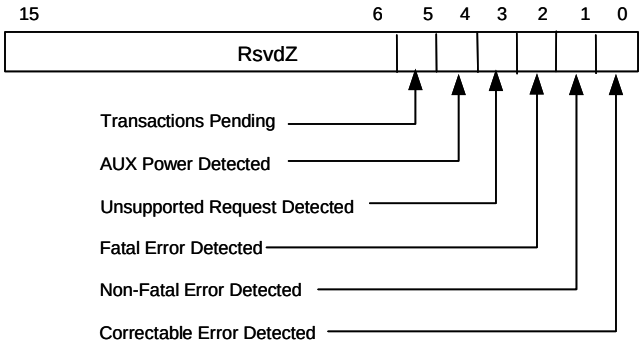

Figure 5-15: Device Status Register

Table 5-13: Device Status Register

Bit Location	Register Description	Attributes
0	**Correctable Error Detected** – This bit indicates status of correctable errors detected. Errors are logged in this register regardless of whether error reporting is enabled or not in the Device Control register. For a multi-function device, each function indicates status of errors as perceived by the respective function. Default value of this field is 0.	RW1C
1	**Non-Fatal Error Detected** – This bit indicates status of non-fatal errors detected. Errors are logged in this register regardless of whether error reporting is enabled or not in the Device Control register. For a multi-function device, each function indicates status of errors as perceived by the respective function. Default value of this field is 0.	RW1C
2	**Fatal Error Detected** - This bit indicates status of fatal errors detected. Errors are logged in this register regardless of whether error reporting is enabled or not in the Device Control register. For a multi-function device, each function indicates status of errors as perceived by the respective function. Default value of this field is 0.	RW1C
3	**Unsupported Request Detected –** This bit indicates that the device received an Unsupported Request. Errors are logged in this register regardless of whether error reporting is enabled or not in the Device Control Register. For a multi-function device, each function indicates status of errors as perceived by the respective function. Default value of this field is 0.	RW1C
4	**AUX Power Detected** - Devices that require AUX power report this bit as set if AUX power is detected by the device.	RO
5	**Transactions Pending** – Indicates whether a device has any transactions pending. A device indicates that transactions are pending (including completions for any outstanding non-posted requests for all used Traffic Classes) by reporting this bit as set. A device may report this bit cleared only when all pending transactions (including completions for any outstanding non-posted requests on any used virtual channel) have been completed. Refer to Section 7.4 for device/function stop synchronization mechanism. This bit must be set by hardware when a 1 is written to the Stop bit in the Device Control register and subsequently cleared (by hardware) when all pending transactions have been completed.	RO

245

5.8.6. Link Capabilities Register (Offset 0Ch)

The Link Capabilities register identifies PCI Express Link specific capabilities. Figure 5-16 details allocation of register fields in the Link Capabilities register; Table 5-14 provides the respective bit definitions.

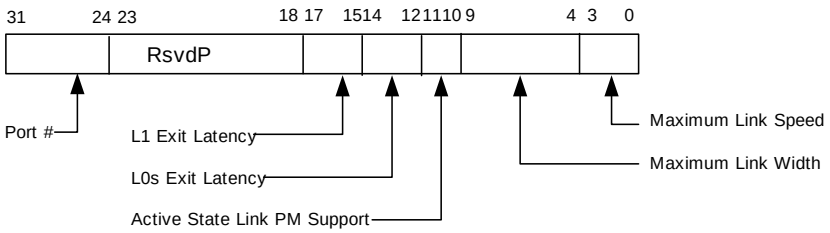

Figure 5-16: Link Capabilities Register

Table 5-14: Link Capabilities Register

Bit Location	Register Description	Attributes
3:0	**Maximum Link Speed** – This field indicates the maximum Link speed of the given PCI Express Link. Defined encodings are: 0001b — 2.5 Gb/s Link All other encodings are reserved.	RO
9:4	**Maximum Link Width** - This field indicates the maximum width of the given PCI Express Link. Defined encodings are: 000000b — Reserved 000001b — x1 000010b — x2 000100b — x4 001000b — x8 001100b — x12 010000b — x16 100000b — x32	RO
11:10	**Active State Link PM Support** – This field indicates the level of active state power management supported on the given PCI Express Link. Defined encodings are: 00b — Reserved 01b — L0s Entry Supported 10b — Reserved 11b — L0s and L1 Supported	RO

<table>
<tr><th>Bit Location</th><th>Register Description</th><th>Attributes</th></tr>
<tr><td>14:12</td><td>L0s Exit Latency – This field indicates the L0s exit latency for the given PCI Express Link. The value reported indicates the length of time this Port requires to complete transition from L0s to L0. Defined encodings are:
<table>
<tr><td>000b</td><td>Less than 64 ns</td></tr>
<tr><td>001b</td><td>64 ns-128 ns</td></tr>
<tr><td>010b</td><td>128 ns-256 ns</td></tr>
<tr><td>011b</td><td>256 ns-512 ns</td></tr>
<tr><td>100b</td><td>512 ns-1 µs</td></tr>
<tr><td>101b</td><td>1 µs-2 µs</td></tr>
<tr><td>110b</td><td>2 µs-4 µs</td></tr>
<tr><td>111b</td><td>Reserved</td></tr>
</table>
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.</td><td>RO</td></tr>
<tr><td>17:15</td><td>L1 Exit Latency – This field indicates the L1 exit latency for the given PCI Express Link. The value reported indicates the length of time this Port requires to complete transition from L1 to L0. Defined encodings are:
<table>
<tr><td>000b</td><td>Less than 1µs</td></tr>
<tr><td>001b</td><td>1 µs-2 µs</td></tr>
<tr><td>010b</td><td>2 µs-4 µs</td></tr>
<tr><td>011b</td><td>4 µs-8 µs</td></tr>
<tr><td>100b</td><td>8 µs-16 µs</td></tr>
<tr><td>101b</td><td>16 µs-32 µs</td></tr>
<tr><td>110b</td><td>32 µs-64 µs</td></tr>
<tr><td>111b</td><td>L1 transition not supported</td></tr>
</table>
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.</td><td>RO</td></tr>
<tr><td>31:24</td><td>Port Number – This field indicates the PCI Express port number for the given PCI Express Link.</td><td>HwInit</td></tr>
</table>

5.8.7. Link Control Register (Offset 10h)

The Link Control register controls PCI Express Link specific parameters. Figure 5-17 details allocation of register fields in the Link Control register; Table 5-15 provides the respective bit definitions.

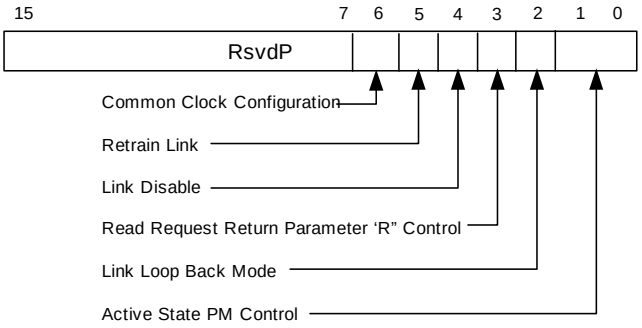

Figure 5-17: Link Control Register

Table 5-15: Link Control Register

Bit Location	Register Description	Attributes
1:0	**Active State Link PM Control** – This field controls the level of active state PM supported on the given PCI Express Link. Defined encodings are: 00b – Disabled 01b – L0s Entry Supported 10b – Reserved 11b – L0s and L1 Entry Supported Default value for this field is 0.	RW
2	**Link Loop Back Mode** – This bit puts a Link in loop-back mode for debug/diagnostic purposes. For multi-function devices, a Link is put in loop-back mode if all functions of component have this bit set. Default value of this field is 0.	RW

Bit Location	Register Description	Attributes
3	**Read Request Return Parameter 'R"Control –** Refer to Section 2.7.6.2.1 for the definition of Read Request Return Parameter. Defined encodings are for "R" capabilities are: 0b – 64 byte 1b – 128 byte PCI Express Endpoints and Switches that do not implement this feature must hardwire the field to 0b. This field is hardwired for a Root Port and returns its "R" support capabilities.	RW
4	**Link Disable** – This bit disables the Link when set; this field is not applicable and reserved for endpoint devices and Upstream Ports of a Switch. Default value of this field is 0.	RW
5	**Retrain Link** – This bit initiates Link retraining when set; this field is not applicable and reserved for endpoint devices and Upstream Ports of a Switch. This bit always returns 0 when read.	RW
6	**Common Clock Configuration –** This bit when set indicates that this component and the component at the opposite end of this Link are operating with a distributed common reference clock. A value of 0 indicates that this component and the component at the opposite end of this Link are operating with asynchronous reference clock. Components utilize this common clock configuration information to report the correct L0s and L1 Exit Latencies. Default value of this field is 0.	RW

5.8.8. Link Status Register (Offset 12h)

The Link Status register provides information about PCI Express Link specific parameters. Figure 5-18 details allocation of register fields in the Link Status register; Table 5-16 provides the respective bit definitions.

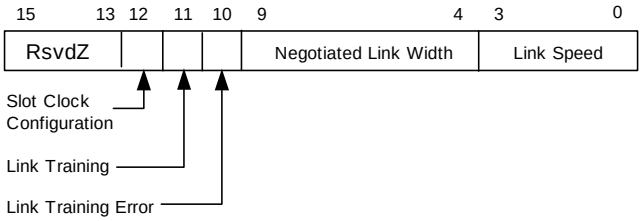

Figure 5-18: Link Status Register

Table 5-16: Link Status Register

Bit Location	Register Description	Attributes
3:0	**Link Speed** – This field indicates the negotiated Link speed of the given PCI Express Link. Defined encodings are: 0001b – 2.5 Gb/s PCI Express Link All other encodings are reserved.	RO
9:4	**Negotiated Link Width** – This field indicates the negotiated width of the given PCI Express Link. Defined encodings are: 000001b – X1 000010b – X2 000100b – X4 001000b – X8 001100b – X12 010000b – X16 100000b – X32 All other encodings are reserved.	RO
10	**Link Training Error** – This read-only bit indicates that a Link training error occurred.	RO
11	**Link Training –** This read-only bit indicates that Link training is in progress; hardware clears this bit once Link training is complete.	RO

Bit Location	Register Description	Attributes
12	**Slot Clock Configuration –** This bit indicates that the component uses the same physical reference clock that the platform provides on the connector. If the device uses an independent clock irrespective of the presence of a reference on the connector, this bit must be clear.	RO

5.8.9. Slot Capabilities Register (Offset 14h)

The Slot Capabilities register identifies PCI Express Slot specific capabilities. Figure 5-19 details allocation of register fields in the Slot Capabilities register; Table 5-17 provides the respective bit definitions.

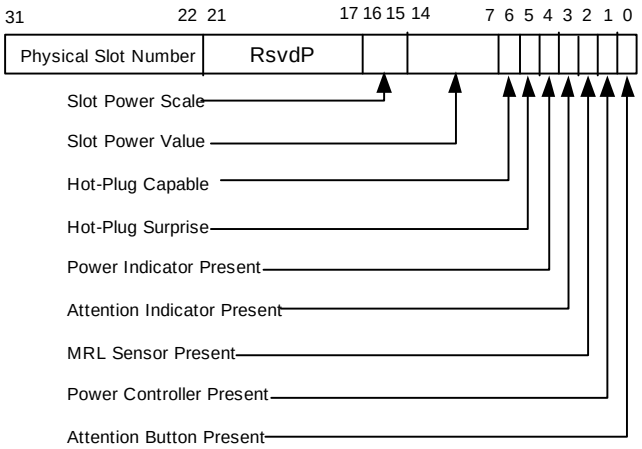

Figure 5-19: Slot Capabilities Register

Table 5-17: Slot Capabilities Register

Bit Location	Register Description	Attributes
0	**Attention Button Present** – This bit when set indicates that an Attention Button is implemented on the chassis for this slot.	HwInit
1	**Power Controller Present** – This bit when set indicates that a Power Controller is implemented for this slot.	HwInit
2	**MRL Sensor Present** – This bit when set indicates that an MRL Sensor is implemented on the chassis for this slot.	HwInit
3	**Attention Indicator Present** – This bit when set indicates that an Attention Indicator is implemented on the chassis for this slot.	HwInit
4	**Power Indicator Present** – This bit when set indicates that a Power Indicator is implemented on the chassis for this slot.	HwInit
5	**Hot-plug Surprise** – This bit when set indicates that a device present in this slot might be removed from the system without any prior notification.	HwInit

6	**Hot-plug Capable** – This bit when set indicates that this slot is capable of supporting Hot-plug operations.	HwInit
14:7	**Slot Power Limit Value** – In combination with the Slot Power Limit Scale value, specifies the upper limit on power supplied by slot. Power limit (in Watts) calculated by multiplying the value in this field by the value in the Slot Power Limit Scale field. This register must be implemented if the Slot Implemented bit is set. The default value is 0000 0000b.	HwInit
16:15	**Slot Power Limit Scale –** Specifies the scale used for the Slot Power Limit Value. Range of Values 00b = 1.0x (25.5-255) 01b = 0.1x (2.55-25.5) 10b = 0.01x (0.255-2.55) 11b = 0.001x (0.0-0.255) This register must be implemented if the Slot Implemented bit is set. The default value is all 00b.	HwInit
31:22	**Physical Slot Number** – This hardware initialized field indicates the physical slot number attached to this Port. This field must be hardware initialized to a value that assigns a slot number that is globally unique within the chassis. These registers should be initialized to 0 for ports connected to devices that are either integrated on the motherboard or integrated within the same silicon as the Switch device or Root Port.	HwInit

5.8.10. Slot Control Register (Offset 18h)

The Slot Control register controls PCI Express Slot specific parameters. Figure 5-20 details allocation of register fields in the Slot Control register; Table 5-18 provides the respective bit definitions.

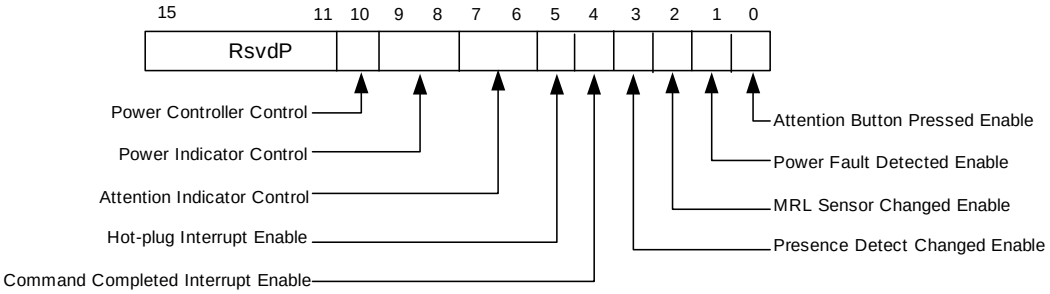

Figure 5-20: Slot Control Register

Table 5-18: Slot Control Register

Bit Location	Register Description	Attributes
0	**Attention Button Pressed Enable –** This bit when set enables the generation of hot plug interrupt or wake message on an attention button pressed event. Default value of this field is 0.	RW
1	**Power Fault Detected Enable –** This bit when set enables the generation of hot plug interrupt or wake message on a power fault event. Default value of this field is 0.	RW
2	**MRL Sensor Changed Enable –** This bit when set enables the generation of hot plug interrupt or wake message on a MRL sensor changed event. Default value of this field is 0.	RW
3	**Presence Detect Changed Enable –** This bit when set enables the generation of hot plug interrupt or wake message on a presence detect changed event. Default value of this field is 0.	RW
4	**Command Completed Interrupt Enable –** This bit when set enables the generation of hot plug interrupt when a command is completed by the Hot plug controller. Default value of this field is 0.	RW
5	**Hot plug Interrupt Enable –** This bit when set enables generation of hot plug interrupt on enabled hot plug events. Default value of this field is 0.	RW

<table>
<tr><th>Bit Location</th><th>Register Description</th><th>Attributes</th></tr>
<tr><td>7:6</td><td>Attention Indicator Control – Reads to this register return the current state of the Attention Indicator; writes to this register set the Attention Indicator. Defined encodings are:
<table>
<tr><td>00b</td><td>Reserved</td></tr>
<tr><td>01b</td><td>On</td></tr>
<tr><td>10b</td><td>Blink</td></tr>
<tr><td>11b</td><td>Off</td></tr>
</table>
Write to this register causes the Port to send the appropriate ATTENTION_INDICATOR_* messages.</td><td>RW</td></tr>
<tr><td>9:8</td><td>Power Indicator Control – Reads to this register return the current state of the Power Indicator; writes to this register set the Power Indicator. Defined encodings are:
<table>
<tr><td>00b</td><td>Reserved</td></tr>
<tr><td>01b</td><td>On</td></tr>
<tr><td>10b</td><td>Blink</td></tr>
<tr><td>11b</td><td>Off</td></tr>
</table>
Writes to this register causes the Port to send the appropriate POWER_INDICATOR_* messages.</td><td>RW</td></tr>
<tr><td>10</td><td>Power Controller Control – When read this register returns the current state of the Power applied to the slot; when written sets the power state of the slot per the defined encodings.
<table>
<tr><td>0b</td><td>Power On</td></tr>
<tr><td>1b</td><td>Power Off</td></tr>
</table></td><td>RW</td></tr>
</table>

5.8.11. Slot Status Register (Offset 1Ah)

The Slot Status register provides information about PCI Express Slot specific parameters. Figure 5-21 details allocation of register fields in the Slot Status register; Table 5-19 provides the respective bit definitions.

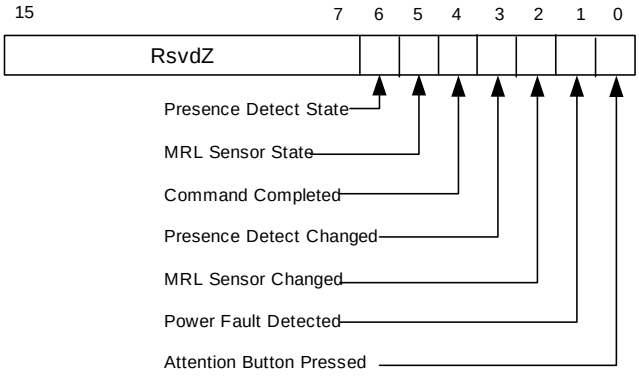

Figure 5-21: Slot Status Register

Table 5-19: Slot Status Register

Bit Location	Register Description	Attributes
0	**Attention Button Pressed**– This bit is set when the attention button is pressed. Default value of this field is 0.	RW1C
1	**Power Fault Detected** – This bit is set when the Power Controller detects a power fault at this slot. Default value of this field is 0.	RW1C
2	**MRL Sensor Changed** – This bit is set when a MRL Sensor state change is detected. Default value of this field is 0.	RW1C
3	**Presence Detect Changed** – This bit is set when a Presence Detect change is detected. Default value of this field is 0.	RW1C
4	**Command Completed –** This bit is set when the hot plug controller completes an issued command. Default value of this field is 0.	RW1C
5	**MRL Sensor State –** This register reports the status of the MRL sensor if it is implemented. Defined encodings are: 0b MRL Closed 1b MRL Open	RO

Bit Location	Register Description	Attributes
6	**Presence Detect State** – This bit indicates the presence of a card in the slot. This bit reflects the status of the Presence Detect pin as defined in the *PCI Express Card Electromechanical Specification*. Defined encodings are: 0b – Slot Empty 1b – Card Present in slot This register is required to be implemented on all Switch Downstream Ports and Root Ports. The Presence Detect State field for Switch Downstream Ports or Root Ports not connected to any slots should be hardwired to 1. This register is required if a slot is implemented.	RO

5.8.12. Root Control Register (Offset 1Ch)

The Root Control register controls PCI Express Root Complex specific parameters. Figure 5-22 details allocation of register fields in the Root Control register; Table 5-20 provides the respective bit definitions.

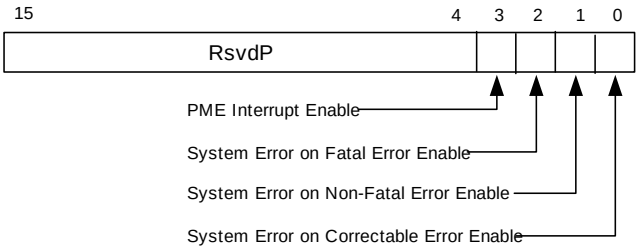

Figure 5-22: Root Control Register

Table 5-20: Root Control Register

Bit Location	Register Description	Attributes
0	**System Error on Correctable Error Enable** – This bit controls the Root Complex's response to correctable errors. If set it indicates that a System Error should be generated if a correctable error is reported by any of the devices in the hierarchy associated with this Root Port.	RW
1	**System Error on Non-Fatal Error Enable** – This bit controls the Root Complex's response to non-fatal errors. If set it indicates that a System Error should be generated if a non-fatal error is reported by any of the devices in the hierarchy associated with this Root Port.	RW
2	**System Error on Fatal Error Enable** – This bit controls the Root Complex's response to fatal errors. If set it indicates that a System Error should be generated if a fatal error is reported by any of the devices in the hierarchy associated with this Root Port.	RW
3	**PME Interrupt Enable** – This bit when set enables interrupt generation upon receipt of a PME message as reflected in the PME Status register bit (see Table 5-21). A PME interrupt is also generated if the PME Status register bit is set when this bit is set from a cleared state. Default value of this field is 0.	RW

5.8.13. Root Status Register (Offset 20h)

The Root Status register provides information about PCI Express device specific parameters. Figure 5-23 details allocation of register fields in the Root Status register; Table 5-21 provides the respective bit definitions.

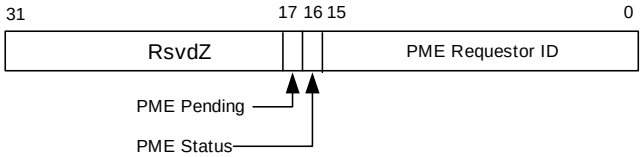

Figure 5-23: Root Status Register

Table 5-21: Root Status Register

Bit Location	Register Description	Attributes
15:0	**PME Requestor ID** – This field indicates the PCI requestor ID of the last PME requestor.	RO
16	**PME Status** – This bit indicates that PME was asserted by the requestor ID indicated in the PME Requestor ID field. Subsequent PMEs are kept pending until the status register is cleared by software by writing a 1.	RW1C
17	**PME Pending** – This read-only bit indicates that another PME is pending when the PME Status bit is set. When the PME Status bit is cleared by software; the PME is delivered by hardware by setting the PME Status bit again and updating the Requestor ID appropriately. The PME pending bit is cleared by hardware if no more PMEs are pending.	RO

5.9. PCI Express Extended Capabilities

PCI Express Extended Capability registers are located in device configuration space at offsets 256 or greater as shown in Figure 5-24 or in the Root Complex Register Block (RCRB). These registers when located in the device configuration space are accessible using only the PCI Express extended configuration space flat memory-mapped access mechanism.

PCI Express Extended Capability structures are allocated using a linked list of optional or required PCI Express Extended Capabilities following a format resembling PCI capability structures. The first DWORD of the capability structure identifies the capability/version and points to the next capability as shown in Figure 5-24.

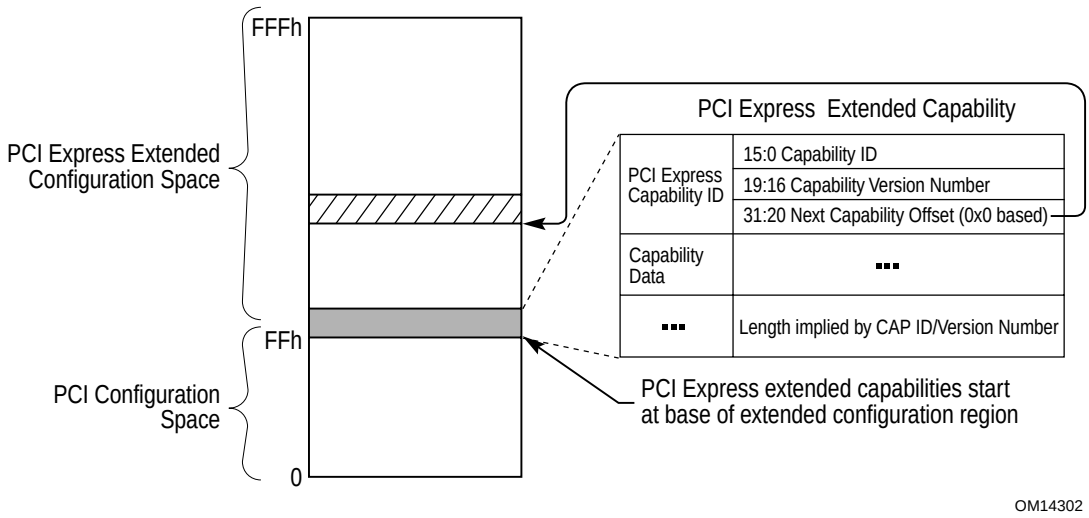

Figure 5-24: PCI Express Extended Configuration Space Layout

5.9.1. Extended Capabilities in Configuration Space

Extended Capabilities in device configuration space always begin at offset 100h with a PCI Express Enhanced Capability Header (Section 5.9.3). Absence of any Extended Capabilities is required to be indicated by an Enhanced Capability Header with a Capability ID of FFFFh and a Next Capability Offset of 0h.

5.9.2. Extended Capabilities in the Root Complex Register Block

Extended Capabilities in a Root Complex Register Block always begin at offset 0h with a PCI Express Enhanced Capability Header (Section 5.9.3). Absence of any Extended Capabilities is required to be indicated by an Enhanced Capability Header with a Capability ID of FFFFh and a Next Capability Offset of 0h.

5.9.3. PCI Express Enhanced Capability Header

All PCI Express extended capabilities must begin with a PCI Express Enhanced Capability Header.

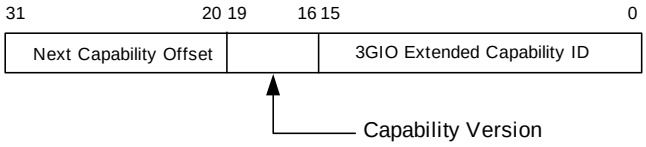

Figure 5-25: PCI Express Enhanced Capability Header

Table 5-22: PCI Express Enhanced Capability Header

Bit Location	Description	Register Attribute
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the extended capability.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the capability structure present.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express capability structure or 000h if no other items exist in the linked list of capabilities. For Extended Capabilities implemented in device configuration space, this offset is relative to the beginning of PCI compatible configuration space and thus must always be either 000h (for terminating list of capabilities) or greater than 0FFh.	RO

5.10. Advanced Error Reporting Capability

The PCI Express Advanced Error Reporting capability is an optional extended capability that may be implemented by PCI Express devices supporting advanced error control and reporting. The Advanced Error Reporting capability structure definition has additional interpretation for Root Ports; software must interpret the PCI Express device/Port Type field (Section 5.8.1) in the PCI Express Capability Structure to determine the availability of additional registers for Root Ports.

Figure 5-26 shows the PCI Express Advanced Error Reporting Capability Structure.

Note that if an error reporting bit field is marked as optional in the error registers, the bits must be implemented or not implemented as a group across the Status, Mask and Severity registers. In other words, a device is required to implement the same error bit fields in corresponding Status, Mask and Severity registers. Bits corresponding to bit fields that are not implemented must be hardwired to 0.

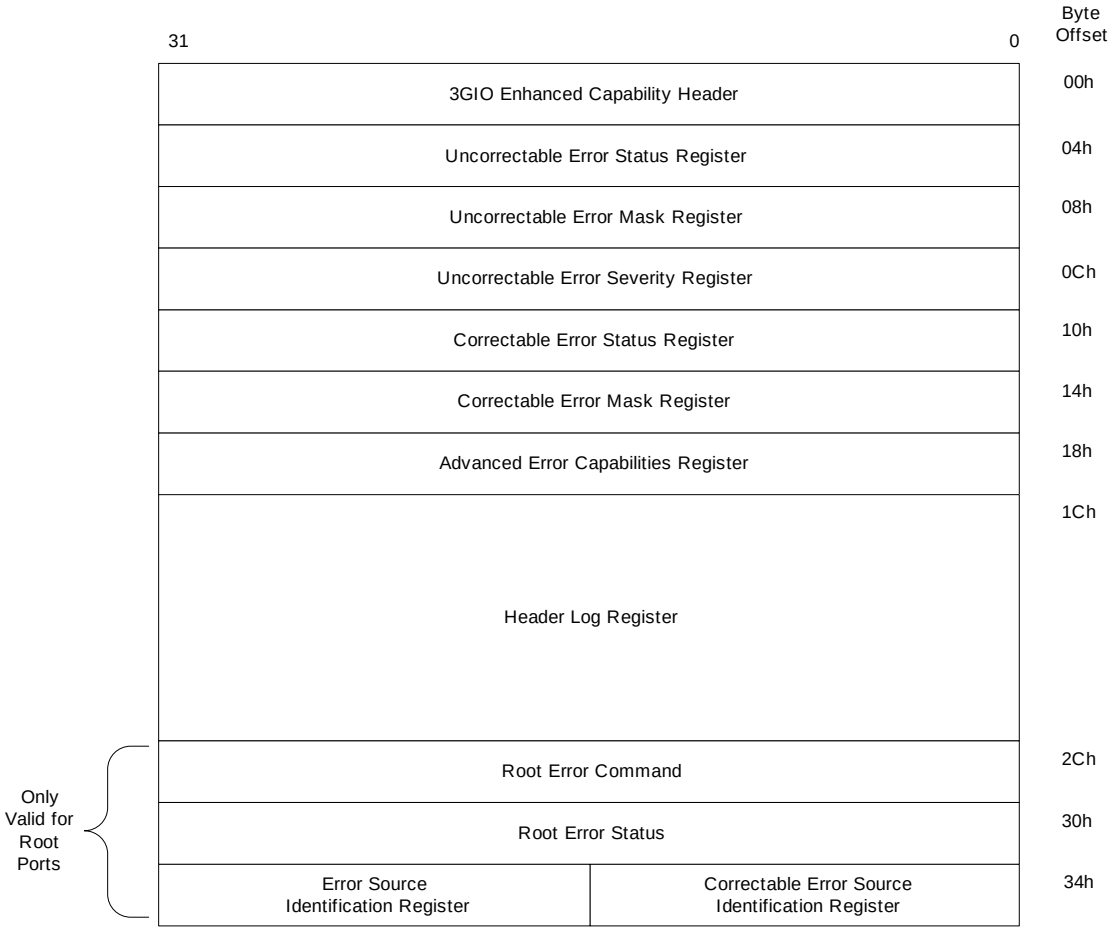

Figure 5-26: PCI Express Advanced Error Reporting Extended Capability Structure

5.10.1. Advanced Error Reporting Enhanced Capability Header (Offset 00h)

See Section 5.9.3 for a description of the PCI Express Enhanced Capability Header. The Extended Capability ID for the Advanced Error Reporting Capability is 0001h.

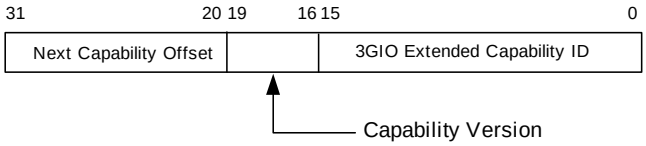

Figure 5-27: Advanced Error Reporting Enhanced Capability Header

Table 5-23: Advanced Error Reporting Enhanced Capability Header

Bit Location	Description	Register Attribute
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the extended capability. Extended Capability ID for the Advanced Error Reporting Capability is 0001h.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the capability structure present. Must be 1h for this version of the specification.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express capability structure or 000h if no other items exist in the linked list of capabilities. For Extended Capabilities implemented in device configuration space, this offset is relative to the beginning of PCI compatible configuration space and thus must always be either 000h (for terminating list of capabilities) or greater than 0FFh.	RO

5.10.2. Uncorrectable Error Status Register (Offset 04h)

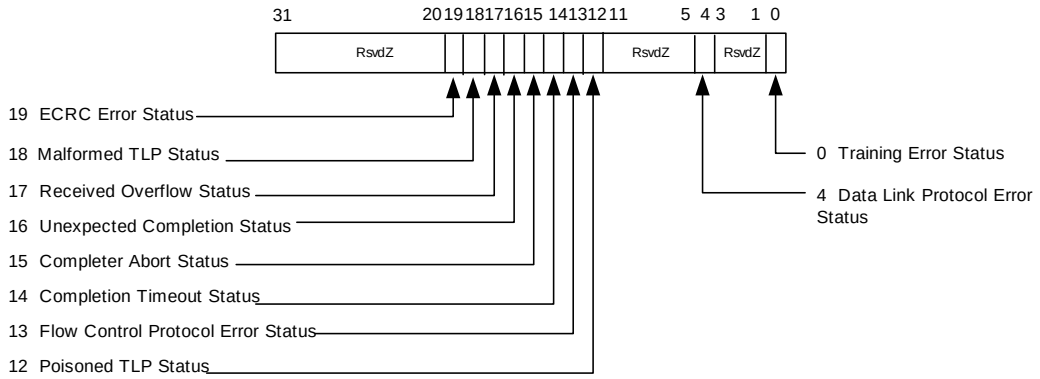

Figure 5-28: Uncorrectable Error Status Register

The Uncorrectable Error Status register reports error status of individual error sources on a PCI Express device. An individual error status bit that is set indicates that a particular error occurred; software may clear an error status by writing a 1 to the respective bit. Refer to Section 7.2 for further details.

Table 5-24: Uncorrectable Error Status Register

Bit Location	Description	Register Attribute	Default Value
0	**Training Error Status** (Optional)	RW1CS	0
4	**Data Link Protocol Error Status**	RW1CS	0
12	**Poisoned TLP Status**	RW1CS	0
13	**Flow Control Protocol Error Status** (Optional)	RW1CS	0
14	**Completion Timeout Status**	RW1CS	0
15	**Completer Abort Status** (Optional)	RW1CS	0
16	**Unexpected Completion Status**	RW1CS	0
17	**Receiver Overflow Status** (Optional)	RW1CS	0
18	**Malformed TLP Status**	RW1CS	0
19	**ECRC Error Status** (Optional)	RW1CS	0

5.10.3. Uncorrectable Error Mask Register (Offset 08h)

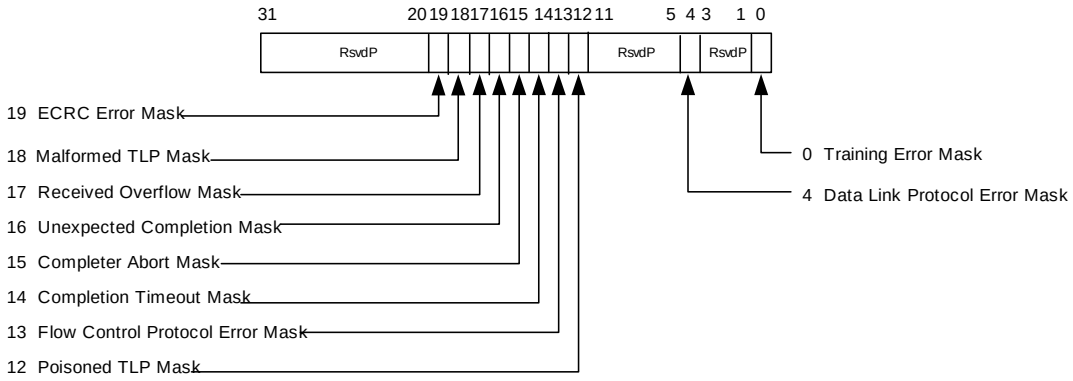

Figure 5-29: Uncorrectable Error Mask Register

The Uncorrectable Error Mask register controls reporting of individual errors by device to the PCI Express Root Complex via a PCI Express error message. A masked error (respective bit set in mask register) is not reported to the PCI Express Root Complex by an individual device. Refer to Section 7.2 for further details. There is a mask bit per bit of the Uncorrectable Error Status register.

Table 5-25: Uncorrectable Error Mask Register

Bit Location	Description	Register Attribute	Default Value
0	**Training Error Mask** (Optional)	RWS	0
4	**Data Link Protocol Error Mask**	RWS	0
12	**Poisoned TLP Mask**	RWS	0
13	**Flow Control Protocol Error Mask** (Optional)	RWS	0
14	**Completion Timeout Mask**	RWS	0
15	**Completer Abort Mask** (Optional)	RWS	0
16	**Unexpected Completion Mask**	RWS	0
17	**Receiver Overflow Mask** (Optional)	RWS	0
18	**Malformed TLP Mask**	RWS	0
19	**ECRC Error Mask** (Optional)	RWS	0

5.10.4. Uncorrectable Error Severity Register (Offset 0Ch)

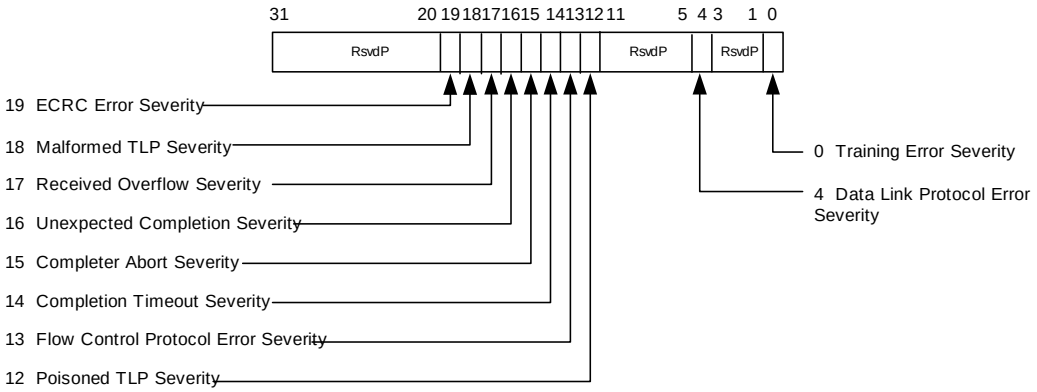

Figure 5-30: Uncorrectable Error Severity Register

The Uncorrectable Error Severity register controls whether an individual error is reported as a non-fatal or fatal error. An error is reported as fatal when the corresponding error bit in the severity register is set. If the bit is cleared, the corresponding error is considered non-fatal. Refer to Section 7.2 for further details.

Table 5-26: Uncorrectable Error Severity Register

Bit Location	Description	Register Attribute	Default Value
0	**Training Error Severity** (Optional)	RWS	1
4	**Data Link Protocol Error Severity**	RWS	1
12	**Poisoned TLP Severity**	RWS	0
13	**Flow Control Protocol Error Severity** (Optional)	RWS	0
14	**Completion Timeout Error Severity**	RWS	0
15	**Completer Abort Error Severity** (Optional)	RWS	0
16	**Unexpected Completion Error Severity**	RWS	0
17	**Receiver Overflow Error Severity** (Optional)	RWS	1
18	**Malformed TLP Severity**	RWS	1
19	**ECRC Error Severity** (Optional)	RWS	0

5.10.5. Correctable Error Status Register (Offset 10h)

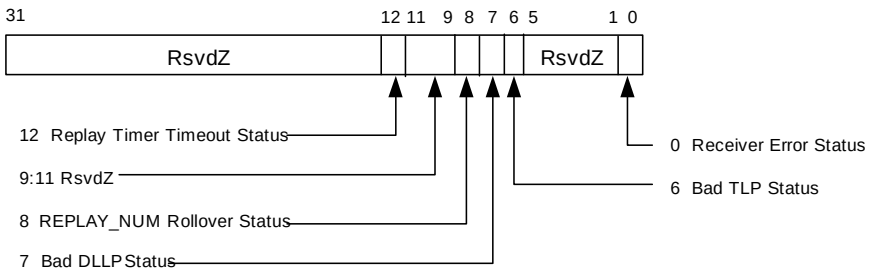

Figure 5-31: Correctable Error Status Register

The Correctable Error Status register reports error status of individual correctable error sources on a PCI Express device. When an individual error status bit is set, it indicates that a particular error occurred; software may clear an error status by writing a 1 to the respective bit. Refer to Section 7.2 for further details.

Table 5-27: Correctable Error Status Register

Bit Location	Description	Register Attribute	Default Value
0	**Receiver Error Status** (Optional)	RW1CS	0
6	**Bad TLP Status**	RW1CS	0
7	**Bad DLLP Status**	RW1CS	0
8	**REPLAY_NUM Rollover Status**	RW1CS	0
12	**Replay Timer Timeout Status**	RW1CS	0

5.10.6. Correctable Error Mask (Offset 14h)

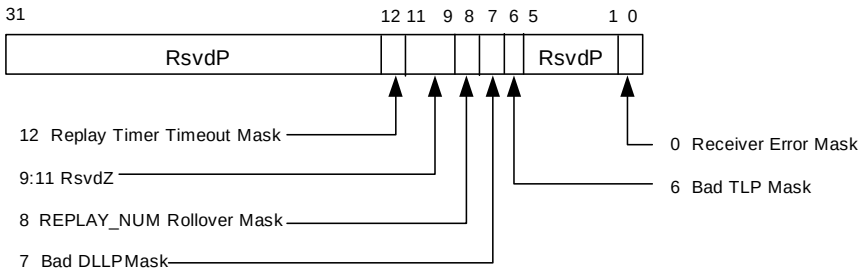

Figure 5-32: Correctable Error Mask Register

The Correctable Error Mask register controls reporting of individual correctable errors by device to the PCI Express Root Complex via a PCI Express error message. A masked error (respective bit set in mask register) is not reported to the PCI Express Root Complex by an

individual device. Refer to Section 7.2 for further details. There is a mask bit per bit in the Correctable Error Status register.

Table 5-28: Correctable Error Mask Register

Bit Location	Description	Register Attribute	Default Value
0	**Receiver Error Mask** (Optional)	RWS	0
6	**Bad TLP Mask**	RWS	0
7	**Bad DLLP Mask**	RWS	0
8	**REPLAY_NUM Rollover Mask**	RWS	0
12	**Replay Timer Timeout Mask**	RWS	0

5.10.7. Advanced Error Capabilities and Control Register (Offset 18h)

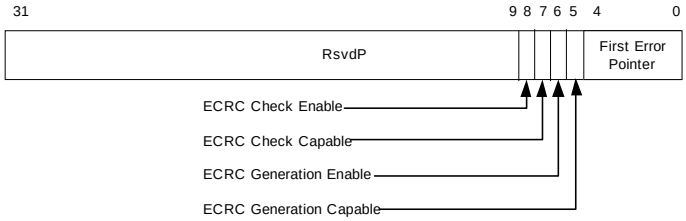

Figure 5-33: Advanced Error Capabilities and Control Register

Figure 5-33 details allocation of register fields in the Advanced Error Capabilities and Control register; Table 5-29 provides the respective bit definitions. Handling of multiple errors is discussed in Section 7.2.4.2.

Table 5-29: Advanced Error Capabilities Register

Bit Location	Description	Register Attribute
4:0	**First Error Pointer -** The First Error Pointer is a read-only register that identifies the bit position of the first error reported in the Uncorrectable Error Status register. Refer to Section 7.2 for further details	ROS
5	**ECRC Generation Capable –** This bit indicates that the device is capable of generating ECRC (see Section 2.10).	RO
6	**ECRC Generation Enable –** This bit when set enables ECRC generation (see Section 2.10).	RWS

Bit Location	Description	Register Attribute
	Default value of this field is 0.	
7	**ECRC Check Capable –** This bit indicates that the device is capable of checking ECRC (see Section 2.10).	RO
8	**ECRC Check Enable –** This bit when set enables ECRC checking (see Section 2.10). Default value of this field is 0.	RWS

5.10.8. Header Log Register (Offset 1Ch)

The header log register captures the header for the transaction that generated an error; refer to Section 7.2 for further details. Section 7.2 also enumerates the conditions where the packet header is logged. This register is 16 bytes and adheres to the format of the headers defined throughout this specification.

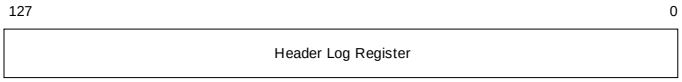

Figure 5-34: Header Log Register

Table 5-30: Header Log Register

Bit Location	Description	Register Attribute	Default Value
127:0	Header of TLP associated with error	ROS	0

5.10.9. Root Error Command Register (Offset 2Ch)

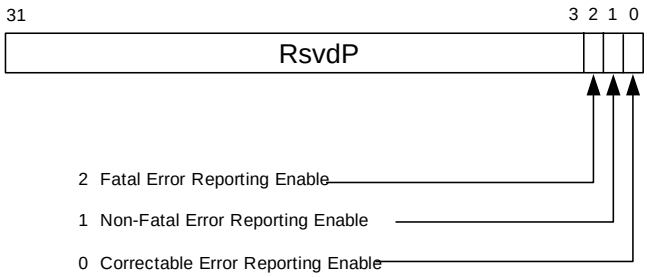

Figure 5-35: Root Error Command Register

The Root Error Command register allows finer 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/disable generation of interrupts (claimed by the Root Port) in addition to system error messages according to the definitions in Table 5-31.

Table 5-31: Root Error Command Register

Bit Location	Description	Register Attribute	Default Value
0	**Correctable Error Reporting Enable** – When set this bit enables the generation of an interrupt when a correctable error is reported by any of the devices in the hierarchy associated with this Root Port. Refer to Section 7.2 for further details.	RW	0
1	**Non-Fatal Error Reporting Enable** – When set this bit enables the generation of an interrupt when a non-fatal error is reported by any of the devices in the hierarchy associated with this Root Port. Refer to Section 7.2 for further details.	RW	0
2	**Fatal Error Reporting Enable** – When set this bit enables the generation of an interrupt when a fatal error is reported by any of the devices in the hierarchy associated with this Root Port. Refer to Section 7.2 for further details.	RW	0

System error generation in response to PCI Express error messages may be turned off by system software using the PCI Express Capability Structure described in Section 5.8 when advanced error reporting via interrupts is enabled. Refer to Section 7.2 for further details.

5.10.10. Root Error Status Register (Offset 30h)

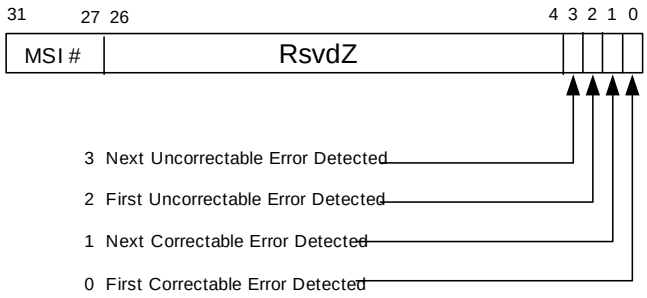

Figure 5-36: Root Error Status Register

The Root Error Status register reports status of errors received by the root complex. 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 Requestor ID is logged in the Error Source Identification register. A set individual error status bit indicates that a particular error occurred; software may clear an error status by writing a 1 to the respective bit. If software does not clear the first reported error before another error message is received, the next error status bit will be set but the Requestor ID of the subsequent error message is discarded. The next error status bits may be cleared by software by writing a 1 to the respective bit as well. Refer to Section 7.2 for further details.

Table 5-32: Root Error Status Register

Bit Location	Description	Register Attribute
0	**First Correctable Error Detected** – Set when a correctable error is received and First Correctable Error Detected is not already set. Default value of this field is 0.	RW1CS
1	**Next Correctable Error Detected** – Set when a correctable error is received and First Correctable Error Detected is already set. This indicates that one or more error message requestor IDs were lost. Default value of this field is 0.	RW1CS
2	**First Uncorrectable Error Detected** – Set when either a fatal or a non-fatal error is received and First Uncorrectable Error Detected is not already set. Default value of this field is 0.	RW1CS

Bit Location	Description	Register Attribute
3	**Next Uncorrectable Error Detected** – Set when either a fatal or a non-fatal error is received and First Uncorrectable Error Detected is already set. This indicates that one or more error message requestor IDs were lost. Default value of this field is 0.	RW1CS
31:27	**Advanced Error Interrupt Message Number –** If this function is allocated more than one MSI interrupt number, this register is required to contain the offset between the base Message Data and the MSI Message that is generated when any of status bits of this capability are set. Hardware is required to update this field so that it is correct if the number of MSI Messages assigned to the device changes.	RO

5.10.11. Error Source Identification Register (Offset 34h)

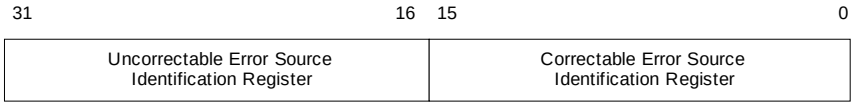

Figure 5-37: Error Source Identification Register

The Error Source identification register identifies the source (Requestor ID) of first correctable and non-fatal/fatal errors reported in the Root Error Status register. Refer to Section 7.2 for further details.

Table 5-33: Error Source Identification Register

Bit Location	Description	Register Attribute
15:0	**Correctable Error Source Identification** – Set with the Requestor ID of the source when a correctable error is received and First Correctable Error Detected is not already set. Default value of this field is 0.	ROS
31:16	**Uncorrectable Error Source Identification** – Set with the Requestor ID of the source when a non-fatal/fatal error is received and the First Uncorrectable Error Detected is not already set. Default value of this field is 0.	ROS

5.11. Virtual Channel Capability

The PCI Express Virtual Channel capability is an optional extended capability that is required to be implemented by PCI Express ports of devices that support PCI Express functionality beyond the general purpose IO traffic, i.e. the default Traffic Class 0 (TC0) over the default Virtual Channel 0 (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 5-38 provides a high level view of the PCI Express Virtual Channel Capability Structure for all devices. This structure controls Virtual Channel assignment for PCI Express links and may be present in Endpoint devices, Switch ports (Upstream and Downstream), Root Ports and RCRB. Some registers/fields in the PCI Express Virtual Channel Capability Structure may have different interpretation for Endpoint devices, Switch ports, Root Ports and RCRB. Software must interpret the PCI Express device/Port Type field (Section 5.8.1) in the PCI Express Capability Structure to determine the availability and meaning of these registers/fields.

The PCI Express Virtual Channel Capability Structure can be present in the Extended Configuration Space of all devices or in RCRB with the restriction that it is only present in the Extended Configuration Space of Function 0 for devices at their Upstream Ports.

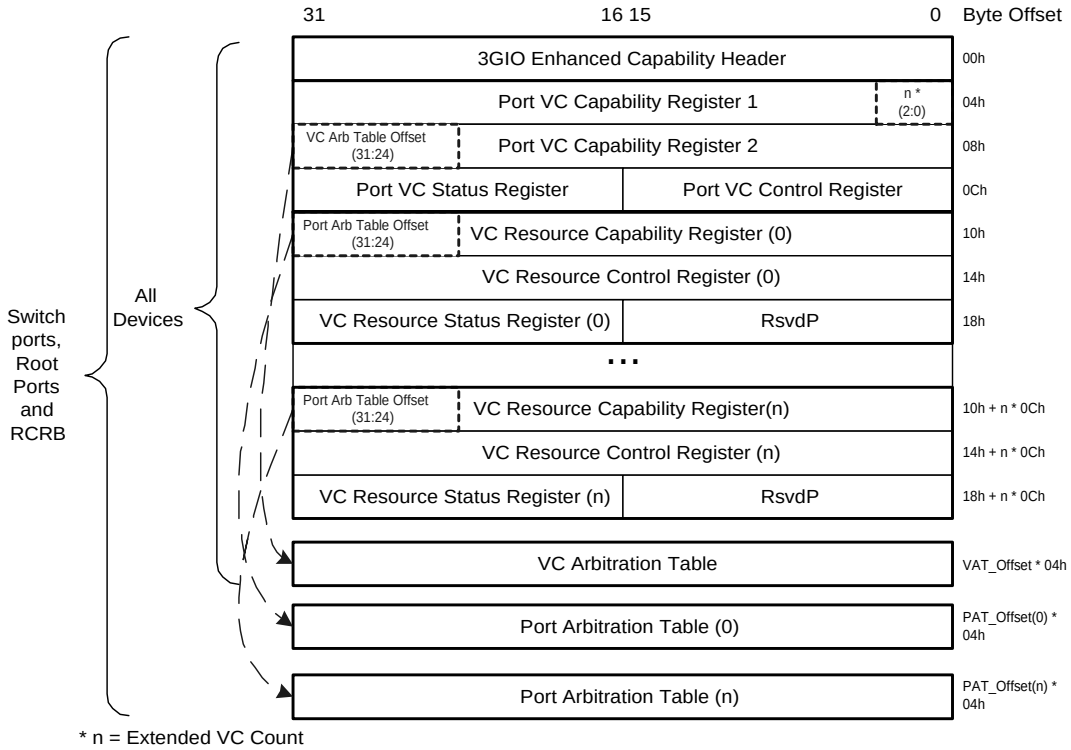

Figure 5-38: PCI Express Virtual Channel Capability Structure

The following registers/fields are defined for PCI Express Virtual Channel Capability Structure.

5.11.1. Virtual Channel Enhanced Capability Header

See Section 5.9.3 for a description of the PCI Express Enhanced Capability Header. The Extended Capability ID for the Virtual Channel Capability is 0002h.

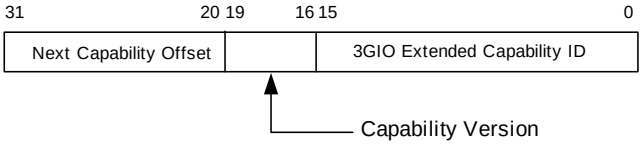

Figure 5-39: Virtual Channel Enhanced Capability Header

Table 5-34: Virtual Channel Enhanced Capability Header

Bit Location	Description	Register Attribute
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the extended capability. Extended Capability ID for the Virtual Channel Capability is 0002h.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the capability structure present. Must be 1h for this version of the specification.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express capability structure or 000h if no other items exist in the linked list of capabilities. For Extended Capabilities implemented in device configuration space, this offset is relative to the beginning of PCI compatible configuration space and thus must always be either 000h (for terminating list of capabilities) or greater than 0FFh.	RO

5.11.2. Port VC Capability Register 1

The Port VC Capability Register 1 describes the configuration of the Virtual Channels associated with a PCI Express port. Figure 5-40 details allocation of register fields in the Port VC Capability Register 1; Table 5-35 provides the respective bit definitions.

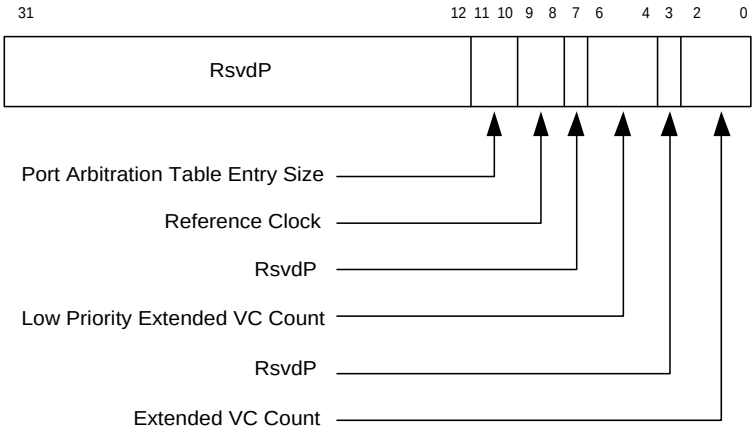

Figure 5-40: Port VC Capability Register 1

Table 5-35: Port VC Capability Register 1

Bit Location	Description	Attribute
2:0	Extended VC Count – Indicates the number of (extended) Virtual Channels in addition to the default VC supported by the device. This field is valid for all devices. The minimum value of this field is 0 (for devices that only support the default VC). The maximum value is 7.	RO
6:4	Low Priority Extended VC Count – Indicates the number of (extended) Virtual Channels in addition to the default VC belonging to the low-priority VC (LPVC) group that has the lowest priority with respect to other VC resources in a strict-priority VC Arbitration. This field is valid for all devices. The minimum value of this field is 0 and the maximum value is Extended VC Count.	RO

<table>
<tr><th>Bit Location</th><th>Description</th><th>Attribute</th></tr>
<tr><td>9:8</td><td>Reference Clock – Indicates the reference clock for Virtual Channels that support time-based WRR Port Arbitration. This field is valid only for RCRB and Switch Upstream Ports. This field is not valid and must be set to 0 for Endpoint devices, Root Ports or Switch Downstream Ports. Defined encodings are:
<table>
<tr><td>00b</td><td>100 ns reference clock</td></tr>
<tr><td>01b – 11b</td><td>Reserved</td></tr>
</table>
Note: Time-based WRR Port Arbitration can be supported by multiple Switch ports when they serve as egress for peer-to-peer traffic. However, only the Upstream Port of a Switch contains valid Reference Clock.</td><td>RO</td></tr>
<tr><td>11:10</td><td>Port Arbitration Table Entry Size – Indicates the size (in bits) of Port Arbitration table entry in the device. This field is valid only for RCRB and Switch Upstream Ports. It is not valid and must be set to 0 for Endpoint devices, Root Ports or Switch Downstream Ports. Defined encodings are:
<table>
<tr><td>00b</td><td>The size of Port Arbitration table entry is 1 bit</td></tr>
<tr><td>01b</td><td>The size of Port Arbitration table entry is 2 bits</td></tr>
<tr><td>10b</td><td>The size of Port Arbitration table entry is 4 bits</td></tr>
<tr><td>11b</td><td>The size of Port Arbitration table entry is 8 bits</td></tr>
</table></td><td>RO</td></tr>
</table>

5.11.3. Port VC Capability Register 2

The Port VC Capability Register 2 provides further information about the configuration of the Virtual Channels associated with a PCI Express port. Figure 5-41 details allocation of register fields in the Port VC Capability Register 2; Table 5-36 provides the respective bit definitions.

31 24	23 8	7 0
VC Arbitration Table Offset	RsvdP	VC Arbitration Capability

Figure 5-41: Port VC Capability Register 2

Table 5-36: Port VC Capability Register 2

Bit Location	Description	Attribute
7:0	**VC Arbitration Capability –** Indicates the types of VC Arbitration supported by the device for the LPVC group. This field is valid for all devices that report a Low Priority Extended VC Count greater than 0. Each bit location within this field corresponds to a VC Arbitration capability defined below. When more than one bit in this field is set, it indicates that the port can be configured to provide different VC arbitration services. Defined bit positions are: Bit 0: Hardware fixed Round-Robin (RR) or RR-like arbitration scheme 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 devices. This field contains the zero-based offset of the table in DQWORDS (16 bytes) from the base address of the Virtual Channel Capability Structure. A value of 0 indicates that the table is not present.	RO

5.11.4. Port VC Control Register

Figure 5-42 details allocation of register fields in the Port VC Control Register; Table 5-37 provides the respective bit definitions.

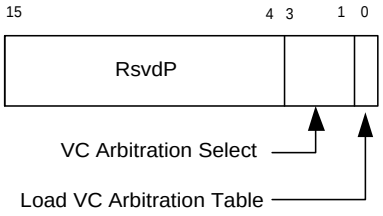

Figure 5-42: Port VC Control Register

Table 5-37: Port VC Control Register

Bit Location	Description	Attribute
0	**Load VC Arbitration Table** – Used for software to update the VC Arbitration Table. This field is valid for all devices when the VC Arbitration Table is used by the selected VC Arbitration. Software sets this bit to request hardware to apply new values programmed into VC Arbitration Table; clearing this bit has no effect. Software checks the VC Arbitration Table Status field to confirm that new values stored in the VC Arbitration Table are latched by the VC arbitration logic. This bit always returns 0 when read.	RW
3:1	**VC Arbitration Select** – Used for software to configure the VC arbitration by selecting one of the supported VC Arbitration schemes indicated by the VC Arbitration Capability field in the Port VC Capability Register 2. This field is valid for all devices. The value of this field is the number corresponding to one of the asserted bits in the VC Arbitration Capability field. This field can not be modified when more than one VC in the LPVC group is enabled.	RW

5.11.5. Port VC Status Register

The Port VC Status Register provides status of the configuration of Virtual Channels associated with a port. Figure 5-43 details allocation of register fields in the Port VC Status Register; Table 5-38 provides the respective bit definitions.

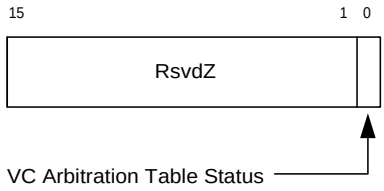

Figure 5-43: Port VC Status Register

Table 5-38: Port VC Status Register

Bit Location	Description	Attribute
0	**VC Arbitration Table Status** – Indicates the coherency status of the VC Arbitration Table. This field is valid for all devices when the VC Arbitration Table is used by the selected VC Arbitration. This bit is set by hardware when any entry of the VC Arbitration Table is written by software. This bit is cleared by hardware when hardware finishes loading values stored in the VC Arbitration Table after software sets the Load VC Arbitration Table field in the Port VC Control Register. Default value of this field is 0.	RO

5.11.6. VC Resource Capability Register

The VC Resource Capability Register describes the capabilities and configuration of a particular Virtual Channel resource. Figure 5-44 details allocation of register fields in the VC Resource Capability Register; Table 5-39 provides the respective bit definitions.

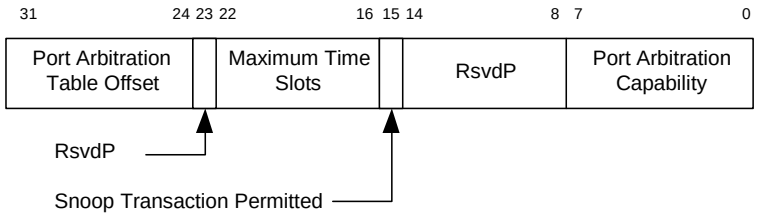

Figure 5-44: VC Resource Capability Register

Table 5-39: VC Resource Capability Register

<table>
<tr><th>Bit Location</th><th>Description</th><th>Attribute</th></tr>
<tr><td>7:0</td><td>Port Arbitration Capability – Indicates types of Port Arbitration supported by the VC resource. This field is valid for all Switch ports and RCRB, but not for PCI Express Endpoint devices or Root Ports.

Each bit location within this field corresponds to a Port Arbitration capability defined below. When more than one bit in this field is set, it indicates that the VC resource can be configured to provide different arbitration services.

Software selects among these capabilities by writing to the Port Arbitration Select field (see below). Defined bit positions are:
<table>
<tr><td>Bit 0</td><td>Hardware fixed Round-Robin (RR) or RR-like arbitration scheme</td></tr>
<tr><td>Bit 1</td><td>Weighted Round Robin (WRR) arbitration with 32 phases</td></tr>
<tr><td>Bit 2</td><td>WRR arbitration with 64 phases</td></tr>
<tr><td>Bit 3</td><td>WRR arbitration with 128 phases</td></tr>
<tr><td>Bit 4</td><td>Time-based WRR with 128 phases</td></tr>
<tr><td>Bits 5-7</td><td>Reserved</td></tr>
</table></td><td>RO</td></tr>
<tr><td>15</td><td>Snoop Transaction Permitted – Indicates if snoop transaction is permitted over the VC resource. This field is valid only for RCRB, but not for PCI Express Endpoint devices, Switch ports or Root Ports.

When this field is set, it indicates that the Root Complex is able to honor the "Snoop Not Required" Attribute field in the TLP header and ensure cache coherency for the transactions over the VC resource. When this field is set to 0, it indicates that the Root Complex ignores the "Snoop Not Required" Attribute field in the TLP header and does not perform Snoop operation for transactions over the VC resource.</td><td>HwInit</td></tr>
<tr><td>22:16</td><td>Maximum Time Slots – Indicates the maximum number of time slots (minus one) that the VC resource is capable of supporting when it is configured for time-based WRR Port Arbitration. For example, a value 0 in this field indicates the supported maximum number of time slots is 1 and a value of 127 indicates the supported maximum number of time slot is 128. This field is valid for all Switch ports, Root Ports and RCRB, but not for PCI Express Endpoint devices. In addition, this field is valid only when Port Arbitration Capability indicates that the VC resource supports time-based WRR Port Arbitration.</td><td>HwIniit</td></tr>
</table>

31:24	**Port Arbitration Table Offset** – Indicates the location of the Port Arbitration Table associated with the VC resource. This field is valid for all Switch ports and RCRB, but not for PCI Express Endpoint devices or Root Ports. This field contains the zero-based offset of the table in DQWORDS (16 bytes) from the base address of the Virtual Channel Capability Structure. A value of 0 indicates that the table is not present.	RO

5.11.7. VC Resource Control Register

Figure 5-45 details allocation of register fields in the VC Resource Control Register; Table 5-40 provides the respective bit definitions.

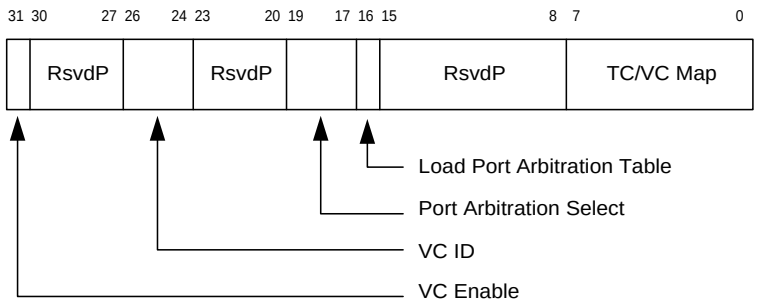

Figure 5-45: VC Resource Control Register

Table 5-40: VC Resource Control Register

Bit Location	Description	Attribute
7:0	**TC/VC Map** – This field indicates the TCs that are mapped to the VC resource. This field is valid for all devices. Bit locations within this field correspond to TC values. For example, when bit 7 is set in this field, TC7 is mapped to this VC resource. When more than one bit in this field is set, it indicates that multiple TCs are mapped to the VC resource. In order to remove one or more TCs from the TC/VC Map of an enabled VC, software must ensure that no new or outstanding transactions with the TC labels are targeted at the given Link. Default value of this field is FFh for the first VC resource and is 00h for other VC resources. Note: Bit 0 of this field is read-only. It must be set by hardware ('hard-wired') for the first VC resource (default VC) and cleared for other VC resources when present.	RW (see the note for exceptions)

Bit Location	Description	Attribute
16	**Load Port Arbitration Table –** This bit, when set, updates the Port Arbitration logic from the Port Arbitration Table for the VC resource. This field is valid for all Switch ports and RCRB, but not for PCI Express Endpoint devices or Root Ports. In addition, this field is only valid when the Port Arbitration Table is used by the selected Port Arbitration scheme (that is indicated by a set bit in the Port Arbitration Capability field selected by Port Arbitration Select). Software sets this bit to signal hardware to update Port Arbitration logic with new values stored in Port Arbitration Table; clearing this bit has no effect. Software uses the Port Arbitration Table Status bit to confirm whether the new values of Port Arbitration Table are completely latched by the arbitration logic. This bit always returns 0 when read. Default value of this field is 0.	RW
19:17	**Port Arbitration Select –** This field configures the VC resource to provide a particular Port Arbitration service. This field is valid only for RCRB, but not for PCI Express Endpoint devices, Switch Ports or Root Ports. 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. This field can not be modified when the VC is already enabled.	RW
26:24	**VC ID** – This field assigns a VC ID to the VC resource (see note for exceptions). This field is valid for all devices. This field can not be modified when the VC is already enabled. Note: For the first VC resource (default VC), this field is a read-only field that must be set to 0 ('hard-wired').	RW

Bit Location	Description	Attribute
31	**VC Enable –** This field, when set, enables a Virtual Channel (see note 1 for exceptions). The Virtual Channel is disabled when this field is cleared. This field is valid for all devices. Software must use the VC Negotiation Pending bit to check whether the VC negotiation is complete. When VC Negotiation Pending bit is cleared, a 1 read from this VC Enable bit indicates that the VC is enabled (Flow Control Initialization is completed for the PCI Express port); a 0 read from this bit indicates that the Virtual Channel is currently disabled. Default value of this field is 1 for the first VC resource and is 0 for other VC resource(s). Notes 1. This bit is hardwired to 1 for the default VC (VC0), i.e., writing to this field has no effect for VC0. 2. To enable a Virtual Channel, the VC Enable bits for that Virtual Channel must be set in both components on a Link. 3. To disable a Virtual Channel, the VC Enable bits for that Virtual Channel must be cleared in both components on a Link. 4. Software must ensure that no traffic is using a Virtual Channel at the time it is disabled. 5. Software must fully disable a Virtual Channel in both components on a Link before re-enabling the Virtual Channel.	RW

5.11.8. VC Resource Status Register

Figure 5-46 details allocation of register fields in the VC Resource Status Register; Table 5-41 provides the respective bit definitions.

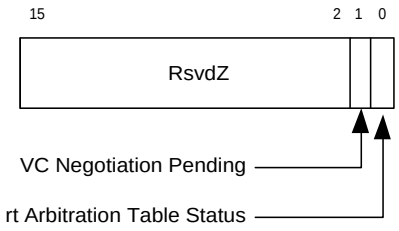

Figure 5-46: VC Resource Status Register

Table 5-41: VC Resource Status Register

Bit Location	Description	Attribute
0	**Port Arbitration Table Status** – This bit indicates the coherency status of the Port Arbitration Table associated with the VC resource. This field is valid for RCRB, but not for PCI Express Endpoint devices, Switch Ports or Root Ports. In addition, this field is valid only when the Port Arbitration Table is used by the selected Port Arbitration for the VC resource. This bit is set by hardware when any entry of the Port Arbitration Table is written to by software. This bit is cleared by hardware when hardware finishes loading values stored in the Port Arbitration Table after software sets the Load Port Arbitration Table field. Default value of this field is 0.	RO
1	**VC Negotiation Pending** –This bit indicates whether the Virtual Channel negotiation (initialization or disabling) is in pending state. This field is valid for all devices. When this bit is set by hardware, it indicates that the VC resource is still in the process of negotiation. This bit is cleared by hardware after the VC negotiation is complete. For a non-default Virtual Channel, software may use this bit when enabling or disabling the VC. For the default VC, this bit indicates the status of the process of Flow Control initialization. Before using a Virtual Channel, software must check whether the VC Negotiation Pending fields for that Virtual Channel are cleared in both components on a Link.	RO

5.11.9. VC Arbitration Table

The VC Arbitration Table is a read-write register array that is used to store the arbitration table for VC Arbitration. This field is valid for all devices when a WRR table is used by the selected VC Arbitration. 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 5-47 depicts the table structure of an example VC Arbitration Table with 32-phases. Each 4-bit table entry corresponds to a phase within a WRR arbitration period. The definition of table entry is depicted in Table 5-42. The lower three bits (bit 0 to bit 2) contain the VC ID value, indicating that the corresponding phase within the WRR arbitration period is assigned to the Virtual Channel indicated by the VC ID.

A phase containing a VC ID that does not correspond to any enabled VCs is simply skipped in the WRR arbitration.

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 5-43.

When the VC Arbitration Table is used by the default VC Arbitration method, the default values of the table entries must be all zero to ensure forward progress for the default VC (with VC ID of 0).

31 28						7 4	3 0	Byte Location
Phase[7]	…	…	…	…		Phase[1]	Phase[0]	00h
Phase[15]	…	…	…	…		Phase[9]	Phase[8]	04h
Phase[23]	…	…	…	…		Phase[17]	Phase[16]	08h
Phase[31]	…	…	…	…	…	Phase[25]	Phase[24]	0Ch

Figure 5-47: Structure of an Example VC Arbitration Table with 32-Phases.

Table 5-42: Definition of the 4-bit Entries in the VC Arbitration Table

Bit Location	Description	Attribute
2:0	VC ID	RW
3	Reserved	RW

Table 5-43 Length of the VC Arbitration Table

VC Arbitration Select	VC Arbitration Table Length (in # of Entries)
001b	32
010b	64
011b	128

5.11.10. Port Arbitration Table

The Port Arbitration Table register is a read-write register array that is used to store the WRR arbitration table for Port Arbitration for the VC resource. This register array is valid for all Switch ports and RCRB, but not for Endpoint devices or Root Ports. It is only present when one or more asserted bits in the Port Arbitration Capability field indicate that the device supports a Port Arbitration scheme that uses a programmable arbitration table. Furthermore, it is only valid when one of the above mentioned bits in the Port Arbitration Capability field is selected by the Port Arbitration Select field.

The Port Arbitration Table represents one port arbitration period. Figure 5-48 shows the structure of an example Port Arbitration Table with 128 phases and 2-bit table entries. Each table entry containing a Port Number corresponds to a phase within a port arbitration period. For example, a table with 2-bit entries can be used by a Switch component with up to 4 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.

- When the WRR Port Arbitration is used for a VC of any given port (as an Egress Port for the traffic flow over the VC), a phase containing that port's Port Number is simply skipped by the Port Arbiter.

- When the Time-based WRR Port Arbitration is used for a VC of any given port, a phase containing that port's Port Number indicates an 'idle' time-slot for the Port Arbiter.

The table entry size is determined by the Port Arbitration Table Entry Size field in the VC Resource Capability Register 1. The length of the table is determined by the Port Arbitration Select field as shown in Table 5-44.

When the Port Arbitration Table is used by the default Port Arbitration for the default VC, the default values for the table entries must contain at least one entry for each of other PCI Express ports of the device to ensure forward progress for the default VC for each port. The table may contain RR or RR-like fair Port Arbitration for the default VC.

31 30					5 4	3 2	1 0	Byte Location
Phase[15]	…	…	…	…		Phase[1]	Phase[0]	00h
Phase[31]	…	…	…	…		Phase[17]	Phase[16]	04h
								08h
								0Ch
								10h
								14h
Phase[111]	…	…	…	…		Phase[97]	Phase[96]	18h
Phase[127]	…	…	…	…	…	Phase[113]	Phase[112]	1Ch

Figure 5-48: Example Port Arbitration Table with 128 Phases and 2-bit Table Entries

Table 5-44: Length of Port Arbitration Table

Port Arbitration Select	Port Arbitration Table Length (in # of Entries)
001b	32
010b	64
011b	128
100b	128

5.12. Device Serial Number Capability

The PCI Express Device Serial Number capability is an optional extended capability that may be implemented by any PCI Express device. The Device Serial Number is a read-only 64-bit value that is unique for a given PCI Express device.

All multi-function devices that implement this capability must implement it for function 0; other functions that implement this capability must return the same Device Serial Number value as that reported by function 0.

A PCI Express multi-device component such as a PCI Express Switch that implements this capability must return the same Device Serial Number for each device.

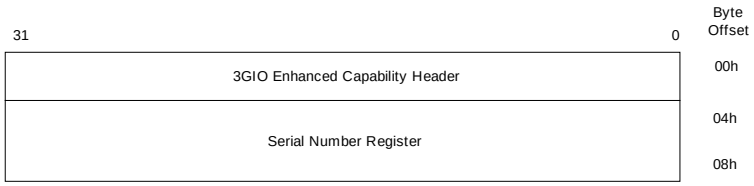

Figure 5-49: PCI Express Device Serial Number Capability Structure

5.12.1. Device Serial Number Enhanced Capability Header (Offset 00h)

See Section 5.9.3 for a description of the PCI Express Enhanced Capability Header. The Extended Capability ID for the Device Serial Number Capability is 0003h.

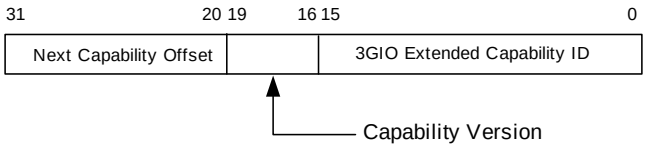

Figure 5-50: Device Serial Number Enhanced Capability Header

Table 5-45: Device Serial Number Enhanced Capability Header

Bit Location	Description	Register Attribute
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	RO

Bit Location	Description	Register Attribute
	the capability structure present. Must be 1h for this version of the specification.	
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express capability structure or 000h if no other items exist in the linked list of capabilities. For Extended Capabilities implemented in device configuration space, this offset is relative to the beginning of PCI compatible configuration space and thus must always be either 000h (for terminating list of capabilities) or greater than 0FFh.	RO

5.12.2. Serial Number Register (Offset 04h)

The Serial Number register is a 64-bit field that contains the IEEE defined 64-bit extended unique identifier (EUI-64™).

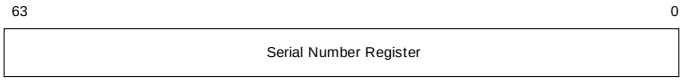

Figure 5-51: Serial Number Register

Table 5-46: Serial Number Register

Bit Location	Description	Register Attribute
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

5.13. Power Budgeting Capability

The PCI Express Power Budgeting Capability allows the system to properly allocate power to devices that are added to the system at runtime. Through this capability, a device can report the power it consumes on a variety of power rails, in a variety of device power management states, in a variety of operating conditions. The system uses this information to ensure that the system is capable of providing the proper power and cooling levels to the device. Failure to properly indicate device power consumption may risk device or system failure.

This capability is required for all devices that are implemented as PCI Express Modules. Implementation of this capability is optional for PCI Express devices that are implemented either as a PCI Express Card or are integrated on the motherboard. Devices that may be implemented either as a PCI Express Module or a PCI Express Card are required to implement this capability.

31 … 0		Byte Offset
3GIO Enhanced Capability Header		00h
RsvdP	Data Select Register	04h
Data Register		08h
RsvdP	Power Budget Capability Register	0Ch

Figure 5-52: PCI Express Power Budgeting Capability Structure

5.13.1. Power Budgeting Enhanced Capability Header (Offset 00h)

See Section 5.9.3 for a description of the PCI Express Enhanced Capability Header. The Extended Capability ID for the Power Budgeting Capability is 0004h.

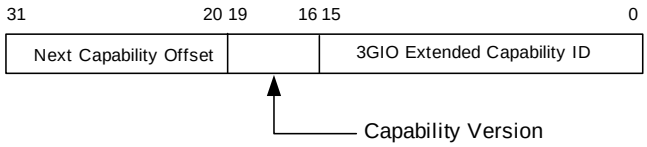

Figure 5-53: Power Budgeting Enhanced Capability Header

Table 5-47: Power Budgeting Enhanced Capability Header

Bit Location	Description	Register Attribute
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the extended capability. Extended Capability ID for the Power Budgeting Capability is 0004h.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the capability structure present. Must be 1h for this version of the specification.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express capability structure or 000h if no other items exist in the linked list of capabilities. For Extended Capabilities implemented in device configuration space, this offset is relative to the beginning of PCI compatible configuration space and thus must always be either 000h (for terminating list of capabilities) or greater than 0FFh.	RO

5.13.2. Data Select Register (Offset 04h)

This read-write register indexes the Power Budgeting Data reported through the Data register and selects the DWORD of Power Budgeting Data that should appear in the Data Register. Index values for this register start at 0 to select the first DWORD of Power Budgeting Data; subsequent DWORDs of Power Budgeting Data are selected by increasing index values.

5.13.3. Data Register (Offset 08h)

This read-only register returns the DWORD of Power Budgeting Data selected by the Data Select Register. Each DWORD of the Power Budgeting Data describes the power usage of the device in a particular 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 should return all zeros.

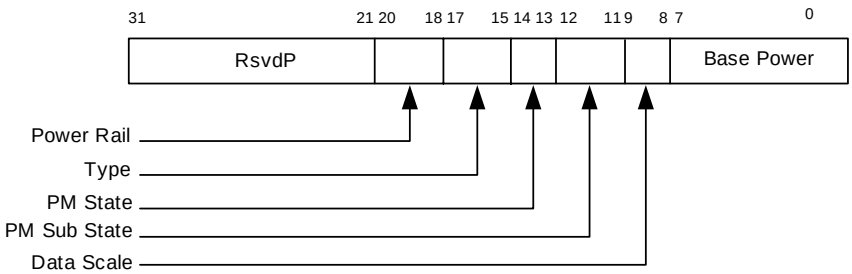

Figure 5-54: Power Budgeting Data Register

The Base Power and Data Scale registers describe the power usage of the device; the Power Rail, Type, PM State, and PM Sub State registers describe the conditions under which the device has this power usage.

Table 5-48: Power Budgeting Data Register

Bit Location	Register Description	Attributes
7:0	**Base Power** – Specifies in Watts the base power value in the given operating condition. This value must be multiplied by the data scale to produce the actual power consumption value.	RO
9:8	**Data Scale** – Specifies the scale to apply to the Base Power value. The power consumption of the device is determined by multiplying the contents of the Base Power register field with the value corresponding to the encoding returned by this field. Defined encodings are: 00b 1.0x 01b 0.1x 10b 0.01x 11b 0.001x	RO

<table>
<tr><th>Bit Location</th><th>Register Description</th><th>Attributes</th></tr>
<tr><td>12:10</td><td>PM Sub State – Specifies the power management sub state of the operating condition being described. Defined encodings are:
<table>
<tr><td>000b</td><td>Default Sub State</td></tr>
<tr><td>001b – 111b</td><td>Device Specific Sub State</td></tr>
</table></td><td>RO</td></tr>
<tr><td>14:13</td><td>PM State – Specifies the power management state of the operating condition being described. Defined encodings are:
<table>
<tr><td>00b</td><td>D0</td></tr>
<tr><td>01b</td><td>D1</td></tr>
<tr><td>10b</td><td>D2</td></tr>
<tr><td>11b</td><td>D3</td></tr>
</table>
A device returns 11b in this field and Aux or PME Aux in the Type register to specify the D3-Cold PM State. An encoding of 11b along with any other Type register value specifies the D3-Hot state.</td><td>RO</td></tr>
<tr><td>17:15</td><td>Type – Specifies the type of the operating condition being described. Defined encodings are:
<table>
<tr><td>000b</td><td>PME Aux</td></tr>
<tr><td>001b</td><td>Auxiliary</td></tr>
<tr><td>010b</td><td>Idle</td></tr>
<tr><td>011b</td><td>Sustained</td></tr>
<tr><td>111b</td><td>Maximum</td></tr>
</table>
All other encodings are reserved.</td><td>RO</td></tr>
<tr><td>19:18</td><td>Power Rail – Specifies the power rail of the operating condition being described. Defined encodings are:
<table>
<tr><td>000b</td><td>Power (12V)</td></tr>
<tr><td>001b</td><td>Power (3.3V)</td></tr>
<tr><td>010b</td><td>Power (1.8V)</td></tr>
<tr><td>111b</td><td>Thermal</td></tr>
</table>
All other encodings are reserved.</td><td>RO</td></tr>
</table>

A device that implements the Power Budgeting Capability is required to provide data values for the D0 Max and D0 Sustained PM State/Type combinations for every power rail from which it consumes power; data for the D0 Max Thermal and D0 Sustained Thermal combinations must also be provided if these values are different from the values reported for D0 Max and D0 Sustained on the power rails.

Devices that support auxiliary power or PME from auxiliary power must provide data for the appropriate power type (Aux or PME Aux).

5.13.4. Power Budget Capability Register (Offset 0Ch)

This register indicates the power budgeting capabilities of a device.

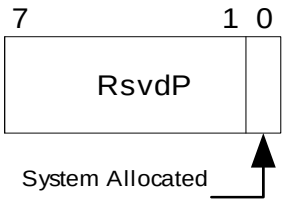

Figure 5-55: Power Budget Capability Register

Table 5-49: Power Budget Capability Register

Bit Location	Register Description	Attributes
0	**System Allocated** – This bit when set indicates that the power budget for the device is included within the system power budget. Reported Power Budgeting Data for this device should be ignored by software for power budgeting decisions if this bit is set.	HwInit

292

6

6. Power Management

This chapter describes PCI Express power management (PCI Express-PM) capabilities and protocols.

6.1. Overview

PCI Express-PM provides the following services:

- A mechanism to identify power management capabilities of a given function
- The ability to transition a function into a certain power management state
- Notification of the current power management state of a function
- The option to wake the system on a specific event

PCI Express-PM is compatible with the *PCI Bus Power Management Interface Specification, Revision 1.1* (PCI-PM), and the *Advanced Configuration and Power Interface Specification, Revision 2.0* (ACPI). This chapter also defines PCI Express native power management extensions. These provide additional power management capabilities beyond the scope of the *PCI Power Management Interface Specification*.

PCI Express-PM defines Link power management states, states that a PCI Express physical Link is permitted to enter in response to either software driven D-state transitions or Active State Link PM activities (Active State Link PM is described later). PCI Express Links 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.

PCI Express components are permitted to wake the system from any supported power management state through the request of a power management event (PME). PCI Express systems may provide the optional auxiliary power supply (Vaux) needed for PME operation from the "off" system states. PCI Express-PM extends beyond its PCI-PM predecessor in this regard as PCI Express PME "messages" include the geographical location (Requestor ID) within the Hierarchy of the requesting agent. These PME messages are in-band TLPs routed from the requesting device to a Root Complex.

Another distinction of the PCI Express-PM PME mechanism is in its separation of the following two tasks that are associated with PME:

- Reactivation (wake) of the I/O Hierarchy (i.e., re-establishing reference clocks and main power rails to the PCI Express components)
- Sending the actual PME Message (vector) to the Root Complex

An autonomous, hardware based active-state mechanism (Active State Link PM) enables power savings even when the connected components are in the D0 state. After a period of idle Link time the Active State Link PM mechanism engages in a physical layer protocol that places the idle Link into a lower power state. Once in the lower power state transitions to the fully operative L0 state are triggered by traffic appearing on either side of the Link. Endpoints initiate entry into a low power Link state. This feature may be disabled by software.

Throughout this document the term Upstream component, or Upstream device, is used to refer to the PCI Express component that is on the end of the PCI Express Link that is hierarchically closer to the root of the PCI Express tree hierarchy. The term Downstream component, or Downstream device, is used to refer to the PCI Express component that is on the end of the Link that is hierarchically further from the root of the PCI Express tree hierarchy.

6.1.1. Statement of Requirements

All PCI Express components, with exception of the Root Complex, are required to meet or exceed the minimum requirements defined by the PCI-PM Software compatible PCI Express-PM features. Root Complexes are required to participate in Link power management DLLP protocols initiated by a downstream device, when all functions of a downstream component enter a PCI-PM Software compatible low power state. For further detail, refer to Section 6.3.2.

The Active State Link PM feature is a required feature (L0s entry at minimum) for all components including Root Complexes, and is configured separately via the native PCI Express configuration mechanisms.

6.2. Link State Power Management

PCI Express defines Link power management states, replacing the bus power management states that were defined by the PCI-PM specification. Link states are not visible to PCI-PM legacy compatible software, and are either derived from the power management D-states of the corresponding components connected to that Link or by Active State power management protocols (Refer to Section 6.4.1).

Note that the PCI Express Physical Layer may define additional intermediate states. See Chapter 4 for more detail on each state and how the Physical Layer handles transitions between states.

PCI Express-PM defines the following Link power management states:

- L0 – Active state.

  All PCI Express transactions and other operations are enabled.

  L0 support is required for both Active State Link power management and PCI-PM compatible power management

- L0s – A low resume latency, energy saving “standby” state.

L0s support is required for Active State Link power management. It is not applicable to PCI-PM compatible power management.

All main power supplies, component reference clocks, and components' internal PLLs must be active at all times during L0s. TLP and DLLP communication over a Link that is in L0s is prohibited. The L0s state is used exclusively for active-state power management.

The PCI Express physical layer provides mechanisms for quick transitions from this state to the L0 state. When common (distributed) reference clocks are used on both sides of a given Link, the transition time from L0s to L0 is typically less than 100 symbol times.

- L1 – Higher latency, lower power "standby" state.

  L1 support is required for PCI-PM compatible power management. L1 is optional for Active State Link power management.

  All platform provided main power supplies and component reference clocks must remain active at all times during L1. The downstream component's internal PLLs may be shut off during L1, enabling greater energy savings at a cost of increased exit latency[27].

  The L1 state is entered whenever all functions of a downstream component on a given PCI Express Link are either programmed to a D-state other than D0, or if the downstream component requests L1 entry (Active State Link PM) and receives positive acknowledgement for the request.

  Exit from L1 is initiated by an upstream initiated transaction targeting the downstream component, or by the downstream component's desire to initiate a transaction heading upstream. Transition from L1 to L0 is typically a few microseconds.

  TLP and DLLP communication over a Link that is in L1 is prohibited.

- L2/L3 Ready – Staging point for removal of main power

  L2/L3 Ready transition protocol support is required

  The L2/L3 Ready state is not directly related to either PCI-PM D-state transitions or to Active State Link power management. L2/L3 Ready is the state that a given Link enters into when the platform is preparing to enter its system sleep state. Following the completion of the L2/L3 Ready state transition protocol for that Link, the Link is then ready for either L2 or L3, but not actually in either of those states until main power has been removed. Depending upon the platform's implementation choices with respect to providing a Vaux supply, after main power has been removed the Link will either settle into L2 (i.e., Vaux is provided), or it will settle into a zero power "off"state (see L3).

---

[27] For example, disabling the internal PLL may be something that is desirable when in $D3_{hot}$, but not so when in D1 or D2.

The L2/L3 Ready state entry transition process must begin as soon as possible following the acknowledgment of a PM_TURN_OFF message, (i.e., the injection of a PM_TO_Ack TLP). The downstream component initiates L2/L3 Ready entry by injecting a PM_Enter_L23 DLLP onto its transmit Port. Refer to Section 6.6 for further detail on power management system messages.

TLP and DLLP communication over a Link that is in L2/L3 Ready is prohibited.

Exit from L2/L3 Ready back to L0 may only be initiated by an upstream initiated transaction targeting the downstream component in the same manner that an upstream initiated transaction would trigger the transition from L1 back to L0. The case where an upstream initiated exit from L2/L3 Ready would occur corresponds to the scenario where, sometime following the transition of the Link to L2/L3 Ready but before main power is removed, the platform power manager decides not to enter the system sleep state.

A Link's transition into the L2/L3 Ready state is one of the final stages involving PCI Express protocol leading up to the platform entering into in a system sleep state wherein main power has been shut off (e.g., ACPI S3 or S4 sleep state).

- L2 – Auxiliary powered Link deep energy saving state.

  L2 support is optional, and dependent upon platform support of Vaux.

  L2 – The downstream component's main power supply inputs and reference clock inputs are shut off.

  - When in L2, all PME detection logic, Link reactivation "Beacon" logic, PME context, and any other "keep alive" logic is powered by Vaux.

  TLP and DLLP communication over a Link that is in L2 is prohibited.

  Exiting the L2 state is accomplished by reestablishing main power and reference clocks to all components within the domain of the power manager, followed by full Link training and initialization. Once a given Link has completed Link training and initialization it is then in the L0 state and may begin sending and receiving TLPs and DLLPs.

- L3 – Link Off state.

  Zero power state.

Refer to Section 4.2 for further detail relating to entering and exiting each of the PCI Express L-states.

Figure 6-1 highlights the legitimate L-state transitions that may occur during the course of Link operation.

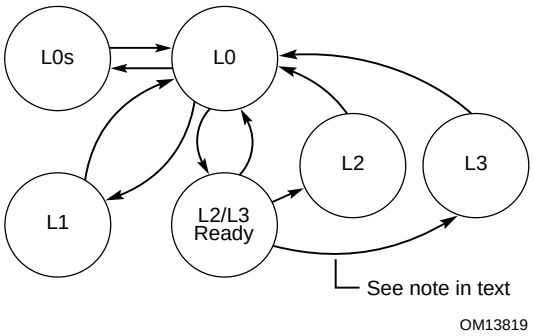

Figure 6-1: Link Power Management State Transitions

The arc noted in Figure 6-1 indicates the case where the platform does not provide Vaux. In this case, the L2/L3 Ready state transition protocol results in a state of readiness for loss of main power, and once removed the Link settles into the L3 state.

Link PM Transitions from any L-state to any other L-state must pass through the L0 state during the transition process with the exception of the L2/L3 Ready to L2 or L3 transitions. In this case, the Link transitions from L2/L3 Ready directly to either L2 or L3 when main power to the component is removed. (This follows along with a corresponding component's D-state transition from $D3_{hot}$ to $D3_{cold}$)

The following sequence, leading up to entering a system sleep state, illustrates the multi-step Link state transition process:

1. System Software directs all functions of a downstream component to $D3_{hot}$.
2. The downstream component then initiates the transition of the Link to L1 as required.
3. System Software then causes the Root Complex to broadcast the PM_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 PM_TO_Ack.
5. After the PM_TO_Ack is sent, the downstream component then initiates the L2/L3 Ready transition protocol.

L0 --> L1 --> L0 --> L2/L3 Ready

Table 6-1 summarizes each L-state, describing when they are used, and the PCI Express platform, and PCI Express component behaviors that correspond to each of them.

A "Yes" entry indicates that support is required (unless otherwise noted). "On" and "Off" entries indicate the required clocking and power delivery. "On/Off" indicates an optional design choice.

Table 6-1: Summary of PCI Express Link Power Management States

	L-State Description	Used by SW Directed PM	Used by Active State Link PM	Platform Reference Clocks	Platform Main Power	Component Internal PLL	Platform Vaux
L0	Fully active Link	Yes (D0)	Yes (D0)	On	On	On	On/Off
L0s	Standby State	No	Yes [1] (D0)	On	On	On	On/Off
L1	Lower Power Standby	Yes ($D1-D3_{hot}$)	Yes[2] (opt., D0)	On	On	On/Off[3]	On/Off
L2/L3 Ready	Staging point for power removal	Yes [4]	No	On	On	On/Off	On/Off
L2	Low Power Sleep State (all clks, main power off)	Yes [5]	No	Off	Off	Off	On [6]
L3	Off (zero power)	n/a	n/a	Off	Off	Off	Off

Notes:

1. L0s exit latency will be greatest in Link configurations characterized by independent reference clock inputs for components connected to opposite ends of a given Link. (vs. a common, distributed reference clock)
2. L1 entry may be requested within Active State Link PM protocol, however its support is optional.
3. L1 exit latency will be greatest for components that internally shut off their PLLs during this state
4. L2/L3 Ready entry sequence is initiated at the completion of the PM_Turn_Off/PM_TO_Ack protocol handshake . It is not directly affiliated with a D-State transition, or a transition in accordance with Active State Link PM policies and procedures.
5. Depending upon the platform implementation, the system's sleep state may utilize the L2 state or transition to being fully off (L3). L2/L3 Ready state transition protocol is initiated by the downstream component following reception and TLP acknowledgement of the PM_Turn_Off TLP Message. While platform support for an L2 sleep state configuration is optional (i.e., support for Vaux delivery), PCI Express component protocol support for transitioning the Link to the L2/L3 Ready state is required.
6. L2 is distinguished from the L3 state only by the presence of Vaux. After the completion of the L2/L3 Ready state transition protocol and before main power has been removed the Link has indicated its readiness main power removal.

6.3. PCI-PM Software Compatible Mechanisms

6.3.1. Device Power Management States (D-States) of a Function

PCI Express supports all PCI-PM device power management states. All functions must support the D0 and D3 states (both $D3_{hot}$ and $D3_{cold}$). The D1 and D2 states are optional. Refer to the *PCI Bus Power Management Interface Specification* for further detail relating to the PCI-PM compatible features described in this specification. Note that where this specification defines detail that departs from the *PCI Bus Power Management Interface Specification*, this specification takes precedence for PCI Express components and Link hierarchies.

6.3.1.1. D0 State

All PCI Express functions must support the D0 state. D0 is divided into two distinct sub-states, the "un-initialized" sub-state and the "active" sub-state. When a PCI Express component initially has its power applied, it defaults to the $D0_{uninitialized}$ state. Components that are in this state will be enumerated and configured by the PCI Express Hierarchy enumeration process. Following the completion of the enumeration and configuration process the function enters the $D0_{active}$ state, the fully operational state for a PCI Express function. A function enters the $D0_{active}$ state whenever any single or combination of the function's Memory Space Enable, I/O Space Enable, or Bus Master Enable bits have been enabled by system software

6.3.1.2. D1 State

D1 support is optional. While in the D1 state, a function must not initiate any TLPs on the Link with the exception of a PME Message as defined in Section 6.3.3. Configuration requests are the only TLP accepted (as target) by a function that is currently in the D1 state. All other received Requests must be handled as Unsupported Requests.

Note that a function's software driver participates in the process of transitioning the function from D0 to D1. It contributes to the process by saving any functional state (if necessary), and otherwise preparing the function for the transition to D1. As part of this quiescence process the function's software driver must ensure that any mid-transaction TLPs (i.e., Requests with outstanding Completions), are terminated prior to handing control to the system configuration software that would then complete the transition to D1.

6.3.1.3. D2 State

D2 support is optional. While in the D2 state, a function must not initiate any TLPs on the Link with the exception of a PME Message as defined in Section 6.3.3. Configuration requests are the only TLP accepted (as target) by a function that is currently in the D2 state. All other received TLPs must be handled as unsupported packets.

Note that a function's software driver participates in the process of transitioning the function from D0 to D2. It contributes to the process by saving any functional state (if necessary), and otherwise preparing the function for the transition to D2. As part of this quiescence process the function's software driver must ensure that any mid-transaction TLPs (i.e., Requests with outstanding Completions), are terminated prior to handing control to the system configuration software that would then complete the transition to D2.

6.3.1.4. D3 State

D3 support is required, (both the $D3_{cold}$ and the $D3_{hot}$ states). Functions supporting PME generation from D3 must support it for both $D3_{cold}$ and the $D3_{hot}$ states.

Functional context does not need to be maintained by functions in the D3 state. Software is required to re- initialize the function following a D3 → D0 transition.

The minimum recovery time following a $D3_{hot}$ → D0 transition is 10 ms. This recovery time may be used by the $D3_{hot}$ → 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.

6.3.1.4.1. $D3_{hot}$ State

When a function is in $D3_{hot}$, it must respond to configuration accesses targeting it. They must also participate in the PM_Turn_Off/PM_TO_Ack protocol. Refer to Section 6.3.3 details. Once in $D3_{hot}$ the function can later be transitioned into $D3_{cold}$ (by removing power from its host component).

Transitions into the $D3_{hot}$ state are used to establish a standard process for graceful saving of functional state immediately prior to entering a deeper power savings state where power is removed.

Note that a function's software driver participates in the process of transitioning the function from D0 to $D3_{hot}$. It contributes to the process by saving any functional state that would otherwise be lost with removal of main power, and otherwise preparing the function for the transition to $D3_{hot}$. As part of this quiescence process the function's software driver must ensure that any outstanding transactions (i.e., Requests with outstanding Completions), are terminated prior to handing control to the system configuration software that would then complete the transition to $D3_{hot}$.

Note that $D3_{hot}$ is also a useful state for reducing power consumption by idle components in an otherwise running system.

300

6.3.1.4.2. $D3_{cold}$ State

A function transitions to the $D3_{cold}$ state when its power is removed. A power-on sequence transitions a function from the $D3_{cold}$ state to the $D0_{Uninititialized}$ state. At this point software must perform a full initialization of the function in order to re-establish all functional context, completing the restoration of the function to its $D0_{active}$ state.

Functions that support PME assertion from $D3_{cold}$ must maintain their PME context for inspection by PME service routine software during the course of the resume process. Functions may only generate PME messages from $D3_{cold}$ if the platform supplies them with a Vaux supply or if they have an independent source of power.[28] PME context consists of all information relating to the function's assertion of PME.

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, SHPC extended context, etc.

A function's PME assertion is acknowledged when system software performs a "write 1 to clear" configuration write to the asserting function's PME_Status bit of its PCI-PM compatible PMCSR register.

An auxiliary power source must be used to support PME event detection, 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 6.3.3 for further PME specific detail. Exit from $D3_{cold}$ is accomplished with assertion of PWRGOOD, (either provided as an auxiliary signal or internally generated by the component), followed by the Link training sequence.

[28] Note that when a component reports support for PME generation from $D3_{hot}$ and $D3_{cold}$ (PMC register) this does not constitute a guarantee that the platform will support the generation of PMEs from $D3_{cold}$. To be certain of this, software must poll the components' PCI Express capability structures to ensure that the components report that Vaux is being provided to them by the platform (refer to Chapter 5 for details).

6.3.2. 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 6-2 depicts the relationships between the power state of a component (Endpoint, Switch) and its Upstream Link.

Table 6-2: Relation Between Power Management States of Link and Components

Downstream Component D-State	Permissible Upstream Component D-State	Permissible Interconnect State
D0	D0	L0, L0s, L1 (1)
D1	D0-D1	L1
D2	D0-D2	L1
$D3_{hot}$	D0-$D3_{hot}$	L1, L2/L3 Ready(2)
$D3_{cold}$	D0-$D3_{cold}$	L2 (3), L3

Notes:

1. All PCI Express components are required to support Active-State Link Power Management with L0s entry during idle at a minimum. The use of L1 within D0 is optional.
2. When all functions within a downstream component are programmed to $D3_{hot}$ the downstream component must request the transition of its Link to the L1 state using the PM_ENTER_L1 DLLP. Once in $D3_{hot}$, following the execution of a PM_TURN_OFF / PM_TO_Ack handshake sequence, the downstream component must then request a Link transition to L2/3 Ready using the PM_ENTER_L23 DLLP. Following the L2/L3 Ready entry transition protocol the downstream component must be ready for loss of main power and reference clock.
3. If Vaux is provided by the platform, the Link sleeps in L2. In the absence of Vaux, the L-state is L3

The conditions governing Link state transition in the software directed PCI-PM compatible power management scheme are defined as:

- A Switch or single function Endpoint device must initiate a Link state transition of its Upstream Port (Switch), or Port (endpoint), to L1 based solely upon that Port being programmed to D1, D2, or $D3_{hot}$. In the case of the Switch, system software bears the responsibility of ensuring that any D-state programming of a Switch's Upstream Port is done in a compliant manner with respect to PCI Express hierarchy-wide PM policies (i.e., the Upstream Port cannot be programmed to a D-state that is any less active that the most active downstream Port and downstream connected component/function(s)).

- Multi-function Endpoints must not initiate a Link state transition to L1 until all of their functions have been programmed to a non-D0 D-state.

6.3.2.1. Entry into the L1 State

Figure 6-2 depicts the process by which a Link is transitioned into the L1 state as a direct result of power management software programming the downstream connected component into a lower power state, (either D1, D2, or $D3_{hot}$ state). This figure and the subsequent description outline the transition process for a single function downstream component that is being programmed to a non-D0 state.

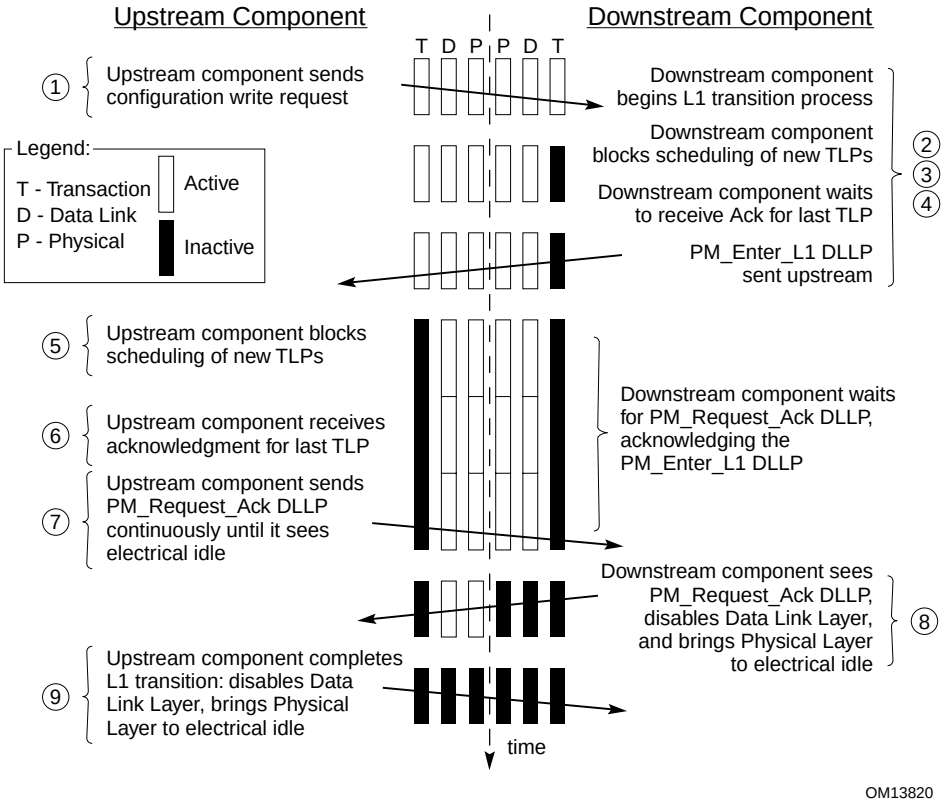

Figure 6-2: Entry into L1 Link State

The following text provides additional detail for the Link state transition process pictured above.

PM Software Request:

1. PM Software (upstream component) sends a TLP configuration request packet to change the downstream function's D-state (D1 for example).

Downstream Component Link State Transition Initiation Process:

2. The downstream component schedules the completion response corresponding to the configuration write to its PMCSR PowerState field. All new TLP scheduling is suspended.
3. The downstream component then waits until it receives a Link layer acknowledgement for the PMCSR write completion, and any other TLPs it had previously sent. The component may retransmit a TLP out of its Link Layer Retry buffer if required to do so by Link layer rules.
4. Once all of the downstream component's TLPs have been acknowledged the downstream component transmits a PM_Enter_L1 DLLP onto its upstream-directed (transmit) Port. The downstream component sends this DLLP continuously until it receives a response from the upstream component[29] (PM_Request_Ack). While waiting for all of its TLPs to be acknowledged the downstream component must not initiate any new TLPs. The downstream component must still however continue to accept TLPs and DLLPs from the upstream component, and it must also continue to respond with DLLPs as needed per Link Layer protocol. Refer to the Electrical chapter for more details on the physical layer behavior.

Upstream Component Link State Transition Process:

5. Upon receiving the PM_Enter_L1 DLLP the upstream component blocks the scheduling of any future TLPs.
6. The upstream component then must wait until it receives a Link layer acknowledgement for the last TLP it had previously sent. The upstream component may retransmit a TLP from its Link layer retry buffer if required to do so by the Link layer rules.
7. Once all of the upstream component's TLPs have been acknowledged the upstream component sends a PM_Request_Ack DLLP downstream. The upstream component sends this DLLP continuously until it observes its receive Lanes enter

[29] If at this point the Downstream component needs to initiate a transfer on the Link, it must first complete the transition to L1 regardless. Once in L1 it is then permitted to initiate an exit L1 to handle the transfer. This corner case represents an event requiring a PME message occurring during the component's transition to L1.

into the electrical idle state. See Chapter 4 for more details on the Physical Layer behavior.[30]

Completing the L1 Link State Transition:

8. Once the downstream component has captured the PM_Request_Ack DLLP on its receive Lanes (signaling that the upstream component acknowledged the transition to L1 request), it then disables its Link layer and brings the upstream directed physical Link into the electrical idle state.
9. When the upstream component observes its receive Lanes enter the electrical idle state, it then stops sending PM_Request_Ack DLLPs, disables its Link layer and brings its transmit Lanes to electrical idle completing the transition of the Link to L1.

   When two components' interconnecting Link is in L1 as a result of the downstream component being programmed to a non-D0 state, both components suspend the operation of their Flow Control Update, DLLP ACK/NAK Latency, and TLP Completion Timeout counter mechanisms[31]. Refer to the Electrical chapter for more detail on the physical layer behavior.

Components on either end of a Link in L1 may optionally disable their internal PLLs in order to conserve more energy. Note however that platform supplied main power, and reference clocks must always be supplied to components on both ends of an L1 Link.

6.3.2.2. Exit from L1 State

L1 exit can be initiated by the component on either end of a PCI Express Link. A downstream component would initiate an L1 exit transition in order to bring the Link to L0 such that it may then inject a PME message.

The upstream component initiates L1 exit to re-establish normal TLP and DLLP communications on the Link.

In either case the physical mechanism for transitioning a Link from L1 to L0 is the same and are described in detail within the Electrical Chapter.

Figure 6-3 outlines a sequence that would trigger an Upstream component to initiate transition of the Link to the L0 state.

---

[30] If, at this point, the Upstream component for any reason needs to initiate a transfer on the Link, it must first complete the transition to L1 regardless. Once in L1 it is then permitted to initiate an exit from L1 to handle the transfer.

[31] This is the required behavior regardless of whether the L1 state was the result of software driven PM protocol, or the result of Active State Link PM protocol.

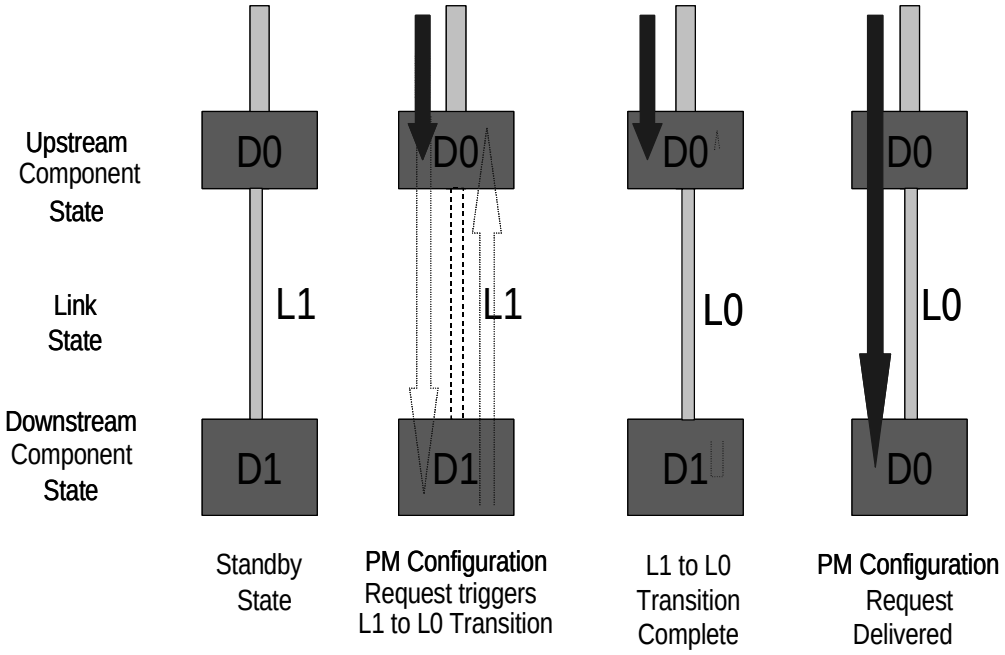

Figure 6-3: Exit from L1 Link State Initiated by Upstream Component

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. 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.
4. Once both directions of the Link are back to the active L0 state, the Upstream Port sends the configuration Packet Downstream.

6.3.2.3. 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 PM_Turn_Off / PM_TO_Ack handshake sequence. Any actions on the part of the downstream component necessary to ready itself for loss of power must be completed prior to initiating the transition to L2/L3 Ready. Once all preparations for loss of power and clock are completed L2/L3 Ready entry is initiated by the downstream component by sending the PM_Enter_L23 DLLP upstream.

  In contrast, the time for L1 entry transition is indicated by programming all of the downstream component's function(s) to non-D0 states, or by Active State Link PM policies. There are no preparations necessary before initiating a transition to L1.

- The downstream component must be in $D3_{hot}$ prior to being transitioned into the L2/L3 Ready state, i.e., a PM_Turn_Off message must never be sent unless all functions downstream of its point of origin are currently in $D3_{hot}$.

  In contrast, a downstream component initiating a transition to L1 would have always initially been in D0, and had just be reprogrammed to D1, D2, or $D3_{hot}$.

- L2/L3 Ready entry transition protocol uses the PM_Enter_L23 DLLP.

  The L1 entry protocol uses the PM_Enter_L1.DLLP.

  In either case, the PM_Enter_Lx DLLP is sent repeatedly until the downstream component observes electrical idle on its receive Port.

6.3.3. Power Management Event Mechanisms

6.3.3.1. Motivation

The PCI Express PME mechanism is software compatible with the PME mechanism defined by the PCI-PM specification. Power Management Events are generated by PCI Express functions as a means of requesting a PM state change. Power Management Events are typically utilized to revive the system or an individual function from a low power state.

Power management software may transition a PCI Express Hierarchy into a low power state, and transition the upstream links of these devices into the non-communicating L2 state[32]. The PCI Express PME generation mechanism is therefore broken into two components:

- Waking a non-communicating Hierarchy. This step is required only if the upstream Link of the device originating the PME is in the non-communicating L2 state, since in that state the device cannot send a PM_PME message upstream.

- Sending a PM_PME message to the root of the PCI Express Hierarchy

PME indications are propagated to the Root Complex in the form of TLP messages. PM_PME messages include the logical location of the requesting agent within the Hierarchy (in the form of the Requester ID of the PME message header). Explicit identification within

[32] The L2 state is defined as "non-communicating" since component reference clock and main power supply are removed in that state.

the PM_PME message is intended to facilitate quicker PME service routine response, and hence shorter resume time.

6.3.3.2. Link Reactivation

The Link reactivation mechanism provides a means of signaling the platform to re-establish power and reference clocks to the components within its domain. Refer to Section 4.2 for details on the in-band mechanism for Link reactivation. Refer to the *PCI Express Card Electromechanical Specification* for details on the out-of-band mechanism for Link reactivation.

Systems that allow PME generation from $D3_{cold}$ state must provide auxiliary power to support Link reactivation when the main system power rails are off.

The reactivation period ends when the upstream-directed Link of a device enters the initialization phase as a result of Link transition from the L2 state to the L0 state–a power-on sequence transitions the device from the $D3_{cold}$ state to the $D0_{uninitialized}$ state.

The downstream device shall cease requesting Link reactivation (either in-band or auxiliary out-of-band) once it has entered the $D0_{uninitialized}$ state.

Once the Link has been re-activated and trained, the requesting agent then propagates a PM_PME message upstream to the Root Complex.

6.3.3.2.1. PME Fence

PCI Express devices need to be notified before their reference clock and main power may be removed so that they can prepare for that eventuality. PCI Express-PM introduces a fence mechanism that serves to initiate the power removal sequence while also coordinating the behavior of the platform's power management controller and PME handling by PCI Express agents.

There exist race conditions where a downstream agent, if not somehow coordinated with the platform's power manager, could potentially initiate a PM_PME message while the power manager was in the process of turning off the main power source to the Link Hierarchy. The net result of hitting this corner condition would be loss of the PME indication. The fence mechanism ensures this does not happen.

PME_Turn_Off Broadcast Message

Before main component power and reference clocks are turned off the Root Complex or Hot Plug controller within a Switch Downstream Port, must issue a broadcast message that instructs all agents downstream of that point within the hierarchy to cease initiation of any subsequent PM_PME messages, effective immediately upon receipt of the PME_Turn_Off message.

Each PCI Express agent is required to respond with a TLP "acknowledgement" Packet, PME_TO_ACK that is, as in the case of a PME Message, always routed upstream. In all

cases, the PM_TO_Ack message must terminate at the PM_Turn_Off message's point of origin.[33]

Note that PM_PME and PME_TO_Ack, like all other PCI Express message packets, are handled as posted transactions. It is their posted transaction nature that ensures that any previously injected PM_PME messages will be pushed ahead of the fence acknowledgement assuring full in-order delivery of any previously initiated PM_PME messages before the "Turn off" acknowledgement ever reaches the initiator of the PM_Turn_Off message.

For the case where a PM_Turn_Off message is initiated upstream of a PCI Express Switch, the PCI Express Switch's Upstream Port must report an "aggregate" acknowledgement only after having received PME_TO_ACK packets from each of their downstream ports individually. Once a PM_TO_Ack Packet has arrived on all downstream ports, the Switch then sends a PM_TO_Ack packet on its upstream Port.

All PCI Express components must accept[34] and acknowledge the PME_Turn_Off Packet from within the $D3_{hot}$ State. Once an Endpoint has sent a PME_TO_Ack Packet on its transmit Link, it must then prepare for removal of its power and reference clocks by initiating a transition to the L2/L3 Ready state.

A Switch must also transition its upstream Link to the L2/L3 Ready state in the same manner as described in the previous paragraph for Endpoints. However, the Switch initiates this transition only after all of its downstream ports have entered L2/L3 Ready state.

The Links attached to the originator of the PME_Turn_Off message are the last to assume the L2/L3 Ready state. This serves as an indication to the power delivery manager[35] that all Links within that portion of the PCI Express hierarchy have:

- Successfully retired all in flight PME messages to the point of PME_Turn_Off message origin
- Performed any necessary local conditioning in preparation for power removal

The power delivery manager must wait a minimum of 100 ns after observing all links corresponding to the point of origin of the PME_Turn_Off message enter L2/L3 Ready before removing the components' reference clock and main power.

---

33 Point of origin for the PM_Turn_Off message could be all of the Root Ports for a given Root Complex (full platform sleep state transition), an individual hot plug capable Root Port, or a hot plug capable Switch Downstream Port.

34 FC credits permitting.

35 Power delivery control within this context relates to control over the entire PCI Express Link hierarchy, or over a subset of PCI Express links ranging down to a single PCI Express Link for sub hierarchies residing downstream of a Hot Plug controller managed interconnect.

Implementation Note: PM_TO_Ack Message Proxy by Switch Devices

One of the PM_Turn_off / PM_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 PM_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 PM_TO_Ack is received at the Root Complex. Once all downstream ports of the Root Complex receive a PM_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 so therefore must wait for all of their connected downstream ports to receive a PM_TO_Ack message before it can send a PM_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 PM_Turn_Off broadcast message has been broadcast downstream of the switch, the switch simply checks off each downstream port having received a PM_TO_Ack. Once the last of its active downstream ports receives a PM_TO_Ack the switch will then send a single PM_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 PM_TO_Ack message and the switch has scored its arrival, the port is then free to drop the packet from its internal queues and free up the corresponding posted request queue FC credits.

Implementation Note: PME_TO_Ack Deadlock Avoidance

As specified earlier, any device that detects a PME_Turn_Off message must reply with a PME_TO_Ack message. However, system behavior must not depend on the correct behavior of any single device. In order to avoid deadlock in the case that one or more devices do not respond with a PME_TO_Ack message, the power manager must not depend on the acceptance of a PME_TO_Ack message. For example, the power manager may timeout after waiting for the PME_TO_Ack message for a given time, after which it proceeds as if the message was accepted.

6.3.3.3. PM_PME Messages

PM_PME messages are posted Transaction Layer Packets (TLPs) that inform the power management software which agent within the PCI Express Hierarchy requests a PM state change. PM_PME messages, like all other Power Management system messages must use the general purpose Transfer Class, TC #0.

PM_PME messages are always routed in the direction of the Root Complex. To send a PM_PME message on its upstream Link, a device must transition the Link to the L0 state (if the Link was not is that state already). Unless otherwise noted, the device will keep the Link in the L0 state following the transmission of a PM_PME message.

6.3.3.3.1. PM_PME 'Backpressure"Deadlock Avoidance

A PCI Express Root Complex is typically implemented with local buffering to temporarily store a finite number of PM_PME messages that could potentially be simultaneously propagating through the PCI Express Hierarchy at any given time. Given a limited number of PM_PME messages that can be stored within the Root Complex, there can be backpressure applied to the upstream directed posted queue in the event that the capacity of this temporary PM_PME message buffer is exceeded.

Deadlock can occur according to the following example scenario:

- Incoming PM_PME messages fill the Root Complex's temporary storage to its full capacity while there are additional PM_PME messages still in the Hierarchy making their way upstream.
- Root Complex, on behalf of system software, issues split configuration read request targeting one of the PME requester's PMCSR (e.g., reading its PME_Status bit).
- The corresponding split completion Packet is required, as per producer/consumer ordering rules, to push all previously posted PM_PME messages out ahead of it, which in this case are PM_PME messages that have no place to go.
- PME service routine cannot make progress, PM_PME message storage situation does not improve.
- 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 6.3.3.2.

6.3.3.4. PME Rules

- All PCI Express components supporting PCI Express-PM must implement the PCI-PM PMC and PMCSR registers in accordance with the PCI-PM specification. These registers reside in the PCI-PM compliant PCI Capability List format.
  - PME capable functions must implement the PME_Status bit, and underlying functional behavior, in their PMCSR configuration register.
  - When a function initiates Link reactivation, 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
- PME capable agents must comply with PME_Turn_Off and PME_TO_Ack fence protocols
- Before a Link or a portion of Hierarchy is transferred into a non-communicating state (i.e., a state they cannot issue a PM_PME Message from), a PME_Turn_Off Message must be broadcast Downstream.

6.3.3.5. PM_PME Delivery State Machine

The following diagram conceptually outlines the PM_PME delivery control state machine. This state machine determines ability of a Link to service PME events by issuing PM_PME immediately vs. requiring initial Link reactivation.

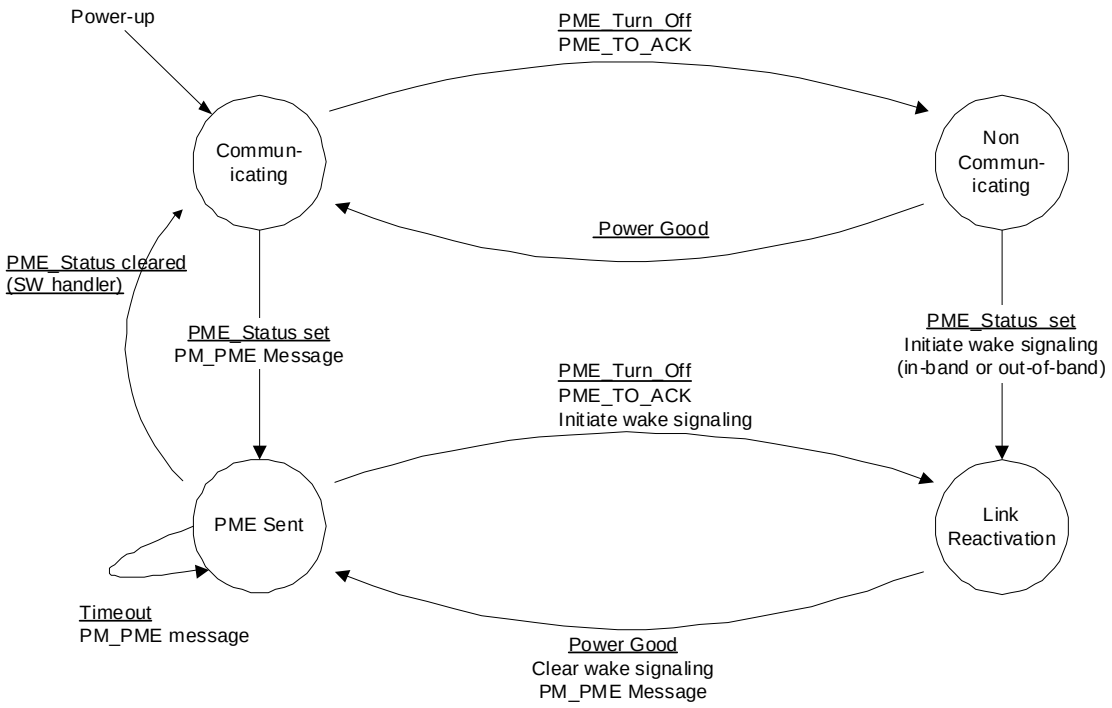

Figure 6-4: A Conceptual PME Control State Machine

Communicating State:

At initial power-up, the Upstream Link the enters "Communicating" state

- If PME_Status is asserted (assuming PME delivery is enabled), a PM_PME Message will be issued Upstream, terminating at the root of the PCI Express Hierarchy. The next state is the "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:

- If a Power Good signal transitions from inactive to active state (an indication that power and clock have been restored), the next state is the "Communicating" state.
- If PME_Status is asserted, the Link will transition to "Link Reactivation" state, and activate the wake 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 cleared by the time the PME service timeout expires, a PM_PME message is re-sent upstream. See Section 6.3.3.3.1 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 PM_TO_ACK message. The device also activates the wake mechanism.

Link Reactivation State

- If a Power Good signal transitions from inactive to active state, the Link resumes a transaction-capable state. The device clears the wake signaling, issues a PM_PME Upstream and transitions into the "PME Sent" state.

Implementation Note: PCI Express-to-PCI Bridge PME Considerations

PCI Express-to-PCI Bridges must "bridge" power management events from the original PME# wire'or signal connections to the PCI Express in-band PME messaging scheme. PCI Express-to-PCI Bridges are required to identify all PME messages that they issue on behalf of downstream legacy PCI functions as coming from the PCI bus segment where the PME# originated.

A design consideration that must be comprehended is the potential for lost PME indications. This particular issue is unique to PCI Express-to-PCI Bridges where the level sensitive PME# signal is transformed into what is effectively an edge triggered PME messaging scheme and manifests itself in a race condition. The corner case corresponds to the situation where one of the legacy PCI components asserts PME# (which now must be input into the bridge, and not routed around it as in PCI-PM PME# routing). Following this the PCI Express-to-PCI bridge injects a PME message on behalf of the legacy agent. If another Legacy PME# assertion occurs (on the same PME# input to the bridge) before the original PME# service routing has cleared the PME_Status bit of the original PME# initiator, then given the wire'or nature of the PCI-PM PME#, PME# input at the bridge will remain asserted following the clearing of the first agent's PME_Status bit.

The net result is that the first PM_PME was serviced successfully, (single PM_PME message propagated upstream facilitated this), however the second PM_PME was lost. In order to avoid loss of PM_PMEs in the conversion of the level-triggered PCI PME to the edge triggered PCI Express PM_PME message, the PCI PME signal must be periodically polled and a PCI Express PM_PME message must be generated if the PCI PME is sensed asserted.

While the above scheme introduces the possibility of spurious PM_PMEs, these are deemed benign and would be ignored by the operating system.

It is the responsibility of the PCI Express-to-PCI Bridge to engage in the PME fence protocol on behalf of its downstream PCI devices. The PME_Turn_Off message will terminate at the PCI Express-to-PCI Bridge, and will not be communicated to the downstream PCI devices. The PCI Express-to-PCI Bridge will not issue a PM_PME message on behalf of a downstream PCI device while its upstream Link is in the L2 non-communicating state.

315

6.4. 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 5 of this specification. Refer to Chapter 5 for specific register locations, bit assignments, and access mechanisms associated with these PCI Express-PM features.

6.4.1. Active-State Power Management

All PCI Express components 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 they must implement Active State Link PM's minimum requirements.

Components in the D0 state (i.e., fully active state) normally keep their Upstream Link in the active L0 state, as defined in Section 6.3.2. Active-state Link power management defines a protocol for components in the D0 state to reduce Link power by placing their Upstream 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 Active State Link Power Management. The L0s low power Link state is optimized for short entry and exit latencies, while providing substantial power savings. If the L0s state is enabled in a device, it is required to bring any transmit Link into L0s state whenever that Link is not in use (refer to Section 6.4.1.1.1 for details relating to the L0s invocation policy). All PCI Express components must support the L0s Link state from within the D0 device state.

The L1 Link state is optimized for maximum power savings at a cost of longer entry and exit latencies. L1 reduces Link power beyond the L0s state for cases where very low power is required and longer transition times are acceptable. Active State Link PM support for the L1 Link state is optional.

Each PCI Express component must report its level of support for Active State Link Power Management in the Active State Link PM Support configuration field.

Each PCI Express component shall also report its L0s and L1 exit latency (the time that they require to transition from the L0s or L1 state to the L0 state). Endpoints must also report the worst-case latency that they can withstand before risking, for example, internal fifo overruns due to the transition latency from L0s or L1 to the L0 state. Power management software can use the provided information to then enable the appropriate level of Active State Link Power Management.

The L0s exit latency may differ significantly if the reference clock for opposing sides of a given Link is provided from the same source, or delivered to each component from a different source. PCI Express-PM software informs each PCI Express device of its clock configuration via the "common clock configuration" bit in their PCI Express Capability Structure's Link Control Register. This bit serves as the determining factor in the L0s exit latency value reported by the device. All PCI Express devices power on with Active State Link Power Management turned off by default. Software can enable active state Link power management using a process described in Section 6.4.1.3.1.

Power management software enables (or disables) Active State Link Power Management in each Port of a component by programming the Active State Link PM Control field. Note that new BIOS code can effectively enable or disable Active State Link PM functionality even when running with a legacy operating system.

Implementation Note: Isochronous Traffic and Active State Link Power Management

Isochronous traffic requires bounded service latency. Active State Link Power Management may add latency to isochronous transactions beyond expected limits. A possible solution would be to disable Active State Link Power Management for devices that are configured with an Isochronous Virtual Channel.

Multi-function endpoints may be programmed with different values in their respective Active_PM_En registers of each function. The policy for such a component will be dictated by the most active common denominator among all D0 functions according to the following rules:

- Functions in non-D0 state (D1 and deeper) are ignored in determining the Active State Link Power Management policy
- If any of the D0 functions has its Active State Power Link Management disabled, (Active State Link PM Control field = 00b), then Active State Link Power Management is disabled for the entire component.
- Else, if at least one of the D0 functions is enabled for L0s only (Active State Link PM Control field = 01b), then Active State Link Power Management is enabled for L0s only
- Else, Active State Link Power Management is enabled for both L0s and L1 states

Note that the components must be capable of changing their behavior during runtime as devices enter and exit low power device states. For example, if one function within a multi-function component is programmed to disable Active State Link Power Management, then Active State Link Power Management will be disabled for that component while that function is in the D0 state. Once the function transitions to a non-D0 state, Active State Power Management will be enabled to at least the L0s state if all other functions are enabled for Active State Link PM.

317

6.4.1.1. L0s Active State Link Power Management State

All PCI Express devices must support the L0s low power Link state. All components power up to a default state where Active State Link PM is disabled.

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 solution which may differ significantly in their **L0s** exit latency, and therefore in the efficiency of Active State Link Power Management. To achieve maximum power savings efficiency with Active State Power Link Management, **L0s** exit latency should be kept low by proper selection of the clocking solution.

6.4.1.1.1. Entry to 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.

A Port that is disabled for the L0s state must not transition its transmitting Lanes to the L0s state. It must still however be able to tolerate having its receiver Port Lanes entering L0s, (as a result of the device at the other end bringing its transmitting Lanes into L0s state), and then later returning to the L0 state.

L0s Invocation Policy

PCI Express ports that are enabled for L0s entry must transition their transmit Lanes to the L0s state if the defined idle conditions are met for a specified period of time. The port may choose this period of time to be anywhere within the range of:

(port's reported L0s exit latency)/4 ≤ t ≤ port's reported L0s exit latency

Defining the invocation time as a range enables the tuning of ASPM behavior, balancing power savings with performance.

Definition of Idle

The definition of "idle" varies with device category

An Endpoint Port or Root Complex Root Port is determined to be idle if the following conditions are met:

- No TLP is pending to transmit over the Link, or no FC credits are available to transmit anything
- No ACK, NAK, or ACK Timeout DLLPs are pending for transmission

A Switch's upstream Port is determined to be idle if the following conditions are met:

- All of the Switch's Downstream Port receive Lanes are in the L0s state
- No pending TLPs to transmit, or no FC credits are available to transmit anything
- No ACK, NAK, or ACK Timeout 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 in the L0s state
- No pending TLPs to transmit on this Link, or no FC credits are available
- No ACK, NAK, or ACK Timeout DLLPs are pending for transmission

See Section 4.2 for details on L0s entry by the Physical Layer.

6.4.1.1.2. Exit from L0s State

Components from either end of a PCI Express Link may initiate an exit from the L0s low power Link state.

Note that a transition from the L0s Link state should never depend on the status (or availability) of FC credits. The Link must be able to reach the Link Active 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 where new credits can be sent across the Link.

Downstream Initiated Exit

An Endpoint or Switch is permitted to initiate an exit from the L0s low power state on its transmit Link, (Upstream Port transmit Lanes in the case of a downstream Switch), if it needs to communicate through the Link. The component initiates a transition to the L0 state on Lanes in the upstream direction as described in Section 4.2.

If the Upstream component is a Switch (i.e., it is not the Root Complex), then it must initiate a transition on its Upstream Port transmit Lanes (if the Upstream Port's transmit Lanes are in a low power state) as soon as it detects an exit from L0s on any of its downstream ports.

Upstream Initiated Exit

The Root Complex or Switch (Downstream Port) is permitted to initiate an exit from L0s low power state on any of its transmit Links if it needs to communicate through the Link. The component initiates a transition to the L0 state on Lanes in the downstream direction as described in Chapter 4.

If the Downstream component is a Switch (i.e., it is not an Endpoint), it must initiate a transition on all of its Downstream Port transmit Lanes that are in L0s at that time as soon as it detects an exit from L0s on its Upstream Port. Links that are already in the L0 state do not participate in the exit transition. Links whose downstream component is in a low power state (i.e., D1-$D3_{hot}$ states) are also not affected by the exit transitions.

For example, consider a Switch with an upstream Port in L0s and a downstream device in a D1 state. A configuration request packet travels downstream to the Switch, intending to ultimately reprogram the downstream device from D1 to D0. The Switch's upstream Port Link will 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 into 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.

6.4.1.2. L1 Active State Link Power Management State

A component may optionally support the Active State Link PM 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 configuration status register defined in Section 5.6.

When supported, L1 entry is disabled by default in the Active State Link PM Control configuration field.

6.4.1.2.1. Entry to L1 State

An Endpoint enabled for L1 Active State Link PM entry may initiate entry into the L1 Link state.

Implementation Note: Initiating L1

This specification does not dictate when an Endpoint must initiate a transition to the L1 state on its transmit Lanes. The interoperable mechanisms for transitioning into and out of L1 are defined within this specification, however the specific Active State Link PM 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 Link has been in the L0s state for a set amount of time.

Three power management messages provide support for Active State Link Power Management of the L1 state:

- PM_ Active_State_Request_L1 (DLLP)
- PM_ Request_ACK (DLLP)
- PM_Active_State_Nak (TLP)

Endpoints that have their Active State Link PM L1 entry enabled negotiate for the resultant L-state with the component on the upstream end of the Link. If the endpoint receives a negative acknowledgement in response to its issuance of a PM_ Active_State_Request_L1 DLLP, then the endpoint must enter the L0s state as soon as possible[36]. Note that the component on the upstream side of the Link may not support L1 Active State Link PM, or it may be disabled and so could legitimately respond to the L1 entry request with a negative acknowledgement.

A Root Complex Root Port, or Switch Downstream Port must accept a request to enter a low power L1 state if all of the following conditions are true:

- The Port supports Active State Link PM L1 entry, and Active State Link PM L1 entry is enabled.
- 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 for an implementation specific set amount of time:

- The Upstream Port supports Active State Link PM L1 entry and it is enabled
- All of the Switch's Downstream Port Links are in the L1 state (or deeper)
- No pending TLPs to transmit
- No pending ACK, NAK, or ACK Timeout DLLPs to transmit
- The Upstream Port's receive Lanes are idle

If the Switch's upstream Port receives a negative acknowledgement in response to its issuance of a PM_Active_State_Request_L1 DLLP, then the Switch's upstream Port transmit Lanes must instead transition to the L0s state as soon as possible[37].

Note that it is legitimate for a Switch to be enabled for the Active State Link PM L1 Link state on any of its downstream ports and to be disabled or not even supportive of Active State Link PM L1 on its upstream Port. In that case, downstream ports may enter the L1 Link state, but the Switch will never initiate an Active State Link PM L1 entry transition on its upstream Port.

---

[36] Assuming that the conditions for L0s entry are met.

[37] Assuming that the conditions for L0s entry are met.

Active State Link PM L1 Negotiation Rules (see Figure 6-5 and Figure 6-6)

- Upon deciding to enter a low power Link state, the downstream component must block scheduling of any TLPs (including completion packets).
- The downstream component must wait until it receives a Link layer acknowledgement for the last TLP it had previously sent. The component may retransmit a TLP if required by the Link layer rules.
- The downstream component must also wait until it accumulates at least the minimum number of credits required to send the largest possible packet for any FC type. Note that this is required so that the component can immediately issue a TLP after it exists the L1 state.
- The downstream component then initiates Active State Link PM negotiation by sending a PM_Active_State_Request_L1 DLLP onto its transmit Lanes. The downstream component sends this DLLP continuously until it receives a response from the upstream device (see below). The downstream component remains in this loop waiting for a response from the Upstream Agent.
    - During this waiting period, the downstream component must not initiate any Transaction Layer transfers. It must still accept TLPs and DLLPs from the upstream component. It also responds with DLLPs as needed by the Link layer protocol.
    - If the Downstream component for any reason needs to initiate a transfer on the Link, it must first complete the transition to the low power Link state. Once in a lower power Link state, the downstream component is then permitted to exit the low power Link state to handle the transfer.
- The Upstream component must immediately respond to the request with either an acceptance or a rejection of the request.

Rules in case of rejection:

- In the case of a rejection, the upstream component must schedule, as soon as possible, a rejection (NAK) by sending the PM_Active_State_Nak Message to the downstream requesting agent. Once the PM_Active_State_Nak Message is sent, the upstream component is permitted to initiate any TLP or DLLP transfers.
- If the request was rejected, the downstream component must immediately transition its transmit Lanes into the L0s state, provided that conditions for L0s entry are met.

Rules in case of acceptance:

- If the upstream agent is ready to accept the request, it must block scheduling of any TLPs.
- The upstream component then must wait until it receives a Link layer acknowledgement for the last TLP it had previously sent. The upstream component may retransmit a TLP if required by the Link layer rules.

- The upstream component must also wait until it accumulates at least the minimum number of credits required to send the largest possible packet for any FC type. Note that this is required so that the component can immediately issue a TLP after it exists the L1 state.
- Once all TLPs have been acknowledged and enough FC credits accumulated, the upstream component sends a PM_Request_Ack DLLP downstream. The upstream component sends this DLLP continuously until it observes its receive Lanes enter into the electrical idle state. See Chapter 4 for more details on the physical layer behavior.
- If the Upstream component needs, for any reason, to initiate a transfer on the Link after it sends a PM_Request_Ack DLLP, it must first complete the transition to the low power state. It is then permitted to exit the low power state to handle the transfer once the Link is back to L0.
- When the downstream component detects a PM_Request_Ack DLLP on its receive Lanes (signaling that the upstream device acknowledged the transition to L1 request), the downstream component then ceases sending the PM_Active_State_Request_L1 DLLP, disables its Link layer 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 its Link layer 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.

323

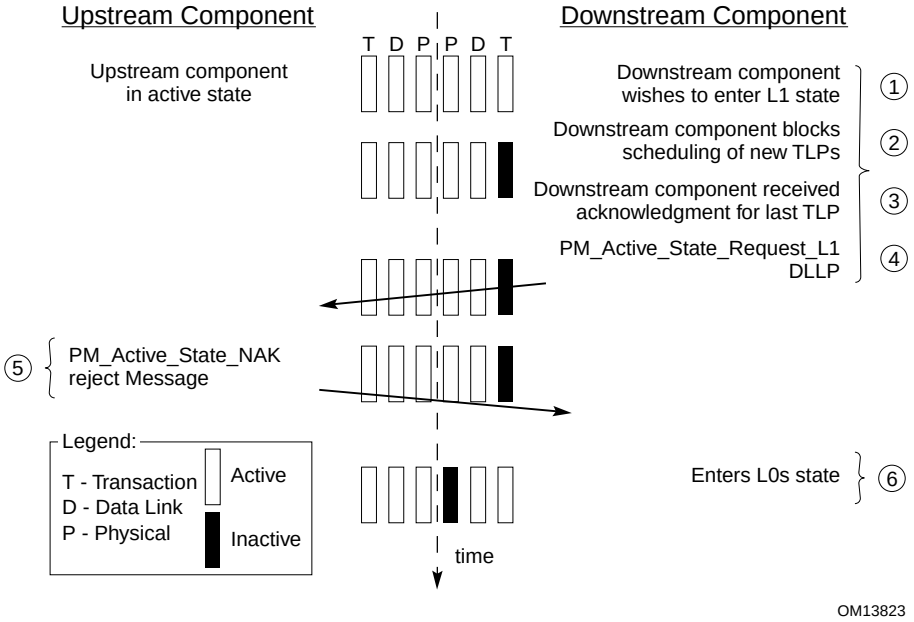

Figure 6-5: L1 Transition Sequence Ending with a Rejection

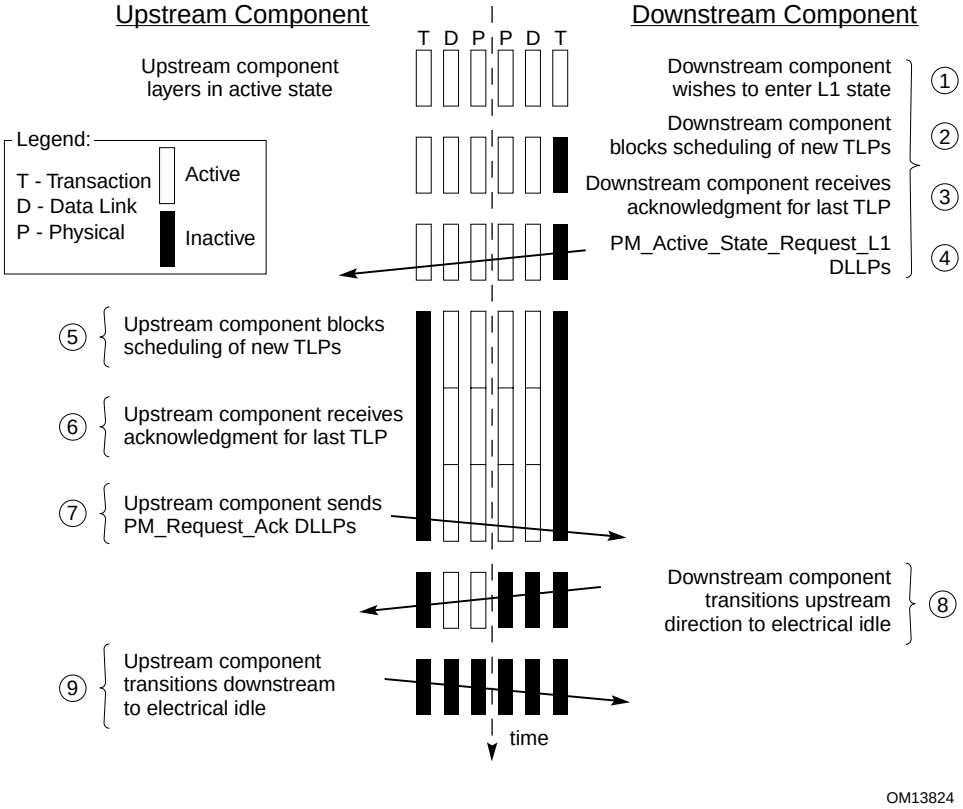

Figure 6-6: L1 Successful Transition Sequence

6.4.1.2.2. Exit from L1 State

Components on either end of a PCI Express Link may initiate an exit from the L1 Link state. Unlike the L1 entry protocol, where both ends negotiate for the resultant Link state, L1 exit does not require any negotiation.

Downstream Initiated Exit

An Endpoint or Switch Upstream Port is permitted to initiate an exit from L1 on its transmit Lanes if it needs to communicate through the Link. The component initiates a transition to the L0 state as described in Chapter 4. The Upstream component must respond by initiating a similar transition of its transmit Lanes.

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 process 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 though several PCI Express switches to experience accumulating latency as it traversed 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 6-7. 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 L1 state on its Upstream Port Link (Link 2) after no more than 1 µs from the beginning of the transition from 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 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, between Links 1, 2, and 3, the Link to complete the transition to 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.

325

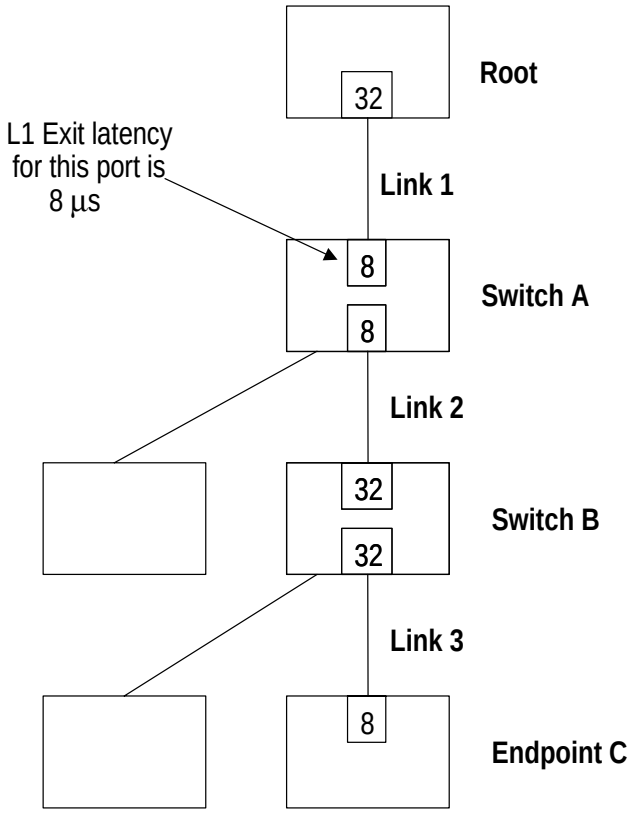

Figure 6-7: Example of L1 Exit Latency Computation

Switches are not required to initiate an L1 exit transition on other of its Downstream Port Links.

Upstream Initiated Exit

A Root Complex, or a Switch is permitted to initiate an exit from L1 on any of its Root Ports, or Downstream Port Links if it needs to communicate through that Link. The component initiates a transition to the L0 state as described in Chapter 4. The Downstream component must respond by initiating a similar transition on its transmit Lanes.

If the Downstream component is a Switch (i.e., it is not an Endpoint), it must initiate a transition on all of its Downstream Links (assuming the Downstream Link is in an Active State Link Power Management 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 had fully exited to L0 before initiating an L1 exit transition on its Downstream Port Links. If that were the case, a message traveling though multiple PCI Express switches would experience accumulating latency as it traverses each Switch.

A Switch is required to initiate a transition from L1 state on all of its Downstream Port Links that are currently in L1 after no more than 1 µs from the beginning of a transition from L1 state on its Upstream Port. Refer to Section 4.2 for details of the Physical Layer signaling during L1 exit. Downstream Port Links that are already in the L0 state do not participate in the exit transition. Downstream Port Links whose downstream component is

in a low power D-state (D1-D3hot) are also not affected by the L1 exit transitions (i.e., such Links must not be transitioned to the L0 state).

6.4.1.3. Active State Link PM Configuration

All PCI Express functions must implement the following configuration bits in support of Active State Link PM. Refer to Chapter 5 for configuration register assignment and access mechanisms.

Each PCI Express component reports its level of support for Active State Link Power Management in the Active State Link PM Support configuration field below. All PCI Express components must support transition to the L0s Link state. Support for transition to the L1 Link state while in $D0_{active}$ state is optional.

Table 6-3: Encoding of the Active State Link PM Support Field

Field	Read/ Write	Default Value	Description
Active State Link PM Support	RO	must be 01b or 11b	00b – Reserved 01b – L0s supported 10b – Reserved 11b – L0s and L1 supported

Each PCI Express component reports the source of its reference clock in its "Slot Clock Configuration bit" located in its PCI Express Capability Structure's Link Status Register.

Table 6-4: Description of the Slot Clock Configuration Field

Field	Read/ Write	Default Value	Description
Slot Clock Configuration	RO	HWInit	This bit indicates that the component uses the same physical reference clock that the platform provides on the connector. If the device uses an independent clock irrespective of the presence of a reference on the connector, this bit must be clear. For root and switch downstream ports, this bit when set, indicates that the downstream port is using the same reference clock as the downstream device or the slot. For switch and bridge upstream ports, this bit when set, indicates that the upstream port is using the same reference clock that the platform provides. Otherwise it is clear.

Each PCI Express component must support the Common Clock Configuration bit in their PCI Express Capability Structure's Link Command 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 6-5: Description of the Common Clock Configuration Field

Field	Read/ Write	Default Value	Description
Common Clock Configuration	RW	0	This bit when set indicates that this component and the component at the opposite end of the Link are operating with a common clock source. A value of 0 indicates that this component and the component at the opposite end of the Link are operating with separate reference clock sources. Default value of this field is 0. Components utilize this common clock configuration information to report the correct L0s and L1 Exit Latencies.

Each PCI Express component reports the L0s and L1 exit latency (the time that they require to transition their transmit Lanes from the L0s or L1 state to the L0 state) in the L0s Exit Latency and the L1 Exit Latency configuration fields, respectively.

Table 6-6: Encoding of the L0s Exit Latency Field

Field	Read/ Write	Default Value	Description
L0s Exit Latency	RO	N/A	000b – Less than 64 ns 001b – 64 ns-128 ns 010b – 128 ns-256 ns 011b – 256 ns-512 ns 100b – 512 ns-1 μs 101b – 1 μs-2 μs 110b –2 μs-4 μs 111b – Reserved

Table 6-7: Encoding of the L1 Exit Latency Field

Field	Read/ Write	Default Value	Description
L1 Exit Latency	RO	N/A	000b – Less than 1 µs 001b – 1 µs-2 µs 010b – 2 µs-4 µs 011b – 4 µs-8 µs 100b – 8 µs-16 µs 101b – 16 µs-32 µs 110b – 32 µs-64 µs 111b – L1 transition not supported

Endpoints also report the additional latency that they can absorb due to the transition from L0s state or L1 state to the L0 state. This is reported in the Endpoint L0s Acceptable Latency and Endpoint L1 Acceptable Latency fields, respectively.

Power management software, using the latency information reported by all components in the PCI Express Hierarchy, can enable the appropriate level of Active State Link Power Management by comparing exit latency for each given path from root to Endpoint against the acceptable latency that each corresponding Endpoint can withstand.

Table 6-8: Encoding of the Endpoint L0s Acceptable Latency Field

Field	Read/ Write	Default Value	Description
Endpoint L0s Acceptable Latency	RO	N/A	000b – Less than 64 ns 001b – 64 ns-128 ns 010b – 128 ns-256 ns 011b – 256 ns-512 ns 100b – 512 ns-1 µs 101b – 1 µs-2 µs 110b – 2 µs-4 µs 111b – More than 4 µs

Table 6-9: Encoding of the Endpoint L1 Acceptable Latency Field

Field	Read/ Write	Default Value	Description
Endpoint L1 Acceptable Latency	RO	N/A	000b – Less than 1 µs 001b – 1 µs-2 µs 010b – 2 µs-4 µs 011b – 4 µs-8 µs 100b – 8 µs-16 µs 101b – 16 µs-32 µs 110b – 32 µs-64 µs 111b – More than 64 µs

Power management software enables (or disables) Active State Link Power Management in each component by programming the Active State Link PM Control field.

Table 6-10: Encoding of the Active State Link PM Control Field

Field	Read/ Write	Default Value	Description
Active State Link PM Control	R/W	00b	00b – Disabled 01b – L0s Entry Enabled 10b – Reserved 11b – L0s and L1 Entry enabled

Active State Link PM Control = 00

Port must not bring a Link into L0s state.

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 L1 request must respond with negative acknowledgement.

Active State Link PM Control = 01

Port must bring a Link into L0s state if all conditions are met.

Ports connected to the Downstream end of the Link must not issue a PM_Active_State_Request_L1 DLLP on its Upstream Link.

Ports connected to the Upstream end of the Link receiving L1 request must respond with negative acknowledgement.

Active State Link PM Control = 11

Port must bring a Link into 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 L1 request and transition into L1 if conditions for Root Complex Root Port or Switch downstream Port in Section 6.4.1.2.1 are met.

6.4.1.3.1. Software Flow for Enabling Active State Link Power Management

Following is an example software algorithm that highlights how to enable active state Link power management in a PCI Express component.

- PCI Express components power up with active state Link power management disabled
- 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, then they are both using the same reference clock source, otherwise 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
- Devices must reflect the appropriate L0s/L1 exit latency in their "L0s/L1 exit latency register bits," per the setting of the "Common Clock Configuration" bit
- PCI Express system software then reads and adds up the L0s/L1 exit latency numbers from all components on a given PCI Express hierarchy reaching up to each endpoint component
- For each endpoint component, PCI Express system software examines the "Endpoint L0s/L1 Acceptable Latency," as reported by the endpoint component in their Link Capabilities register, and enables (or leaves disabled) L0s/L1 entry (via the Active State Lin PM Control bits in the Link Control register) accordingly in some or all of the intervening device ports on that hierarchy

6.5. Auxiliary Power Support

6.5.1. Auxiliary Power Enabling

The PCI-PM specification requires that a function must support PME generation in order to consume the maximum allowance of auxiliary current (375 mA vs. 20 mA). However, there are instances where functions need to consume power even if they are "PME Disabled," or PME incapable by design. One example is a component with its system management mode active during a system low power state.

PCI Express PM provides a new control bit, "Aux_En," that provides the means for enabling a function to draw the maximum allowance of auxiliary current independent of its level of support for PME generation.

A PCI Express function requests aux power allocation by specifying a non-zero value in the Aux_Current field of the Power Management Capabilities Register (PMC). Refer to Chapter 5 for the Aux_En register bit assignment, and access mechanism.

Legacy PCI-PM software is unaware of this new bit and will only be able to enable aux current to a given function based on the function's reported PME support, the Aux_Current field value and the function's PME_Enable bit.

Allocation of aux power using Aux_En is determined as follows:

Aux_En=1b:

Aux power is allocated as requested in the Aux_Current field of the Power Management Capabilities Register (PMC), independent of the PME_En bit in the Power Management Control/Status Register (PMCSR). The PME_En bit still controls the ability to master PME.

Aux_En = 0b:

Aux power allocation is controlled by the PME_En bit as defined in the PCI-PM specification.

The Aux_En bit is sticky meaning that its state is not affected by transitions from the $D3_{cold}$ to the $D0_{Uninitilaized}$ state.

6.6. Power Management System Messages and DLLPs

Table 6-11 defines the location of each PM packet in the PCI Express stack.

Table 6-11: Power Management System Messages and DLLPs

Packet	Type
PM_Enter_L1	DLLP
PM_Enter_L23	DLLP
PM_Active_State_Request_L1	DLLP
PM_Request_Ack	DLLP
PM_Active_State_Nak	Transaction Layer message
PM_PME	Transaction Layer message
PME_Turn_Off	Transaction Layer message
PME_TO_Ack	Transaction Layer message

6.6.1. Power Management System Messages

Power management messages follow the general rules for PCI Express system messages. Message fields follow the following rules:

- Length Field is reserved.
- Attribute Field must be set to the default values (all 0's).
- Address Filed is reserved.
- Requester ID
    - PM_PME message
        - Endpoints report their upstream Link bus number and the device and function number where the PME originated.
        - PCI Express to PCI Bridges - When the PME comes from a legacy agent on a PCI bus downstream, then the PM_PME Message requester ID reports the legacy bus number where the PME originated from, and the device and function number reported must both be zero. When the PCI Express-PCI bridge initiates an internal PME message (e.g., at time when hot plug event comes in and the SHPC is in non-D0 state), the requester ID is the bus number associated with that function. The device # and function # are whatever internal function needs to be awakened, e.g., the SHPC function in this example. In the example depicted in Figure 6-8, a PME is generated by the SHPC function. The requester ID in the PM_PME message contains bus = 7, device = 1, function = 1.

333

- All other messages report their upstream Link bus number, and device and function number must both be zero.

- Virtual Channel Field must use the default virtual channel (VC0)

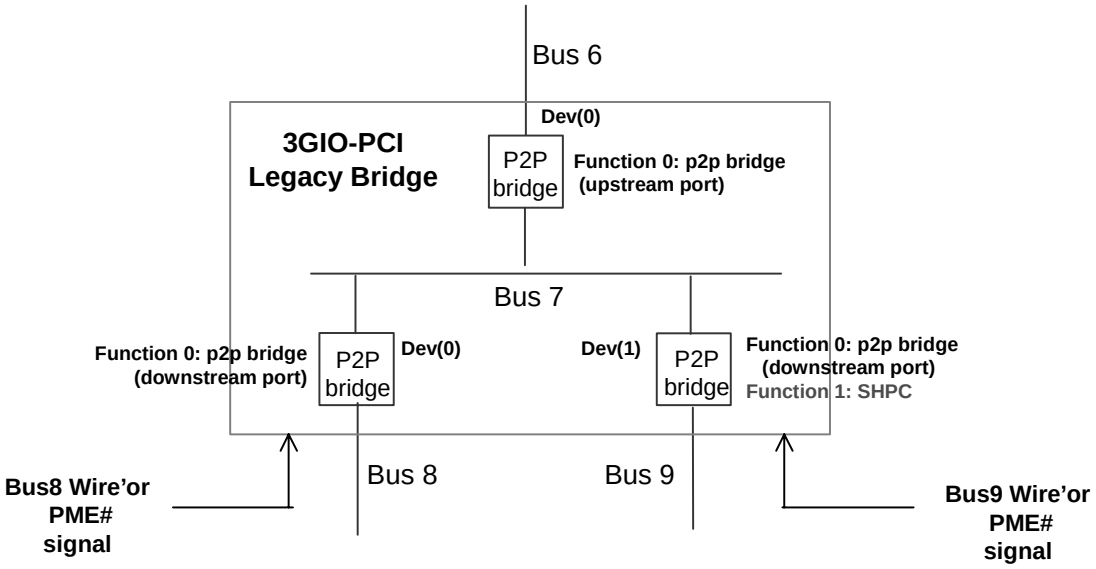

Figure 6-8: Example of PME Message Addressing in a PCI Express-to-PCI Bridge

6.6.2. Power Management DLLPs

For information on the structure of the power management DLLPs, refer to Section 3.4.

7

7. PCI Express System Architecture

This chapter addresses various aspects of PCI Express interconnect architecture in a platform context. It covers the details of interrupt support, error signaling and logging, VCs, isochronous support, Hot Plug, and Lock.

7.1. Interrupt Support

The PCI Express interrupt model supports two mechanisms:

- INTx emulation
- Message Signaled Interrupt (MSI) Support.

For legacy compatibility, PCI Express provides a PCI INTx emulation mechanism to signal interrupts to the system interrupt controller (typically part of the system core-logic). This mechanism is compatible with existing PCI software, and provides the same level and type of service as corresponding PCI interrupt signaling mechanism and is independent of system interrupt controller specifics. This legacy compatibility mechanism allows boot device support without requiring complex BIOS-level interrupt configuration/control service stacks. It virtualizes PCI physical interrupt signals by using an in-band signaling mechanism.

In addition to PCI INTx compatible interrupt emulation, PCI Express requires support of Message Signaled Interrupt (MSI) mechanism. The PCI Express MSI mechanism is compatible with the MSI capability defined in the PCI 2.3 Specification.

7.1.1. Rationale for PCI Express Interrupt Model

PCI Express takes an evolutionary approach from PCI with respect to interrupt support.

As required for PCI/PCI-X interrupt mechanisms, each device is required to differentiate between INTx (legacy) and MSI (native) modes of operation. The PCI Express device complexity required to support both schemes is no different than that for PCI/PCI-X devices today. The advantages of this approach include:

- Compatibility with existing PCI software models
- Direct support for boot devices
- Easier End of Life (EOL) for INTx legacy mechanisms.

Existing software model is used to differentiate legacy (INTx) vs. MSI modes of operation; thus, no special software support is required for PCI Express.

7.1.2. PCI Compatible INTx Emulation

PCI Express supports the PCI interrupts as defined in the PCI Specification, rev. 2.3 including the Interrupt Pin and Interrupt Line registers of the PCI configuration space for PCI devices. PCI Express devices support these registers for backwards compatibility; however, interrupts are asserted using in-band messages in the form of Transaction Layer Packets (TLPs) rather than asserting physical pins.

PCI Express defines two message Transactions, 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 routed to the Root Complex where the Requester ID information (included in all requestor packets) enables flexibility in mapping device interrupts to the system interrupt controller. PCI Express devices must use assert/de-assert messages in pairs to emulate PCI interrupt level-triggered signaling. Actual mapping of PCI Express INTx emulation to system interrupts is implementation specific as is mapping of physical interrupt signals in PCI today.

The legacy INTx emulation mechanism may be depreciated in a future version of this specification.

7.1.3. INTx Emulation Software Model

The software model for legacy INTx emulation matches that of PCI. The system BIOS reporting of chipset/platform interrupt mapping and the association of a device's interrupt with PCI interrupt lines is handled in exactly the same manner as with previous PCI systems. Legacy software reads from the device's Interrupt Pin register to determine if the device is interrupt driven. A value between 01 and 04 indicates that the device uses interrupt pin to generate an interrupt.

Note that similarly to physical interrupt signals, the INTx emulation mechanism may potentially cause spurious interrupts that must be handled by the system software.

7.1.4. Message Signaled Interrupt (MSI) Support

The Message Signaled Interrupt (MSI) capability is defined in the PCI 2.3 Specification.

MSI interrupt support, which is optional for PCI 2.3 devices, is required for PCI Express devices. MSI-capable devices deliver interrupts by performing memory write transactions. MSI is an edge-triggered interrupt; interrupt sharing is prohibited when using MSI.

Neither the PCI 2.3 Specification nor this specification support level-triggered MSI interrupts.

Note that, unlike INTx emulation messages, MSIs are not restricted to TC0.

7.1.5. MSI Software Model

It is the system software's responsibility to ensure that multiple MSI-capable devices cannot generate the same interrupt message.

It is implementation specific whether an interrupt message is accepted or potentially lost when an interrupt with the same interrupt message vector is already in service. Additional interrupt messages with same vector may be potentially lost depending on the specifics of the core-logic (i.e., chipset) and system interrupt controller implementation.

MSI-capable devices that require servicing of every interrupt message must not generate multiple outstanding interrupt messages with the same vector. These devices must not generate another interrupt message until the device driver indicates that the previous interrupt message of the same vector was serviced. The device driver might indicate that an interrupt is serviced by reading the device's interrupt status register.

For particular usage model it might be acceptable to generate a new MSI message without requiring the device driver to acknowledge the previous interrupt message. A common example occurs with timers (e.g., timer interrupts). There is no guarantee that all the interrupt messages from such a device will be serviced. If all interrupt events must be recognized in a deterministic manner, devices that are source of interrupts must not generate successive MSIs without having an explicit acknowledgement that each MSI has been detected and serviced. This explicit acknowledgement is typically supported by interrupt handler software reading or writing to a particular internal status or control register of interrupting device. Details of this "handshake" mechanism such as using either a read or write synchronizing operation and the location and type of address space (Memory, I/O, or Configuration space) of a control/status register, are implementation specific. Note, however, that for the purpose of supporting synchronization between hardware (interrupt source) and software (interrupt handler), it is recommended to use memory-mapped register locations.

Certain PCI devices and their drivers rely on INTx-type level-triggered interrupt behavior (addressed by the PCI Express legacy INTx emulation mechanism). These devices and their drivers must be redesigned to take advantage of the MSI capability and edge-triggered interrupt semantics.

7.1.6. 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 6.

7.1.7. PME Software Model

From a software standpoint, PME behaves like an edge-triggered interrupt. This is different from the level-triggered PME mechanism used for PCI. However, this does not impact operating system software compatibility as PME reporting to the operating system is abstracted by the ACPI BIOS. Current ACPI-compatible operating systems support both edge-triggered and level-triggered modes for PME.

To signal PME, a PCI Express device generates a PME message on the PCI Express Link. System power management logic that is typically part of the core-logic (chipset), receives this PME message and asserts a ACPI General Purpose Event (GPE) corresponding to the PME message. ACPI ASL code may utilize the PCI Express Requestor ID in the PM_PME to inform the operating system which device caused the wake.

Note that PCI Express architecture guarantees that the PME message is delivered reliably since PCI Express Messages are communicated using TLPs. For more details on PME signaling, see Chapter 6 (Power Management) of this specification.

7.1.8. PME Routing Between PCI Express and PCI Hierarchies

PME-capable PCI devices assert the PME# pin to signal a power management event. The physical PME signal from PCI devices may either be converted to PCI Express a in-band PME message by a PCI Express-PCI bridge or routed directly to a GPE pin on the core logic chipset. Delivery of PME signaling from PCI devices is implementation specific as it is in PCI-based systems today. When converting from PCI level-triggered PME signaling to edge-triggered PCI Express PME messages, care must be taken not to lose any PMEs from PCI devices. Such a conversion mechanism may also result in spurious PMEs being generated.

7.2. 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.

7.2.1. Scope

This section explains the error signaling and logging requirements for PCI Express components. This includes errors which occur on the PCI Express interface itself and those errors which occur on behalf of transactions initiated on PCI Express. This section does not focus on errors which occur within the component that are unrelated to a particular PCI Express transaction. This type of error signaling is better handled through proprietary methods employing device-specific interrupts.

PCI Express defines two error reporting paradigms: the baseline capability and the Advanced Error Reporting capability. The baseline error reporting capabilities are required of all PCI Express devices and define the minimum error reporting requirements. The Advanced Error Reporting capability is defined for more robust error reporting and is implemented with a specific PCI Express capability structure (refer to Chapter 5 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.

7.2.2. 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.

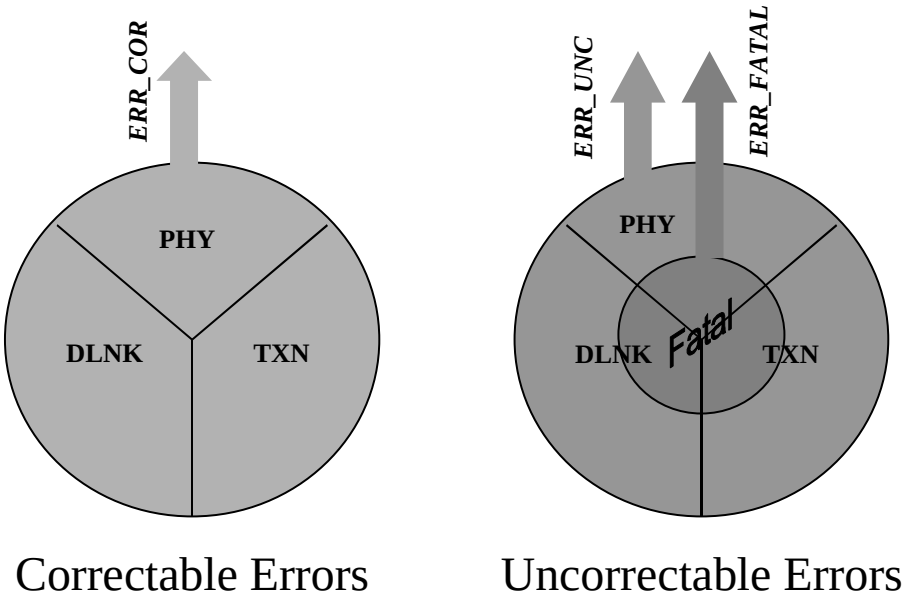

Figure 7-1: Error Classification

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.

7.2.2.1. Correctable Errors

Correctable errors include those error conditions where the PCI Express protocol 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 which is corrected by Data Link Level Retry is considered a correctable error. Logging the frequency of correctable errors may be helpful for profiling the integrity of a Link.

7.2.2.2. Uncorrectable Errors

Uncorrectable errors are those error conditions that impact functionality of the interface. There is no PCI Express mechanism defined in this specification to correct these errors. For more robust error handling by the system, PCI Express further classifies uncorrectable errors as Fatal and Non-fatal.

7.2.2.2.1. Fatal Errors

Fatal errors are uncorrectable error conditions which render the particular PCI Express Link and related hardware unreliable. For fatal errors, a reset of the Link may be required to return to reliable operation. Platform handling of fatal errors, and any efforts to limit the effects of these errors, is platform implementation specific.

Comparing with PCI/PCI-X, reporting a fatal error is somewhat analogous to asserting SERR#.

7.2.2.2.2. Non-Fatal Errors

Non-fatal errors are uncorrectable errors which cause a particular transaction to be unreliable but the Link is otherwise fully functional. Isolating non-fatal from fatal errors provides system management software the opportunity to recover from the error without resetting the Link(s) and disturbing other transactions in progress. Devices not associated with the transaction in error are not impacted by the error.

7.2.3. Error Signaling

There are two complementary mechanisms in PCI Express which allow the agent detecting an error to alert the system or the initiating device that an error has occurred. The first mechanism is through a Completion Status and the second method is with in-band error messages.

Note that it is the responsibility of the agent detecting the error to signal the error appropriately.

Section 7.2.5 enumerates all the errors and how the hardware is required to respond when the error is detected.

7.2.3.1. Completion Status

The Completion Status field in the Completion header indicates when the associated Request failed (refer to Section 2.7.5). This is the only method of error reporting in PCI Express which enables the Requestor to associate an error with a specific Request. In other words, since Non-Posted Requests are not considered complete until after the Completion returns, the Completion Status field gives the initiator an opportunity to "fix" the problem at some higher level protocol (outside the scope of this specification). For example, if a Read is issued to prefetchable memory space and the Completion returns with a Unsupported Request Completion Status, perhaps due to a temporary condition, the initiator may choose to reissue the Read Request without side effects. Note that from a PCI Express point of view, the reissued Read Request is a distinct Request, and there is no relationship (on PCI Express) between the first Request and the reissued Request.

7.2.3.2. Error Messages

Error messages are sent to the Root Complex for reporting the detection of errors according to the severity of the error.

When multiple errors of the same severity are detected, the corresponding error messages may be merged for different errors of the same severity. At least one error message must be sent for detected errors of each severity level.

Table 7-1: Error Messages

Error Message	Description
ERR_COR	This Message is issued when the component or device detects a correctable error on the PCI Express interface. Refer to Section 7.2.2.1 for the definition of a correctable error.
ERR_NONFATAL	This Message is issued when the component or device detects a non-fatal, uncorrectable error on the PCI Express interface. Refer to Section 7.2.2.2.2 for the definition of a non-fatal, uncorrectable error.
ERR_FATAL	This Message is issued when the component or device detects a fatal, uncorrectable error on the PCI Express interface. Refer to Section 7.2.2.2.1 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.

7.2.3.2.1. Uncorrectable Error Severity Programming (Advanced Error Reporting)

For devices implementing the Advanced Error Reporting capability, the Uncorrectable Errors Severity register allows each uncorrectable error to be programmed to Fatal or Non-Fatal. Uncorrectable errors are not recoverable using defined PCI Express mechanisms. However, some platforms or devices might consider a particular error fatal to a Link or device while another platform considers that error non-fatal. The default value of the

Uncorrectable Errors Severity register serves as a starting point for this specification but the register can be reprogrammed if the device driver or platform software requires more robust error handling.

Baseline error handling does not support severity programming.

7.2.3.2.2. Masking Error Messages

Section 7.2.5 lists all the errors governed by this specification and enumerates when each of the above error messages are issued. For devices implementing the Advanced Error Reporting capability, each of the errors are captured in the Uncorrectable Error Status register or Correctable Error Status register. The Uncorrectable Errors Mask register and Correctable Errors Mask register allows each error condition to be masked independently.

For devices that do not implement the Advanced Error Reporting capability, errors are reported with the Error Status field of the Link Status register and masked with the Error Control field of the Link Command register (see Section 5.6).

When an error is masked, it is still logged but the error reporting Message is not sent to the Root Complex. Errors masked with the Link Command register are masked independent of the bit settings in the Uncorrectable Errors Mask register and Correctable Errors Mask register.

7.2.3.2.3. Error Pollution

Error pollution can occur if error conditions for a given transaction are not isolated to the error's first occurrence. For example, assume the Physical Layer detects a Receiver Error. This error is detected at the Physical Layer and an error is reported to the Root Complex. To avoid having this error propagate and cause subsequent errors at upper layers (for example, a TLP error at the Data Link Layer), making it more difficult to determine the root cause of the error, subsequent errors which occur for the same packet will not be signaled at the Data Link or Transaction layers. Similarly, when the Data Link layer detects an error, subsequent errors which occur for the same packet will not be signaled at the Transaction layer. This behavior applies only to errors which are associated with a particular packet – other errors are reported for each occurrence.

7.2.4. Error Logging

Section 7.2.5 lists all the errors governed by this specification and for each error, the logging requirements are specified. devices that do not support the Advanced Error Reporting capability log only the Link Status register bits indicating that a Correctable, Uncorrectable-Non-fatal, or Uncorrectable-Fatal error has occurred. Note that some errors are also reported using the reporting mechanisms in the PCI compatible (Type 00h and 01h) configuration registers. Section 5.5 describes how these register bits are affected by the different types of error conditions described in this section.

For devices supporting the Advanced Error Reporting capability, each of the errors in Table 7-2, Table 7-3: and Table 7-4 corresponds to a particular bit in the Uncorrectable Error Status register or Correctable Error Status register, except for Unsupported Request, which is covered in the PCI Express device registers directly (see Section 5.8). These registers are used by software to determine more precisely which error and what severity occurred. For many of the Transaction Layer errors the associated TLP Header is logged in the Header Log register. This helps system software to isolate errors to a particular application, and is useful for robust error handling by allowing system software to keep the remainder of the platform running normally.

7.2.4.1. Root Complex Considerations (Advanced Error Reporting)

In addition to the above logging, a Root Complex that supports the Advanced Error Reporting capability is required to implement the Error Source Identification register, which records the Requester ID of the first ERR_NONFATAL/ERR_FATAL (uncorrectable errors) and ERR_COR (correctable errors) messages received by the Root Complex. System software written to support Advanced Error Reporting can use the Root Port Error Status register to determine which fields hold valid information.

7.2.4.2. 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 occur on that particular PCI Express interface. The bits remain set until explicitly cleared by software or reset. Since multiple bits might be set in the Uncorrectable Error Status register, the First Error Pointer register points to the uncorrectable error that occurred first, except in the case where a non-fatal uncorrectable error is followed by a fatal error, in which case the information for the first fatal error is stored. Likewise, the TLP Header Log register stores the Header for the first occurrence of a particular severity error, but is replaced when a higher severity error is detected. For example: The TLP Header Log register is loaded due to a correctable error, a subsequent correctable error leaves the TLP Header Log register unmodified, but a following uncorrectable error causes the replacement of the original contents of the TLP Header Log register.

7.2.5. Error Listing and Rules

The tables below list all of the PCI Express errors which are defined by this specification. Each error is listed with a short-hand name, how the error is detected in hardware, the default severity of the error, and the expected action taken by the agent which detects the error. These actions form the rules for PCI Express error reporting and logging.

The Default Severity column specifies the default severity for the error without any software reprogramming. For devices supporting the Advanced Error Reporting capability, the uncorrectable errors are programmable to Fatal or Non-fatal with the Error Severity register. Devices without Advanced Error Reporting capability use the default associations and are not reprogrammable.

Table 7-2: Physical Layer Error List

Error Name	Default Severity	Detecting Agent Action
Receiver Error	Correctable	*Receiver (if checking)*: Send ERR_CORR to Root Complex unless masked.
Training Error	Uncorrectable (Fatal)	If checking, send ERR_FATAL/ERR_NONFATAL to Root Complex[38] unless masked

Table 7-3: Data Link Layer Error List

Error Name	Severity	Detecting Agent Action
Bad TLP	Correctable	*Receiver*: Send ERR_CORR to Root Complex unless masked. If the detecting agent supports the Advanced Error Reporting Capability, log the header of the TLP that encountered the error. Note that the header may be unreliable.
Bad DLLP		*Receiver*: Send ERR_CORR to Root Complex unless masked.
Replay Timeout		*Transmitter*: Send ERR_CORR to Root Complex unless masked.
REPLAY NUM Rollover		*Transmitter*: Send ERR_CORR to Root Complex unless masked.

[38] Only the component closer to the Root Complex is typically capable of sending the error Message.

Error Name	Severity	Detecting Agent Action
Data Link Layer Protocol Error	Uncorrectable (Fatal)	If checking, send ERR_FATAL/ERR_NONFATAL to Root Complex unless masked.

Table 7-4: Transaction Layer Error List

Error Name	Severity	Detecting Agent Action
Poisoned TLP Received	Uncorrectable (Non-Fatal)	*Receiver (if data poisoning is supported)*: Send ERR_NONFATAL/ ERR_FATAL to Root Complex unless masked. If the detecting agent supports the Advanced Error Reporting Capability, log the header of the poisoned TLP.
ECRC Check		*Receiver*: Send ERR_NONFATAL/ ERR_FATAL to Root Complex unless masked. If the detecting agent supports the Advanced Error Reporting Capability, log the header of the TLP that encounter the ECRC error.
Unsupported Request (UR)		*Request Receiver*: Send ERR_NONFATAL/ ERR_FATAL to Root Complex unless masked. If the detecting agent supports the Advanced Error Reporting Capability, log the header of the transaction that encountered the error.
Completion Timeout		*Requester*: Send ERR_NONFATAL/ERR_FATAL to Root Complex.
Completer Abort		*Completer (if device generates Completer Abort status)*: Send ERR_NONFATAL/ERR_FATAL to Root Complex.
Unexpected Completion		*Receiver*: Send ERR_NONFATAL/ERR_FATAL to Root Complex. If the detecting agent supports the Advanced Error Reporting Capability, log the header of the Completion that encountered the error. Note that if Unexpected Completion is a result of misrouting, the Completion Timeout mechanism will be triggered at the original Requester.
Receiver Overflow	Uncorrectable (Fatal)	*Receiver (if checking)*: Send ERR_FATAL/ERR_NONFATAL to Root Complex.

Error Name	Severity	Detecting Agent Action
Flow Control Protocol Error		*Receiver (if checking)*: Send ERR_FATAL/ERR_NONFATAL to Root Complex.
Malformed TLP		*Receiver*: Send ERR_FATAL/ERR_NONFATAL to Root Complex. If the detecting agent supports the Advanced Error Reporting Capability, log the header of the TLP that encountered the error.

7.2.5.1. PCI Mapping

In order to support PCI driver and software compatibility, PCI Express error conditions, where appropriate, must be mapped onto the PCI Status register bits for error reporting.

In other words, when certain PCI Express errors are detected, the appropriate PCI Status register bit is set alerting the error to legacy PCI software. While the PCI Express error results in setting the 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.

7.2.6. Real and Virtual PCI Bridge Error Handling

Virtual PCI Bridge configuration headers are associated with each PCI Express Port in a Root Complex or a Switch. Naturally, PCI/PCI-X Bridges also implement PCI Bridge configuration headers. For all of these cases, PCI Express error concepts require appropriate mapping to the PCI error reporting structures. This section addresses the cases related to the virtual PCI Bridge associated with PCI Express Ports in Root Complex and Switch cases. The mapping for PCI/PCI-X Bridges is similar, and is covered in detail elsewhere.

7.2.6.1. Error Forwarding and PCI Mapping for Bridge - Rules

In general, a TLP is either passed from one side of the Virtual PCI Bridge to the other, or is handled at the ingress side of the Bridge according to the same rules which apply to the ultimate recipient of a TLP. The following rules cover PCI Express specific error related cases:

- If a Request does not address a space mapped to the egress side of the Bridge, the Request is terminated at the ingress side as an Unsupported Request
- Poisoned TLPs are forwarded according to the same rules as non-poisoned TLPs
    - When forwarding a poisoned TLP:
        - the Receiving side must set the Detected Parity Error bit in the (Secondary) Status register
        - the Transmitting side must set the Master Data Parity Error bit in the Secondary Status register if the Parity Error Response bit in the Bridge Control register is set
- ERR_COR, ERR_NONFATAL and ERR_FATAL are forwarded from the secondary interface to primary interface, if the SERR# Enable bit in the Command and Bridge Control register is set

7.3. Virtual Channel Support

7.3.1. Introduction and Scope

Virtual Channel mechanism provides a foundation for supporting differentiated services within the PCI Express fabric. It enables deployment of independent physical resources that together with traffic labeling are required for optimized handling of differentiated traffic. Traffic labeling is supported using Transaction Class TLP-level labels. Exact policy for traffic differentiation is determined by the TC/VC mapping and by the VC-based arbitration. The TC/VC mapping depends on the platform application requirements. These requirements drive the choice of VC arbitration algorithm and configurability/programmability of arbiters allows detailed tuning of the traffic servicing policy.

Basic definition of Virtual Channel mechanism and associated Traffic Class labeling mechanism is covered in Chapter 2 of this specification. VC configuration/programming model is defined in Section 5.11 of this document.

The remaining sections of this chapter cover VC mechanisms from the system perspective. They address the next level details on:

- Supported TC/VC configurations
- VC-based arbitration – algorithms and rules
- Traffic ordering considerations
- Isochronous support as a specific usage model

7.3.2. Supported TC/VC Configurations

A Virtual Channel is established when one or multiple TCs are associated with a physical VC resource designated by the VC ID. Every Traffic Class that is supported must be mapped to one of the Virtual Channels. The baseline PCI Express configuration requires support for the default TC0/VC0 pair that is "hardwired" i.e., not configurable. Any support above that level is optional. The TC/VC configuration process is controlled by system software using programming model described in Section 5.11.

To simplify for interoperability when configuring multiple VCs over a PCI Express Link, this specification provides restricting rules to limit the set of valid VC configurations that can be found in Section 2.6. In general, mapping of TCs to VCs other than TC0/VC0 is up to system software. Two basic TC/VC configurations are described here as examples:

- Symmetrical TC to VC Mapping
- TC to VC Re-mapping

Note that multi-port components (Switches and Root Complex) are required to support independent TC/VC mapping for each PCI Express port, therefore they must support both configurations.

7.3.2.1. Symmetrical TC to VC Mapping

Differentiated servicing of transactions with different TC labels can be realized through proper mapping of TCs to VCs that provide certain service disciplines. In many applications, same service discipline is applied to transactions with the same TC label regardless of the source of the transaction. Therefore, within a PCI Express fabric component such as a Switch, the setting of TC to VC mapping is selected such that it is the same for all ports of the Switch. This is called symmetrical TC to VC mapping.

Figure 7-2 shows a symmetrical TC to VC mapping example, where the Switch has two downstream ports and one upstream Port and supports four Virtual Channels at each Port. After VC configuration is established, the upstream Port and the downstream Port connecting to Endpoint B have four VCs enabled with VC ID of 0, 1, 2, 3, respectively and have the following TC to VC mapping: TC(0-1)/VC0, TC(2-4)/VC1, TC(5-6)/VC2, TC7/VC3. The connection to Endpoint A only has two VCs enabled with the following TC to VC mapping: TC(0-1)/VC0, TC7/VC3. In this example, the second VC (at the Link that connects Endpoint A and Switch) is assigned a VC ID of 3. In this configuration, all enabled VCs of Switch ports have the same set of associated TCs.

Note that TC[2:6] are not mapped to the Link that connects Endpoint A and Switch, which means that traffic labeled with TC[2:6] is not allowed between the Switch and Endpoint A. Traffic labeled with a TC number that is not in the list of TCs enabled for a PCI Express Port is treated as an illegal transaction. Corresponding packets will be dropped at the receiving Port. This mechanism is referred to as TC filtering.

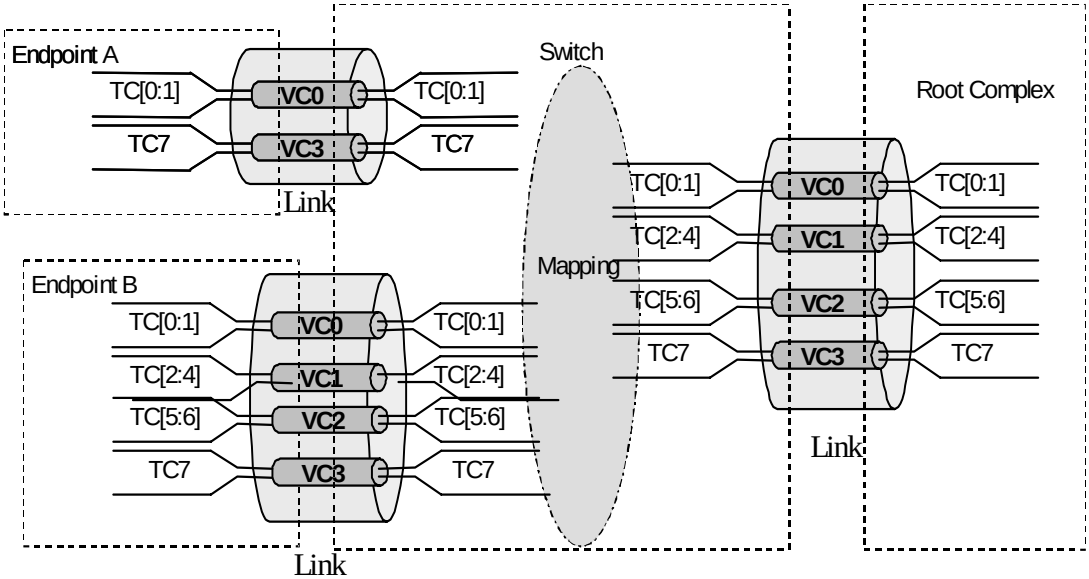

Figure 7-2: An Example of Symmetrical TC to VC Mapping

7.3.2.2. TC to VC Re-mapping

For some systems where PCI Express components with different capability are connected in a PCI Express fabric, an improved traffic differentiation can be achieved using TC to VC re-mapping. TC to VC re-mapping refers to the configuration of a multi-port PCI Express component whereas for traffic flowing from an Ingress Port to an Egress Port of the component, the TC to VC mapping of the two ports are different.

Figure 7-3 shows an example of TC to VC re-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 the default VC (VC0) is enabled and all TCs are mapped to VC0. In this example while TC7 is mapped to VC0 at the downstream Port, it is re-mapped to VC1 at the upstream Port. Although the Endpoint device only supports the default VC, when it labels transactions with different TCs, transactions with TC7 label from/to the Endpoint device can take advantage of the two Virtual Channels enabled between the Switch and the Root Complex.

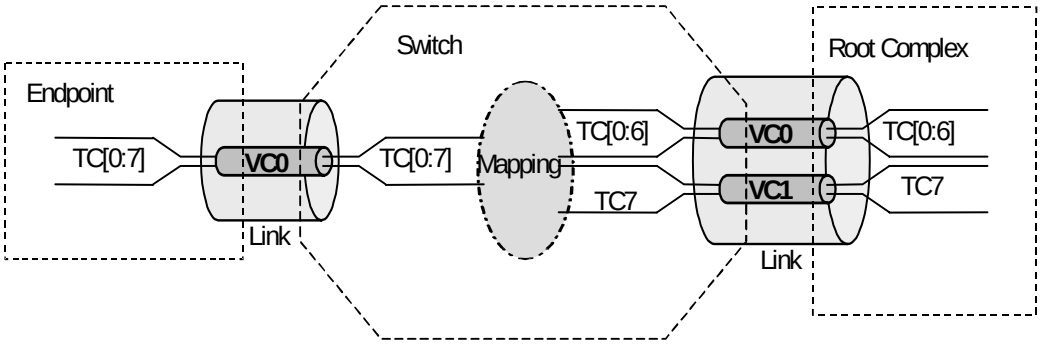

Figure 7-3: An Example of Asymmetrical TC to VC Mapping

Implementation Note: Multiple TCs Over a Single VC

A single VC implementation may benefit from using multiple TC labels. TC labels provide ordering domains that may be used to differentiate traffic within the endpoint or the RC independent of the number of VCs supported.

In a simple configuration there might be only a default VC supported. Within this platform the traffic differentiation may not be supported in an optimum manner since the different traffic classes cannot be physically segregated. However, the benefits of carrying multiple TC labels can still be exploited particularly in the small and "shallow" topologies where Endpoints are connected directly to RC rather then through cascaded switches. In these topologies traffic that is targeting RC only needs to traverse a single Link, and an optimized scheduling of packets on both sides (Endpoint and RC) based on TC labels may accomplish significant improvement over the case when a single TC label is used. Still, inability to route differentiated traffic through separate resources with fully independent flow-control and independent ordering exposes all of the traffic to the potential blocking head-of-line conditions. Optimizing Endpoint internal architecture to minimize the exposure to the blocking conditions can reduce those risks.

7.3.3. VC Arbitration

Arbitration is one of the key aspects of Virtual Channel mechanism and it is defined in a manner that fully enables configurability to the specific application. In general, definition of PCI Express VC-based arbitration mechanism is driven by the following objectives:

- To provide data flow forward progress required to avoid false transaction timeouts.
- 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.

As PCI Express Links are bidirectional, each PCI Express port can be an ingress or an Egress Port depending on the direction of traffic flow. This is illustrated by the example of a 3-port Switch in Figure 7-4, where traffic flows between Switch ports are highlighted with different types of lines. In the following sections, PCI Express VC Arbitration is defined

using a Switch arbitration model since Switch is the PCI Express element that represents a functional superset from the arbitration perspective.

In addition, one-directional data flows are used in the description.

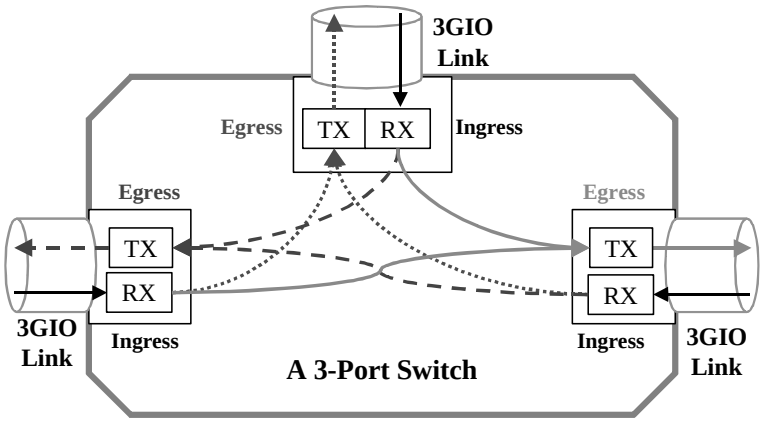

Figure 7-4: An Example of Traffic Flow Illustrating Ingress and Egress

7.3.3.1. Traffic Flow and Switch Arbitration Model

The following set of figures (Figure 7-5 and Figure 7-6) illustrates traffic flow through the Switch and summarizes the key aspects of the arbitration.

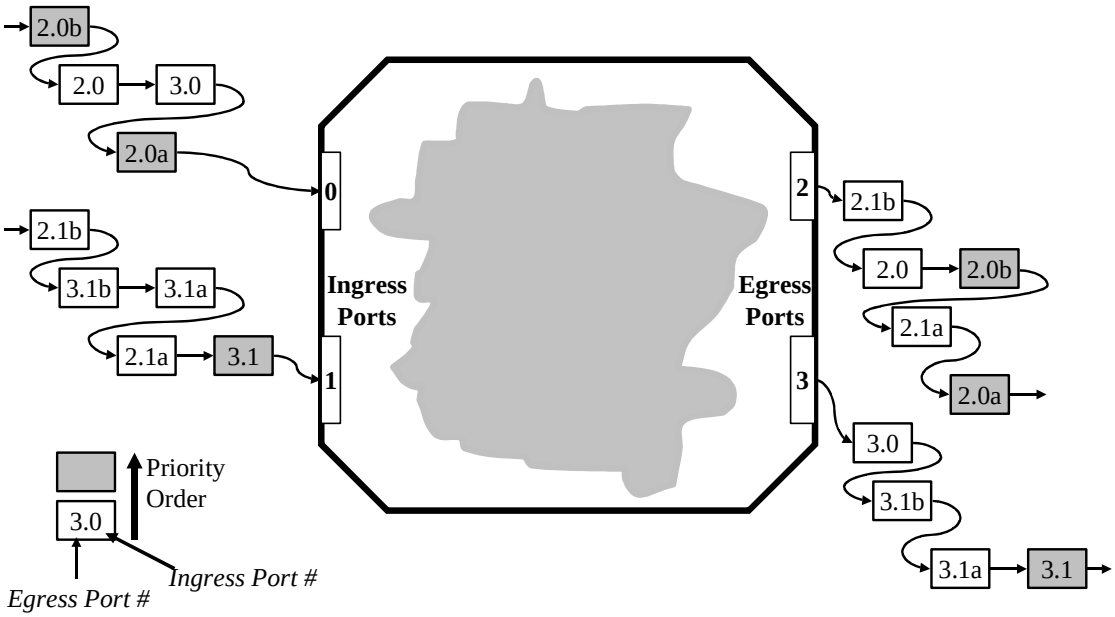

Figure 7-5: An Example of Differentiated Traffic Flow Through a Switch

At each Ingress Port an incoming traffic stream is represented in Figure 7-5 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. For example, designation "3.0" means that this packet is arriving at Port #0 (ingress) and it 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 7-6 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 a 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 of ports on the Switch. PCI Express architecture enables peer-to-peer communication through the Switch and, therefore, possible scenarios using the same example may include multiple separate and simultaneous ingress to egress flows (e.g., Port 0 to Port 2 and Port 1 to Port 3).

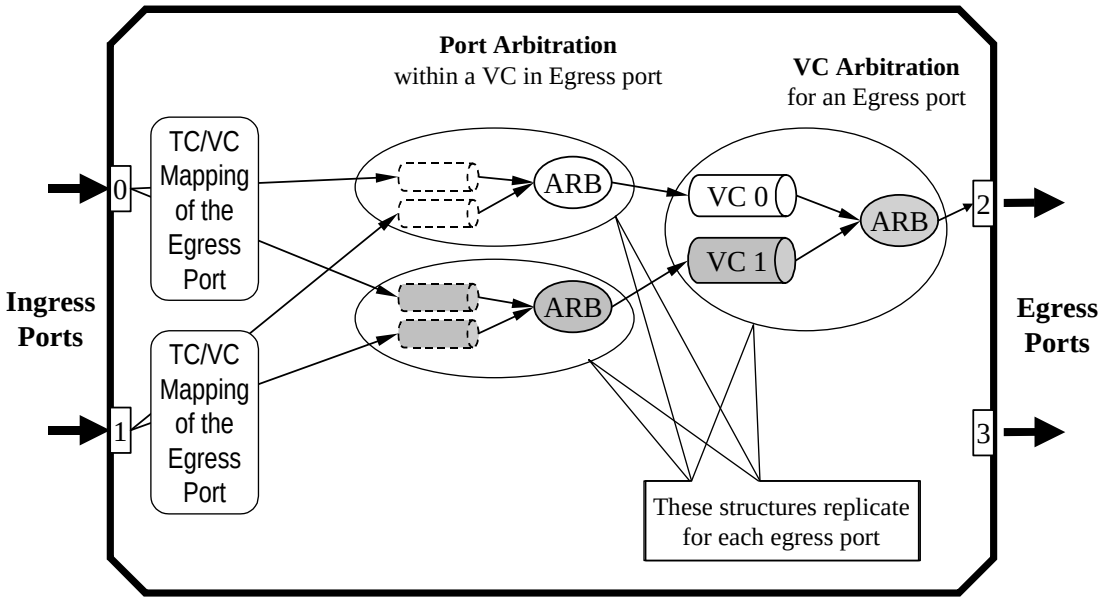

Figure 7-6: Switch Arbitration Structure

Routing of traffic received by the Switch on Port 0 and Port 1 and destined to Port 2 can be conceptually described by the following two steps. 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 Egress Port. This arbitration is referred to as the Port Arbitration.

Once traffic reaches corresponding destination VC resource in the Egress Port, it is subject of 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 Requirements at Egress Port

Part of the VC control logic resources at every 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., DLL 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 presence of pending packet to prevent blockage of traffic from other VCs. Note that since each VC includes buffering resources for Posted, Non-Posted Requests and Completion packets, the VC control logic must also take into account availability of flow control credits for the particular candidate packet. In addition, VC control logic must observe ordering rules (see Section 2.5 for more details) for Posted/Non-Posted/Completion transactions to prevent deadlocks and violation of producer-consumer ordering model.

Implementation Note: Arbitration for Multi-Function Endpoints

The arbitration of data flows from different functions of a multi-function Endpoint is beyond the scope of this specification. Mapping of different data flows (within multi-function Endpoint) to different TCs and VCs is implementation specific. Multi-function Endpoints, however, should support PCI Express VC-based arbitration control mechanism if multiple VCs are implemented for the PCI Express Link.

When a common VC on the PCI Express Link is shared by multiple functions, the aggregated traffic over the VC is subject to the bandwidth and latency regulations for that VC on the PCI Express Link. The multi-function Endpoints should implement proper arbitration for data flows from different functions in order to share the Link resources and achieve desired end to end services.

353

7.3.3.2. VC Arbitration – Arbitration Between VCs

The VC Identification (VC ID) provides an inherent, i.e., default "prioritization" of VCs. Therefore, all VC resources are arranged in ascending order of relative priority in the PCI Express Virtual Channel Capability Structure. As shown in an example in Figure 7-7 where 8 VCs are supported by a PCI Express Port, the VCs are associated with default priority levels where VC0 is a lowest priority and VC7 is the highest priority.

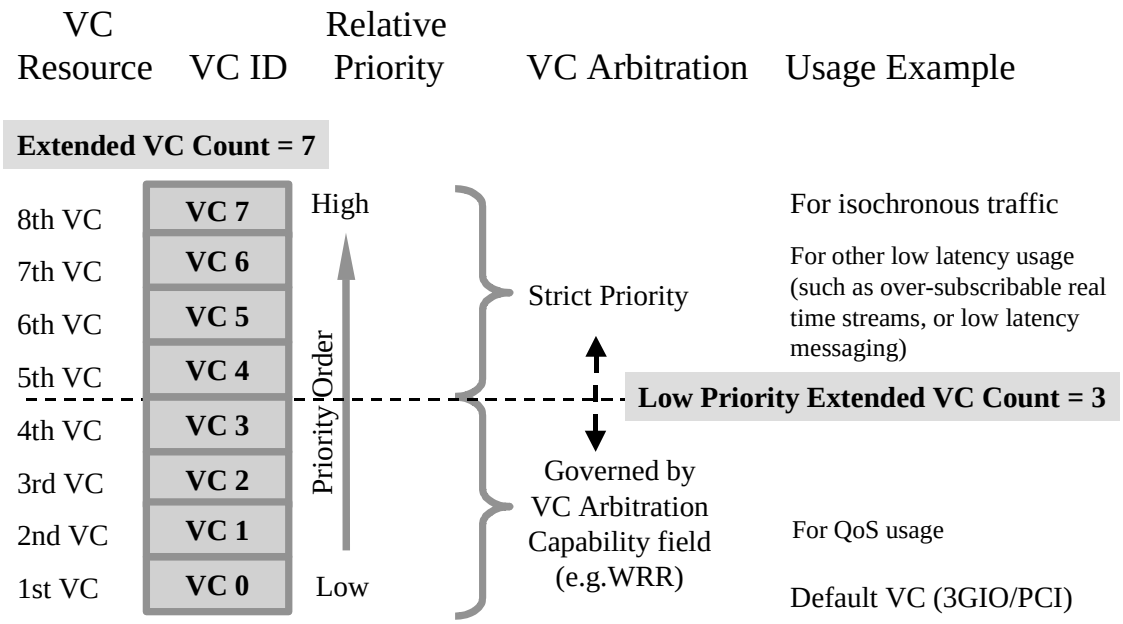

Figure 7-7: VC ID and Priority Order – An Example

However, this inherent prioritization does not imply restrictions in terms of algorithms that can be deployed for handling VC arbitration. Before VCs above the default VC0 can be enabled, they must be configured for the appropriate arbitration policy. PCI Express architecture defines the following arbitration 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

The PCI Express VC Capability programming model allows mixing of different arbitration methods by grouping VCs into the two groups, the lower group (VC0 to VC3) and the upper group (VC4 to VC7) as shown by the example in Figure 7-7. The upper group operates using strict priority scheme and the lower group as a whole is treated as the lowest priority member in the strict priority arbitration stack. The arbitration within the lower group can be configured to one of the supported arbitration methods. The size of this group is indicated by the Low Priority Extended VC Count field in the Port VC Capability Register 1. The arbitration methods are listed in the VC Arbitration Capability field in the Port VC Capability Register 2. See Section 5.11 for details. When the Low Priority Extended VC

Count field is set to zero, all VCs are governed by the strict-priority VC arbitration; when the field is equal to the Extended VC Count, all VCs are governed by the VC arbitration indicated by the VC Arbitration Capability field.

7.3.3.2.1. Strict Priority Arbitration Model

Strict priority provides minimal latency for high priority transactions. However, it may create a potential starvation for a low priority traffic if it is not applied correctly. Using strict priority scheme implies that traffic at every priority level (except at the lowest) is regulated in terms of both maximum peak bandwidth and duration of the peak bandwidth usage. This regulation must be provided at the sources of the traffic or at the ports where traffic enters the PCI Express fabric. It is assumed that lowest priority will be provided with an adequate leftover of bandwidth to allow reasonable forward progress and to prevent application timeouts. For example, isochronous traffic requires to be served as the highest priority by each PCI Express component. As detailed in Section 7.3.4, regulation of isochronous resource usage is managed by software and is enforced by PCI Express fabric components such as Switches and Root Complex in the manner that over-subscription is prevented.

7.3.3.2.2. Round Robin Arbitration Model

Round Robin model is used to provide simple arbitration that allows at transaction-level equal[39] 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 weight regulates 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 for WRR scheme is support for QoS policy where different QoS levels can be provided using different weights.

Although weight can be fixed (by hardware implementation) for certain applications, to provide more generic support for different applications PCI Express components that support WRR scheme are recommended to make it programmable. Programming of WRR is controlled using software interface defined in Section 5.11.

[39] Note that this does not imply equivalence and fairness in the terms of bandwidth usage.

7.3.3.3. Port Arbitration – Arbitration Within VC

Arbitration within VC refers to the arbitration between the traffic that is mapped onto the same VC but is coming from different Ingress Ports. Inherent prioritization scheme that makes sense when talking about arbitration among VCs in this context is not applicable since it would imply strict arbitration priority for different ports. Traffic from different ports can be arbitrated using the following supported schemes:

- Hardware-fixed Round Robin or RR-like arbitration scheme
- Programmable WRR arbitration scheme
- Programmable Time-based WRR arbitration scheme

Hardware-fixed RR or RR-like scheme is the simplest to implement since it does not require any programmability. It makes all ports equal priority, which is acceptable for applications where no software-managed differentiation or per-port-based bandwidth budgeting is required.

Programmable WRR allows flexibility that 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 7.3.3.2. 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 on the amount of traffic that can be injected from different ports within certain fixed period of time. This is required for certain applications such as isochronous where traffic needs to meet a strict deadline requirement. Section 7.3.4 provides basic rules to support isochronous applications. For more details on time-based arbitration and on the isochronous as a usage model for this arbitration scheme refer to Appendix A.

7.3.4. Isochronous Support

Servicing isochronous data transfer requires a system to provide not only guaranteed data bandwidth but also deterministic service latency. The isochronous support mechanisms in PCI Express are defined to ensure that isochronous traffic receives its allocated bandwidth over a relevant period of time while also preventing starvation of the other traffic in the system. Isochronous support mechanisms apply to communication between Endpoint and Root Complex as well as to peer-to-peer communication with the following restrictions:

- In the Endpoint to Root Complex communication model, isochronous traffic consists of read and write requests to the Root Complex and read completions from the Root Complex.
- In the Peer-to-Peer model, isochronous traffic is limited to unicast push-only transactions (memory writes or messages). The push-only transactions can be within a single host domain or across multiple host domains.

Isochronous service is realized through proper use of PCI Express mechanisms such as Traffic Class (TC) transaction labeling, Virtual Channel (VC) data-transfer protocol, and TC to VC mapping. End to end isochronous service requires software to set up proper configuration along the path between the Requester to Completer. This section describes the rules for software configuration and the rules hardware components must follow to provide end to end isochronous services. More information and background material regarding isochronous applications and isochronous design guide can be found in Appendix A..

7.3.4.1. Rules for Software Configuration

System software MUST obey the following rules to configure PCI Express fabric for isochronous transactions:

- Within a PCI Express hierarchy domain or within multiple PCI Express hierarchy domains spawned from Root Ports that have a common Root Complex Register Block (RCRB), software must designate one or more TCs for isochronous transactions. In the rest of this section, a TC designated for isochronous transactions is referred to as an Isochronous TC.
- The setting of the Attribute fields of all isochronous requests targeting the same Completer must be fixed and identical.
- On any PCI Express Link, software must assign all Isochronous TCs to the VC with the highest VC ID. In the rest of this section, this VC is referred to as the Isochronous VC.
- On any PCI Express port, software must configure the Isochronous VC so that it is served with the highest priority in VC arbitration. This is accomplished by either enabling strict priority VC arbitration (where the Isochronous VC having the highest VC ID would be served with the highest priority) or by configuring other VC arbitration mechanism to achieve equivalent effect.
- For Switch ports and RCRB, the Isochronous VC must support and be configured with a time-based Port Arbitration.
- Software must not assign other TCs to the Isochronous VC.
- Software must not assign Isochronous TC to any other VC.
- Software must not assign the number of isochronous transactions to a PCI Express port or RCRB that exceeds the Maximum Time Slots capability reported by the PCI Express port or RCRB. Software must not assign all PCI Express Link capacity to isochronous traffic in order to ensure forward progress of other transactions.
- Software must limit the Max_Payload_Size for each PCI Express hierarchy domain to meet the isochronous latency requirements.

7.3.4.2. 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.
- All read and write requests must never cross naturally aligned address boundaries.
- When system software indicates to the device driver of the Requester that snoop transaction is not allowed by the Completer, the Requester must set the "Snoop Not Required" Attribute bit.
- MSI must not be mapped to any TC used for isochronous traffic.

Note: An isochronous Requester that uses MSI mechanism must select a different TC (other than the one used for isochronous traffic) to transmit MSI packets. Before MSI can be generated, the Requester is required to perform synchronization (for example "flushing" using Memory Read of zero length).

7.3.4.3. Rules for Completers

A Completer providing isochronous services must obey the following rules:

- A Completer must not apply backpressure (due to the flow control) to isochronous requests injected uniformly to the PCI Express Link.
- A Completer must report its isochronous bandwidth capability in the Max Time Slots field in the VC Resource Capability Register intended for isochronous use. 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 RCRB and support time-based Port Arbitration mechanism for the Isochronous VC. Note that the time-based Port Arbitration only applies to request transactions.

7.3.4.4. Rules for Switch Components

A Switch component providing isochronous services must obey the following rules:

- A Switch port must not apply backpressure (due to flow control) to isochronous requests injected uniformly to the PCI Express Link.
  - A Switch component must observe the maximum isochronous transaction latency.
  - A Switch component must return isochronous read completions in strictly the same order as the corresponding isochronous read requests.
  - A Switch component must serve and forward isochronous write requests in strictly the same order.

- A Switch component must support time-based Port Arbitration mechanism for the Isochronous VC. Note that the time-based Port Arbitration only applies to request transactions but not to completion transactions.
- A Switch component must allow isochronous write requests (peer to peer) to pass isochronous read completions (Root Complex to Endpoint).

7.4. Device Synchronization STOP Mechanism

Renumbering bus numbers by system software during system operation may cause requestor ID (based upon bus numbers) for a given device to change; as a result, any requests or completions for that device still in flight may 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. A device synchronization stop mechanism is provided to allow system software to ensure that no transactions are in flight with respect to a particular endpoint device before performing a bus renumbering operation that causes the bus number (and requestor ID) to change for a given device.

The device synchronization stop mechanism for endpoint devices is implemented via the Stop mechanism and the associated Stop and Transactions Pending bits (see Section 5.8).

System software signals a device to stop by setting the Stop bit in the Device Command register of the device. The Stop operation is assumed to have completed by software if a device signals that no more transactions are pending by clearing the Transactions Pending status bit in the Device Status register; a device is not permitted to issue any new requests after the Stop bit is set.

Prior to clearing the Transaction Pending bit, an endpoint must ensure that:

- Completions for outstanding non-posted requests for all used Traffic Classes have been received by the corresponding Requestors.
- All requests with completions initiated by this device have returned completions.
- All posted requests of all Traffic Classes have been "flushed" (i.e., have been received by intended targets) in all directions including between endpoint and host, and between peer-to-peer endpoints.

Implementation Note: Flush Mechanisms

In a simple case such as that of an endpoint device communicating only with host memory, "flush" can be implemented using a directed memory read. A memory read needs to be performed on all TCs that the device is using. If a device has pending peer-to-peer transactions (including pending completions), then it must use a non-posted transaction such as a directed memory read targeted to specific peer destination to perform the "flush." The specific mechanism used is implementation specific but must be performed by hardware without software assistance.

7.5. Locked Transactions

7.5.1. Introduction

Locked Transaction support is required to prevent deadlock in systems that use legacy software which causes the accesses to I/O devices. Note that some CPUs may generate locked accesses as a result of executing instructions that implicitly trigger lock. Some legacy software misuses these transactions and generates locked sequences even when exclusive access is not required. Because locked accesses to I/O devices introduce potential deadlocks apart from those mentioned above, as well as serious performance degradation, PCI Express Endpoints are prohibited from supporting locked accesses, and new software must not use instructions which will cause locked accesses to I/O devices. Legacy Endpoints support locked accesses only for compatibility with existing software.

Only the Root Complex is allowed to initiate Locked Requests on PCI Express. Locked Requests initiated by Endpoints and Bridges are not supported. This is consistent with limitations for locked transaction use outlined in the *PCI Local Bus Specification, Rev 2.3* (Appendix F- Exclusive Accesses).

This section specifies the rules associated with supporting locked accesses from the Host CPU to Legacy Endpoints, including the propagation of those transactions through Switches and PCI Express/PCI Bridges.

7.5.2. Initiation and Propagation of Locked Transactions - Rules

Locked sequences are generated by the Host CPU(s) as one or more reads followed by an equal 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.

- Lock is initiated on PCI Express using the "lock"–type Read Request/Completion (MRdLk/CplDLk) and terminated with the Unlock Message
    - MRdLk, CplDLk and Unlock semantics are allowed only for the default Traffic Class (TC0)
- The Unlock Message is broadcast from the Root Complex to all Endpoints and Bridges
    - Any device which is not involved in the locked sequence must ignore this Message

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 also use MRdLk Requests
    - The Completions for any MRdLk Request use the CplDLk Completion type

- All writes for the locked sequence use MWr Requests
- The Unlock Message is used to indicate the end of a locked sequence
    - A Switch must propagate Unlock Messages to all Ports other than the Ingress Port, regardless of the state of the Switch or PCI Express/PCI Bridge with respect to lock
    - A PCI Express/PCI Bridge may propagate Unlock by deasserting LOCK# on its PCI interface
- 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

7.5.3. 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 the Egress Port is enabled to use one or more non-default VCs (VCs other than VC0), and if a Request specifies a Traffic Class which maps to a non-default VC on the Egress Port, then the Request must not be blocked
    - 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, 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-default VCs 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 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

7.5.4. 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 PCI-to-PCI Bridge Architecture Specification 1.1).

- When a PCI Express/PCI Bridge propagates a Locked Read Request through the Bridge, it must block all Requests not associated with the locked access from being propagated through the Bridge in the same direction as the Locked Read Request
- When the Locked Completion for the first Locked Read Request is returned, the Bridge must block all Requests not associated with the locked access from flowing through the Bridge in either direction
- The Bridge must remain blocked as described above until the Bridge is unlocked from the side which initiated the locked access
    - The Bridge unlocks itself and propagates the unlock to the other side of the Bridge

7.5.5. Root Complex and Lock - Rules

A Root Complex is permitted to support locked transactions as a Requestor. If locked transactions are supported, a Root Complex must follow the sequence described in Section 7.5.2 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.

7.5.6. 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
    - Once locked, the Legacy Endpoint must remain locked until it receives the Unlock Message
- While locked, a Legacy Endpoint must not issue any Requests using Traffic Classes which map to the default Virtual Channel (VC0)

  Note that this requirement applies to all possible sources of Requests within the Endpoint, in the case where there is more than one possible source of Requests.
    - Requests may be issued using Traffic Classes which map to VCs other than the default Virtual Channel

7.5.7. 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).

7.6. PCI Express Reset -Rules

This section specifies the behavior of PCI Express Link reset. The reset can be generated by the platform or on the component, but any relationship between the PCI Express Link reset and component or platform reset is component or platform specific (respectively).

- There must be a hardware mechanism for setting or returning all Port state to the initial conditions specified in this document – this mechanism is called "Power Good Reset"
    - A "Power Good Reset" will occur following the application of power to the component. This is called a "cold" reset
    - In some cases, it may be possible for the "Power Good Reset" mechanism to be triggered by hardware without the removal and re-application of power to the component. This is called a "warm" reset
    - Note that there is also an in-band mechanism for propagating reset across a Link. This is called a "hot" reset and is described in Section 4.2.4.5.
    - Note also that entering the DL_Inactive state is in some ways identical to a "hot" reset – see Section 2.13.
- On exit from any type of reset (cold, warm, or hot), all Port registers and state machines must be set to their initialization values as specified in this document

- On exit from a "Power Good Reset", the Physical Layer will attempt to bring up the Link (see Section 4.2.5). Once both components on a Link have entered the initial Link Training state, they will proceed through Link initialization for the Physical Layer and then through Flow Control initialization for VC0, making the Data Link and Transaction Layers ready to use the Link
    - Following Flow Control initialization for VC0, it is possible for TLPs and DLLPs to be transferred across the Link

Following a reset, some devices may require additional time before they are able to respond to Requests they receive. Particularly for Configuration Requests it is necessary that components and devices behave in a deterministic way, which the following rules address. The first set of rules address requirements for components and devices:

- A component must enter the initial active Link Training state within 80 ms of the end of "Power Good Reset" (Link Training is described in Section 4.2.5)
    - Note: In some systems, it is possible that the two components on a Link may exit "Power Good Reset" at different times. Each component must observe the requirement to enter the initial active Link Training state within 80 ms of the end of "Power Good 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 address requirements placed on the system:

- To allow components to perform internal initialization, system software must wait for at least 100 ms from the end of a reset (cold/warm/hot) before it is permitted to issue Configuration Requests to PCI Express devices
    - A system must guarantee that all components intended to be software visible at boot time are ready to receive Configuration Requests within 100 ms of the end of "Power Good Reset" – how this is done is beyond the scope of this specification
- The Root Complex and/or system software must allow 1.0s (+50% / -0%) after a reset (hot/warm/cold), before it may determine that a device which fails to return a Successful Completion status for a valid Configuration Request is a broken device
    - i.e.: if the Root Complex repeats Configuration Requests terminated with Configuration Request Retry Status, then it must continue repeating the Request(s) until 1s after T_{0RC}, at which point it is permitted to terminate the Request as a UR
    - 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 segment behind a PCI Express/PCI(-X) Bridge, the timing parameter T_{rhfa} must be respected

When a Link is in normal operation, the following rules apply:

- If, for whatever reason, a normally operating Link goes down, the Transaction and Data Link Layers will enter the DL_Inactive state (see Sections 2.13 and 3.2.1)
- For any virtual or actual PCI Bridge, any of the following must cause a reset of the secondary side of the Bridge using the Physical Layer mechanism for communicating Link Reset (see Section 4.2.4.5):
    - Setting the Secondary Bus Reset bit of the Bridge Control register
    - Entering DL_Inactive on the primary side of the Bridge
    - Link reset using the Physical Layer mechanism for communicating Link Reset

Certain aspects of "Power Good 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 "Power Good Reset". For example, it might be required that all PCI Express components within a chassis observe the assertion and deassertion of "Power Good 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 "Power Good Reset."

In all cases where power is supplied, the following parameters must be defined:

- T_{pvpgl} – "Power Good" must remain inactive at least this long after power becomes valid
- T_{pwrgd} – When deasserted, "Power Good" must remain deasserted at least this long
- T_{fail} – When power becomes invalid, "Power Good" must be deasserted within this time

Additional parameters may be specified.

In all cases where a reference clock is supplied, the following parameter must be defined:

- $T_{pwrgd\text{-}clk}$ – "Power Good" must remain inactive at least this long after any supplied reference clock stable

Additional parameters may be specified.

365

7.7. PCI Express Native Hot Plug Support

The PCI Express architecture is designed to natively support both hot plug and hot remove of devices. This section defines the standard usage model defined for all PCI Express form factors supporting Hot plug and hot removal of devices. This usage model provides the foundation for how indicators and push-buttons should behave if implemented in a system. The definitions of indicators and push-buttons apply to all PCI Express Hot-Plug models.

7.7.1. PCI Express Hot Plug Usage Model

7.7.1.1. Why Specify a Usage Model?

A standard usage model is beneficial to customers who buy systems with hot-plug slots because many customers utilize hardware and software from different vendors. A standard usage model allows customers to use the hot-plug slots on all of their systems without having to retrain operators. The PCI Express Hot-Plug standard usage model is derived from the standard usage model defined in the *PCI Standard Hot-Plug Controller and Subsystem Specification, Rev 1.0* and is identical from the user perspective. Note that only slight changes were made in register definitions and conformance to the standard usage model is required by all PCI Express form factors that implement hot-plug and use indicators and buttons.

Implementation Note: All PCI Express Form Factors that Support Hot-Plug/Remove Should Not Deviate from the Standard Usage Model

Deviating from the Standard Usage Model causes the solution to be non-PCI Express complaint and will create issues that would not exist otherwise, such as:

- User confusion
- More extensive hardware testing
- Functional incompatibilities with system software
- Encountering untested paths in system software

7.7.1.2. Elements of the Standard Usage Model

Table 7-5: Elements of the Standard Usage Model

Element	Purpose
Indicators	Shows the power and attention state of the slot
Manually-operated Retention Latch (MRL)	Holds add-in cards in place
MRL Sensor	Allows the port and system software to detect the MRL being opened
Electromechanical Interlock	Prevents removal of add-in cards while slot is powered
Attention Button	Allows user to request hot-plug operations
Software User Interface	Allows user to request hot-plug operations
Slot Numbering	Provides visual identification of slots

7.7.1.2.1. Indicators

The Standard Usage Model defines two indicators; the Power Indicator and the Attention indicator. The Platform can provide the two indicators per slot or module bay and the indicators can be can be implemented on the chassis or the module, see form factor hot plug requirements for implementation details. 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 status registers associated with the indicator.

The Hot-Plug capable port controls blink frequency, duty cycle, and phase. Blinking indicators operate at a frequency of 1 to 2 Hz and 50% (+/- 5%) duty cycle. Blinking indicators are not required to be synchronous and in-phase between ports.

Indicators must be placed in close proximity to their associated hot-plug slot if indicators are implemented on the chassis so that the association between the indicators and the hot-plug slot is clear.

Both indicators are completely under the control of system software. The Switch device or Root Port never changes the state of an indicator in response to an event such as a power fault or unexpected MRL opening unless commanded to do so by software. An exception is granted to Platforms capable of detecting stuck-on power faults. In the specific case of a stuck-on power fault, the Platform is permitted to override the Switch device or Root Port and force the Power Indicator to be on (as an indication that the add-in card should not be removed). In all cases, the ports internal state for the Power Indicator must match the software selected state. The handling by system software of stuck-on faults is optional and not described elsewhere. Therefore, the Platform vendor must ensure that this optional feature of the Standard Usage Model is addressed via other software, Platform documentation, or by other means.

7.7.1.2.1.1 Attention Indicator

The Attention Indicator is yellow or amber in color and is used to indicate that an operational problem exists or that the hot-plug slot is being identified so that a human operator can locate it easily.

Table 7-6: Attention Indicator States

Indicator Appearance	Meaning
Off	Normal - Normal operation
On	Attention - Operational problem at this slot
Blinking	Locate - Slot is being identified at the user's request

Attention Indicator Off

When the Attention Indicator is off, it means that neither the add-in card (if one is present) nor the hot-plug slot requires attention.

Attention Indicator On

When the Attention Indicator is on, it means an operational problem exists at the card or slot.

An operational problem is a condition that prevents continued operation of an add-in card. The operating system or other system software determines whether a specific condition prevents continued operation of an add-in card and whether lighting the Attention Indicator is appropriate. Examples of operational problems include problems related to external cabling, add-in cards, software drivers, and power faults. In general, when the Attention Indicator is on, it means that an operation was attempted and failed or that an unexpected event occurred.

The Attention Indicator is not used to report problems detected while validating the request for a hot-plug operation. Validation is a term applied to any check that system software performs to assure that the requested operation is viable, permitted, and will not cause problems. Examples of validation failures include denial of permission to perform a hot-plug operation, insufficient power budget, and other conditions that may be detected before an operation begins.

Attention Indicator Blinking

When the Attention Indicator is blinking, it means that system software is identifying this slot for a human operator to find. This behavior is controlled by a user (for example, from a software user interface or management tool).

7.7.1.2.1.2 Power Indicator

The Power Indicator is green in color and is used to indicate the power state of the slot.

Table 7-7: Power Indicator States

Indicator Appearance	Meaning
Off	Power Off - Insertion or removal of add-in cards is permitted. All supply voltages (except Vaux) have been removed from the slot if required for add-in card removal. Note that Vaux is removed when the MRL is open.
On	Power On - The slot is powered on. Insertion or removal of add-in cards is not permitted.
Blinking	Power Transition - The slot is in the process of powering up or down. Insertion or removal of add-in cards is not permitted.

Power Indicator Off

When the Power Indicator is off, it means that insertion or removal of an add-in card is permitted. Main power to the slot is off if required by the form factor, example of main power removal is the PCI Express card form factor. If the Platform provides Vaux to hot-plug slots and the MRL is closed, any signals switched by the MRL are connected to the slot even when the Power Indicator is off. Signals switched by the MRL are disconnected when the MRL is opened. System software must cause a slot's Power Indicator to be turned off when the slot is not powered and/or it is permissible to insert or remove add-in cards. See the appropriate electromechanical specifications for form factor details.

Power Indicator On

When the Power Indicator is on, it means that main power to the slot is on and that insertion or removal of an add-in card is not permitted.

Power Indicator Blinking

When the Power Indicator is blinking, it means that the slot is powering up or powering down and that insertion or removal of an add-in card is not permitted. A blinking Power Indicator also provides visual feedback to the human operator when the Attention Button is pressed.

7.7.1.2.2. Manually-operated Retention Latch (MRL)

An MRL is a manually-operated retention mechanism that holds an add-in card in the slot and prevents the user from removing the card. The MRL rigidly holds the card in the slot so that cables may be attached without the risk of creating intermittent contact. MRLs that hold down two or more add-in cards simultaneously are permitted in Platforms that do not provide MRL Sensors.

7.7.1.2.3. MRL Sensor

The MRL Sensor is a Switch, optical device, or other type of sensor that reports the position of a slot's MRL to the port. The MRL Sensor reports closed when the MRL is fully closed and open at all other times (that is, fully open and intermediate positions).

If Vaux is wired to hot-plug slots, the signals switched by the MRL must be automatically removed from the slot when the MRL Sensor indicates that the MRL is open and must be restored to the slot when the MRL Sensor indicates that MRL has closed again.

The MRL Sensor allows the port to monitor the position of the MRL and therefore allows the port to detect unexpected openings of the MRL. When an unexpected opening of the MRL associated with a slot is detected, the port changes the state of that slot to disabled and notifies system software. The port does not autonomously change the state of either the Power Indicator or Attention Indicator.

7.7.1.2.4. Electromechanical Interlock

An electromechanical interlock is a mechanism for physically locking the add-in card or MRL in place until the system software and port release it. Implementation of the interlock is optional. There is no mechanism in the programming interface for explicit control of electromechanical interlocks. The Standard Usage Model assumes that if electromechanical interlocks are implemented, they are controlled by the same port output signal that enables main power to the slot. Systems may optionally expand control of interlocks to provide physical security of the add-in cards.

7.7.1.2.5. Attention Button

An Attention Button is a momentary-contact push-button, located adjacent to each hot-plug slot or on a module that is pressed by the user to initiate a hot-insertion or a hot-removal at that slot.

The Power Indicator provides visual feedback to the human operator (if the system software accepts the request initiated by the Attention Button) by blinking. Once the Power Indicator begins blinking, a 5-second abort interval exists during which a second depression of the Attention Button cancels the operation.

If an operation initiated by an Attention Button fails for any reason, it is recommended that system software present a message explaining the failure via a software user interface or add the message to a system log.

7.7.1.2.6. Software User Interface

System software provides a user interface that allows hot-insertions and hot-removals to be initiated and that allows occupied slots to be monitored. A detailed discussion of hot-plug user interfaces is operating system specific and is therefore beyond the scope of this document.

On systems with multiple hot-plug slots, the system software must allow the user to initiate operations at each slot independent of the states of all other slots. Therefore, the user is

permitted to initiate a hot-plug operation on one slot using either the software user interface or the Attention Button while a hot-plug operation on another slot is in process, regardless of which interface was used to start the first operation.

7.7.1.2.7. Slot Numbering

A Physical Slot Identifier (as defined in PCI HP 1.1, Section 1.5) consists of an optional chassis number and the physical slot number of the hot-plug slot. System software determines the physical slot number from registers in the port. The chassis number is 0 for the main chassis. The chassis number for other chassis must be a non-zero value obtained from a PCI-to-PCI bridge's Chassis Number register (see PCI Bridge 1.1, Section 13.4).

The Standard Usage Model also requires that each physical slot number is globally unique within a chassis.

7.7.2. Event Behavior

Depending on the power state of the Switch device or Root Port, it may be programmed to generate a system interrupt or PME (see Table 7-8).

Table 7-8: Event Behavior

Event	Register Bit Set When Detected	Cleared by	Port Optionally Generates the Following When Event is Detected:
Presence Detect Change	Presence Detect Event Status	Writing a 1 to the detected bit	System Interrupt, PME
Attention Button Pressed	Attention Button Pressed Event	Writing a 1 to the detected bit	System Interrupt, PME
MRL Sensor Changed	MRL Sensor Change Detected Event	Writing a 1 to the detected bit	System Interrupt, PME
Power Fault	Power Fault Detected Event	Writing a 1 to the detected bit.	System Interrupt, PME

7.7.3. Registers Grouped by Device Association

The registers listed below are grouped by device to convey all registers associated with implementing each device in ports. These registers are unique to each Switch device or Root Port implementing hot plug slots.

7.7.3.1. Attention Button Registers

Description	Register Attribute	Default Value
Attention Button Present – This bit indicates if an Attention Button is implemented on the chassis or card.	HwInit	N/A
Attention Button Pressed – This bit is set when the Attention Button is pressed. This register is set by the debounced output of an Attention Button. This bit is also set by the port receiving the Attention_Button_Pressed message from the end device.	RW1C	0
Attention Button Pressed Enable – This bit when set enables the generation of the hot plug interrupt or a wake signal on an Attention Button Pressed event.	RW	0

7.7.3.2. Attention Indicator Registers

Description	Register Attribute	Default Value
Attention Indicator Present – This bit indicates if an Attention Indicator is implemented on the chassis or card.	HwInit	N/A
Attention Indicator Control – When read this register returns the current state of the Attention Indicator; when written the Attention Indicator is set to this state. If an Attention Indicator is implemented on the card, when written, the port will send the appropriate Attention Indicator message (determined by the decoding) to the device on the card. Defined encodings are: 00b Reserved 01b On 10b Blink 11b Off	RW	N/A

7.7.3.3. Power Indicator Registers

Description	Register Attribute	Default Value
Power Indicator Present – This bit indicates if a Power Indicator is implemented on the chassis or card.	HwInit	N/A
Power Indicator Control – When read this register returns the current state of the Power Indicator; when written the Power Indicator is set to this state. If a Power Indicator is implemented on the card, when written, the port will send the appropriate Power Indicator message (determined by the decoding) to the device on the card. Defined encodings are: 00b Reserved 01b On 10b Blink 11b Off	RW	N/A

7.7.3.4. Power Controller Registers

Description	Register Attribute	Default Value
Power Controller Present – This bit indicates if a Power Controller is implemented for this slot.	HwInit	N/A
Power Controller Control – When read, this register returns the current state of the Power applied to the slot; when written, the Power Controller turns on or off power to slot. Defined encodings are: 0b Power On 1b Power Off	RW	N/A
Power Fault Detected – This bit is set when the Power Controller detects a power fault at this slot.	RW1C	0
Power Fault Detected Enable – This bit when set enables the generation of the hot plug interrupt or a wake signal on a power fault event.	RW	0

7.7.3.5. Presence Detect Registers

Description	Register Attribute	Default Value
Presence Detect State – This bit indicates the presence of a card. in the slot. The bit will reflect the status of the Presence Detect pin as defined in the *PCI Express Card Electromechanical Specification*. Defined encodings are: 0b – Slot Empty 1b – Card Present in slot This register is required to be implemented on all Switch devices and Root Ports. The presence detect pin for Switch devices or Root Ports not connected to slots should be hardwired to 1.	RO	N/A
Presence Detect Changed Event – This bit is set when the value of Presence Detect State changes.	RW1C	0

7.7.3.6. MRL Sensor Registers

Description	Register Attribute	Default Value
MRL Sensor Present – This bit indicates if an MRL Sensor is implemented on the chassis.	HwInit	N/A
MRL Sensor Changed – This bit is set when the value of the MRL Sensor State changed.	RW1C	0
Presence Detect Changed Enable – This bit when set enables the generation of the hot plug interrupt or a wake signal on a presence detect changed event.	RW	0
MRL Sensor State – This register reports the status of the MRL sensor if it is implemented. Defined encodings are: 0b – MRL Closed 1b – MRL Open	RO	N/A

7.7.3.7. Port Capabilities and Slot Information Registers

Description	Register Attribute	Default Value
Slot Implemented – This bit when set indicates that the Link associated with this downstream port is connected to a slot, as oppose to being connected to an integrated device or being disabled.	HwInit	N/A
Physical Slot Number – This hardware initialized field indicates the physical slot number attached to the port. This field must be hardware initialized to a value that assigns a slot number that is globally unique within the chassis. These registers should be initialized to 0 for ports connected to integrated devices on the motherboard or integrated within the same silicon as the Switch device or Root Port.	HwInit	N/A
Hot-Plug capable – This bit when set indicates this slot is capable of supporting Hot-Plug.	HwInit	N/A
Hot-Plug Surprise – This bit when set indicates that the device might be removed from the system without any prior notification.	HwInit	N/A

7.7.3.8. Hot Plug Interrupt Control Registers

Description	Register Attribute	Default Value
Hot Plug Interrupt enable – This bit when set enables generation of the hot plug interrupt on enabled hot plug events.	RW	0
Command Completed Interrupt Enable – This bit when set enables the generation of hot plug interrupt when a command is completed by the hot plug control logic.	RW	0

7.7.4. Messages

The messages defined here allow for cards to implement indicators and buttons on the card without having to connect signals directly to the port. Detailed explanation of each message is located in Chapter 2.

7.7.4.1. Messages for Attention Indicator

This series of messages allows the Attention Indicator be implemented on the card as opposed to the chassis. These messages are sent by the downstream port to the device and instruct the device to set its Attention Indicator to the indicated state. The following messages are used:

ATTENTION_INDICATOR_ON

ATTENTION_INDICATOR_BLINK

ATTENTION_INDICATOR_OFF

All Endpoint devices are required to handle the Attention Indicator messages even if the device does not implement the indicators.

7.7.4.2. Messages for Power Indicator

This series of messages allows the Power Indicator be implemented on the card as opposed to the chassis. These messages are sent by the downstream port to the device and instruct the device to set its Power Indicator to the indicated state. The following messages are used:

POWER_INDICATOR_ON

POWER_INDICATOR_BLINK

POWER_INDICATOR_OFF

All Endpoint devices are required to handle the Power Indicator messages even if the device does not implement the indicators.

7.7.4.3. Messages for Attention Button

ATTENTION_BUTTON_PRESSED - This message allows the attention button to be implemented on the card and informs the port that the attention button has been pressed. Upon receipt of this message the port terminates the message and sets the Attention Button Pressed bit in the Hot-plug Event Register.

All down stream ports of switches and root ports are required to handle the Attention_Button_Pressed message.

7.7.5. PCI Express Hot Plug Interrupt/Wake Signal Logic

A port with hot plug compatibilities supports generation of hot plug interrupts on the following hot plug events:

- Attention Button Pressed
- Power Fault Detected
- MRL Sensor Changed
- Presence Detect Changed

When the system is in a sleep state or if the hot plug capable port is in a device state D1, D2, or D3hot, the enabled hot plug controller events generate a wake message (using PME mechanism) instead of a hot plug interrupt.

A hot plug capable port also supports generation of hot plug interrupt when the hot plug control logic completes an issued command. However, if the system is in a sleep state or if the hot plug capable port is in a device state D1, D2, or D3hot, a wake event will not be generated.

Figure 7-8 shows the logical connection between the hot plug event logic and the system interrupt/wake generation logic.

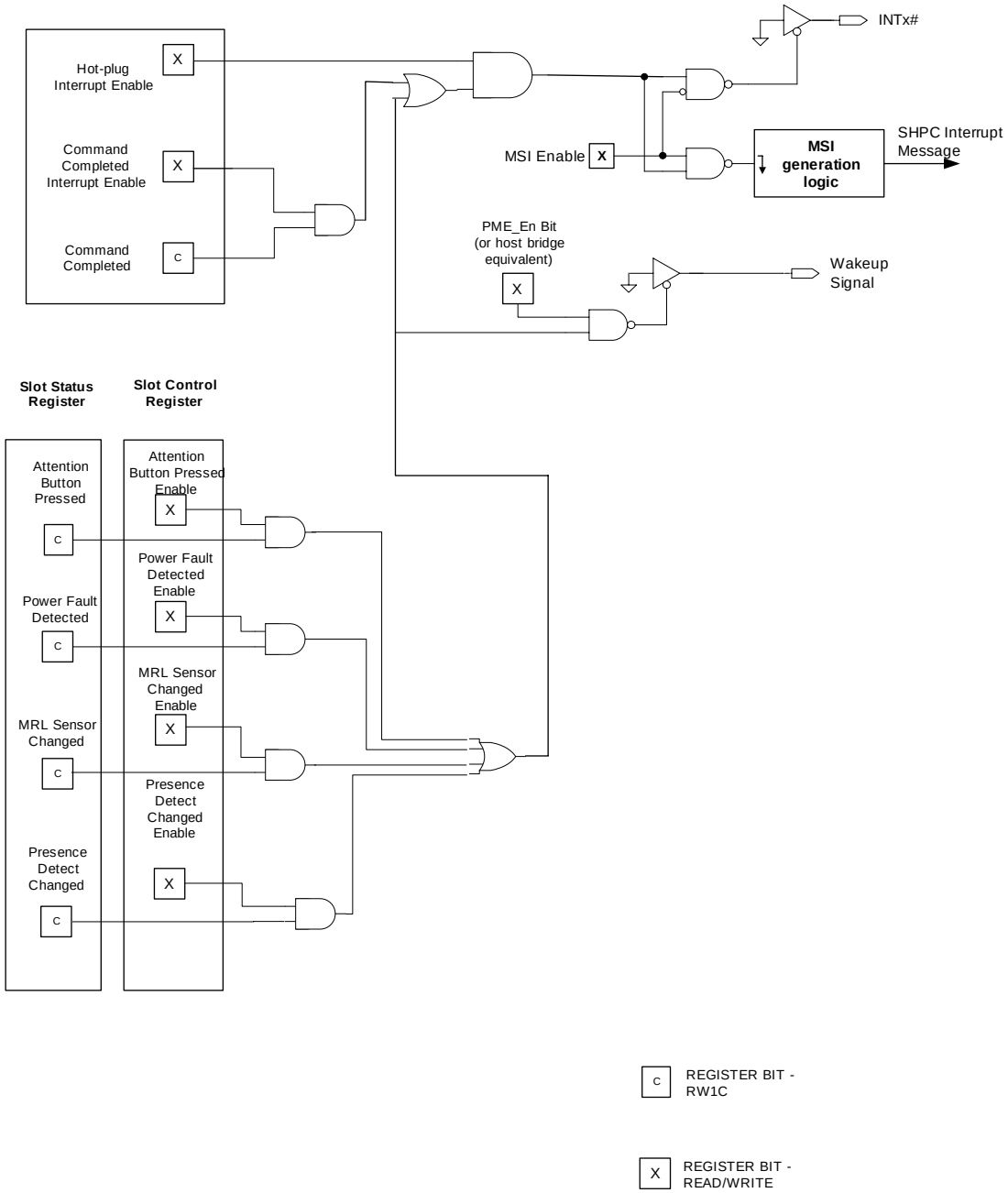

Figure 7-8: Hot Plug Logic

7.7.6. The Operating System Hot Plug Method

Some systems that include hot plug capable root ports and switches that are released before ACPI-compliant operating systems with native hot plug support are available, can use ACPI firmware for propagating hot plug events. Firmware control of the hot plug registers must be disabled if an operating system with native support is used. Platforms that provide ACPI firmware to propagate hot plug events must also provide a control method to transfer control to the operating system. This method is called Operating System Hot Plug (OSHP) and is provided for each port that is hot plug capable and being controlled by ACPI firmware.

Operating systems with native hot plug support must execute the OSHP method, if present, for each hot plug capable port before accessing the hot plug registers and when returning from a hibernated state. If a port's OSHP method is executed multiple times, and the switch to operating system control has already been achieved, the method must return successfully without doing anything. After the OSHP method is executed, the firmware must not access the ports hot plug registers. If any signals such as the System Interrupt or PME# have been redirected for servicing by the firmware, they must be restored appropriately for operating system control.

The following is an example of a namespace entry for an SHPC that is managed by firmware.

```
Device(PPB1){

      ...

      Method(OSHP, 0) {
            // Disable firmware access to SHPC and  restore
            // the normal System Interrupt and Wakeup Signal
            // connection.
      }
      ...
}
```

Implementation Note: Controlling Hot Plug by Using ACPI

When using ACPI to control the hot plug events, the following should be considered:

Firmware should redirect the System Interrupt to a GPE so that APCI can service the interrupts instead of the operating system. An appropriate *_Exx* GPE handler should be provided. When an operating system with native hot plug support executes the OSHP method, the firmware restores the normal System Interrupt so the interrupts can be serviced by the operating system.

7.8. Power Budgeting Capability

With the addition of a hot plug capability for add-in cards, 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 system to achieve these goals. There are many ways in which the system can implement the actual power management capabilities, and as such, they are beyond the scope of this specification.

Devices that will be present on hot pluggable add-in cards are required to implement the power budgeting capabilities. Devices that are implemented for use on add-in cards or on the motherboard have the option of supporting the power budgeting capability. Devices that are designed for both add-in cards and modules must implement power budgeting. The devices and/or add-in cards are required by PCI Express 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 add-in card.

7.8.1. 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 (motherboard devices).
- Total number of slots and the types of slots.

System firmware is responsible for allocating power for all devices on the motherboard that do not have power budgeting capabilities. The firmware may or may not include standard PCI Express devices that are connected to the standard power rails. When the firmware allocates the power for a device then it must set the SYSTEM_ALLOC bit to "1" to indicate that it has been properly allocated. The power budget manager is responsible for allocating all PCI Express devices including motherboard devices that have the power budgeting capability and have not been marked allocated. The power budget manager is responsible for determining if hot plugged devices can be budgeted 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.

7.9. Slot Power Limit Control

PCI Express provides a mechanism for software controlled limiting of the maximum power per slot that PCI Express card/module (associated with that slot) can consume. The key elements of this mechanism are the:

- Slot Power Limit Value and Scale fields of the Slot Capability register implemented in the Downstream Ports of a Root Complex and a Switch
- Slot Power Limit Value and Scale fields of the Device Capability register implemented in the Upstream Ports of a Endpoint, Switch and PCI Express-PCI Bridge
- Set_Slot_Power_Limit message that conveys the content of the Slot Power Limit Value and Scale fields of the Slot Capability register of the Downstream Port (of a Root Complex or a Switch) to the corresponding Slot Power Limit Value and Scale fields of the Device Capability register in the Upstream Port of the component connected to the same Link

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 IO expansion using add-in cards/modules
- power delivery capabilities
- thermal capabilities

This software is responsible for correctly programming the Slot Power Limit Value and Scale fields of the Slot Capability registers of the Downstream Ports connected to IO expansion slots. After the value has been written into the register within the Downstream Port, it is conveyed to the other component connected to that port using the Set_Slot_Power_Limit message (see Section 2.8.1.5. The receipient of the message must use the value in the message data payload to limit usage of the power for the entire card/module, unless the card/module will never exceed the lowest value specified in the correspondnig electromechanical specification. It is assumed that device driver software associated with card/module will be able (by reading the values of the Slot Power Limit Value and Scale fields of the Device Capability register) to configure hardware of the card/module to guarantee that the card/module will not exceed imposed limit. In the case where the platform imposes a limit that is below minimum needed for adequate operation, the device driver will be able to communicate this discreprency to higher level configuration software.

The following rules cover the Slot Power Limit control mechanism:

For Cards/Modules:

- Until and unless a Set_Slot_Power_Limit message is received indicating a Slot Power Limit value greater than the lowest value specified in the electromechanical specification for the card/module's form factor, the card/module must not consume more than the lowest value specified.

381

- A card/module must never consume more power than what was specified in the most recently received Set_Slot_Power_Limit message.
- Endpoint, Switch and PCI Express-PCI Bridge components that are targeted for integration on a card/module where total consumed power is below lowest limit defined for the targeted form factor are permitted to ignore Set_Slot_Power_Limit messages, and to return a value of 0 in the Slot Power Limit Value and Scale fields of the Device Capability register
- Such components still must be able to receive the Set_Slot_Power_Limit message correctly but simply discard the message

For Root Complex and Switches which source slots:

- A Downstream Port must not transmit a Set_Slot_Power_Limit message which indicates a limit that is lower than the lowest value specified in the electromechanical specification for the slot's form factor.

Implementation Note: Slot Power Limit Control Registers

Typically Slot Power Limit registers within Downstream Ports of Root Complex or a Switch Device will be programmed by platform-specific software. Some implementations may use a hardware method for initializing the values of these registers and therefore not require software support.

Endpoint, Switch and PCI Express-PCI Bridge components that are targeted for integration on the card/module where total consumed power is below lowest limit defined for that form factor are allowed to ignore Set_Slot_Power_Limit messages. Note that PCI Express components that take this implementation approach may not be compatible with potential future defined form factors. Such form factors may impose a lower power limit which is below the minimum required by a new card/module based on the existing component.

382

A. Isochronous Applications and Support

A.1. Introduction

Data traffic in PCI Express environment can be generally classified in two categories: bulk-data traffic and real-time traffic.

While a PCI Express Endpoint device with bulk data transfer generally requires high throughput and low latency to achieve good performance, it can tolerate occasional data transfers that complete with arbitrarily long delays. The normal semantics for general-purpose I/O transactions, as defined for PCI Express default Traffic Class (TC0), are supported by the default PCI Express Virtual Channel (VC0). VC0 supports bulk data transfer by providing "best-effort" class of service. This means, since there is no traffic regulation for the VC0, during any given time period, any device may issue more transactions than PCI Express Links can support and may saturate the physical PCI Express Links. Therefore, there is no guaranteed bandwidth or deterministic latency provided to the device by the VC0. This is why the default general purpose I/O Traffic Class is referred to as the "best-effort" Traffic Class.

On the other hand, a PCI Express Endpoint device with real-time data transfer requirements, such as audio and video data streaming, would continuously/periodically generate PCI Express transactions. The amount of bandwidth that device can consume will depend on device's requirements and may be subject of limitation that can be imposed by the PCI Express platform software and hardware. Isochronous data transfer protocol in PCI Express is designed to provide not only guaranteed data bandwidth but also deterministic service latency. 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 level of service falling in between what are required for bulk data traffic and isochronous data traffic. These types of transactions can be supported in general by using Traffic Classes (TC1 to TC7) associated with differentiated services. However, details of service policies for these Traffic Classes are not addressed in this section.

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. In the Peer-to-Peer model, isochronous traffic is limited to unicast push-only transactions (memory writes or messages). The push-only transactions can be within a single host domain or across multiple

383

host domains. Figure A-1 shows an example of a simple system with both communication models. In the figure, devices A, B, called Requesters, are PCI Express Endpoint devices capable of issuing isochronous request transactions, while device C and Root Complex, called Completers, are capable of being the targets of isochronous request transactions. An Endpoint-to-Root-Complex communication is established between device A and the Root Complex, and a Peer-to-Peer communication is established between device B and device C. In the rest of this section, Requester and Completer will be used to make reference to PCI Express elements involved in transactions. The specific aspects of each communication model will be called out explicitly.

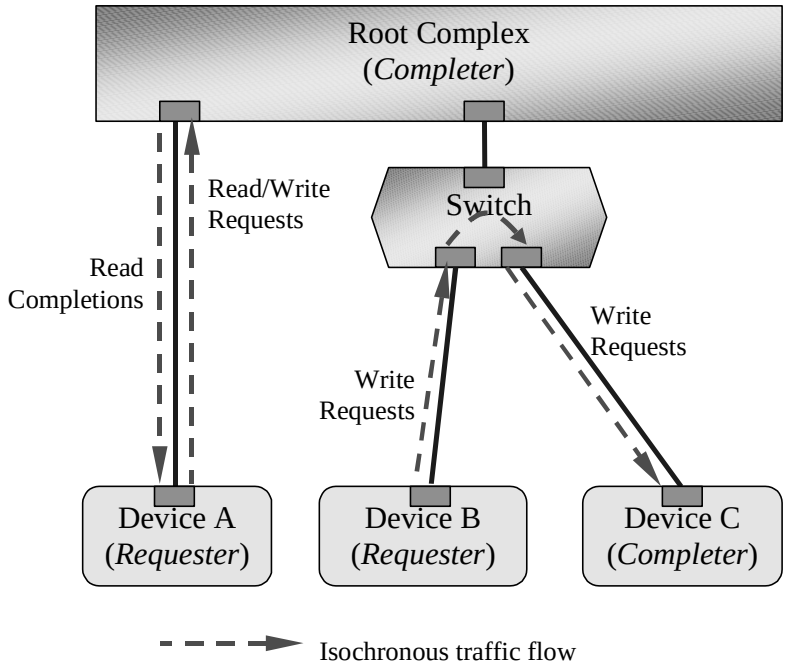

Figure A-1: An Example Showing Endpoint-to-Root-Complex and Peer-to-Peer Communication Models

- Guaranteed bandwidth and deterministic latency requires end to end isochronous service. If the Isochronous TC is ever mixed with other Traffic Classes in the same Virtual Channel on a PCI Express Link, then head of line blocking caused by traffic interaction and flow control may compromise the Quality of Service (QoS) for isochronous transactions. Although some level of QoS may be provided if this traffic mixing occurs only on a small portion of the data path, it may not be quantifiable. Therefore, for the rest of this Section, we assume that dedicated Virtual Channels are provided for the Isochronous TC on each PCI Express Link to provide end to end isochronous service and all PCI Express components along the path between the Requester and the Completer meet the requirements described in this Section. The dedicated Virtual Channel for the Isochronous TC can also be called Isochronous VC. Specifically, system software must obey the rules described in Section 2.6.4.

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 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 too many isochronous requests are issued in a short time window. This potentially causes excessive service latencies for both isochronous traffic and non-isochronous traffic.

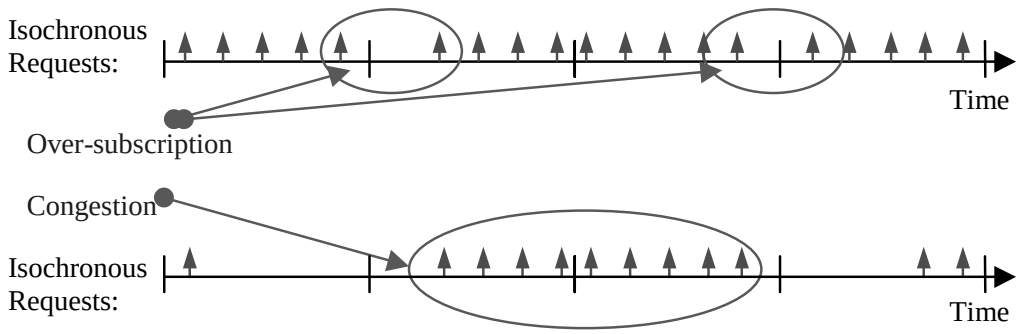

Figure A-2: Two Basic Bandwidth Resourcing Problems: Over-Subscription and Congestion

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 device) by the Completer (Root Complex for Endpoint-to-Root-Complex communication and another PCI Express Endpoint device for Peer-to-Peer communication) and by 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. In this way the isochronous service to other well-behaved (compliant) Requesters will not be affected by the non-compliant device.

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, write requests (and messages) are budgeted together. A Requester may divide the isochronous bandwidth between read requests and write requests as appropriate.

In the (push-only) Peer-to-Peer model, isochronous bandwidth only applies to request transactions.

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). Up to one isochronous request is allowed within one virtual timeslot. The virtual timeslot supported by a PCI Express component is reported through the Reference Clock field in the PCI Express Virtual Channel Capability Structure defined in Section 5.11. When Reference Clock = 00b, duration of a virtual timeslot t is 100 ns. Duration of isochronous time period T depends on the number of phases of the supported time-based WRR port arbitration table size. When the time-based WRR Port Arbitration Table size equals to 128, there are 128 virtual timeslots (t) in an isochronous time period, i.e. T = 12.8 ms.

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., 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 5.8.4). After configuration, Max Payload Size is fixed within a PCI Express hierarchy domain. 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

$$Y = \text{Max_Payload_Size}.$$

In order for Completers to meet isochronous contract, Requesters must ensure that any isochronous request contains a naturally aligned data block. A transaction with partial write is treated as a normally accounted transaction. A Completer must account for partial writes as part of bandwidth assignment (for worst case servicing time).

A.2.3. Isochronous Bandwidth Allocation

Given T, t and Y, the maximum virtual timeslots within a time period is

$$N_{\max} = \frac{T}{t},$$

and the maximum specifiable isochronous bandwidth is

$$BW_{\max} = \frac{Y}{t}.$$

The granularity with which isochronous bandwidth can be allocated is defined as:

$$BW_{granularity} = \frac{Y}{T}.$$

Given T and t at 12.8 ms 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

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

$$N_{link} = \frac{BW_{link}}{BW_{granularity}}.$$

For a Switch port serving as an Egress Port (or a 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 5.11. 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 that 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 are given value of PN. The Egress Port may admit one isochronous request transaction from the ingress port for further service only when the table entry reached by the Egress Port's isochronous time ticker (that increments by 1 every t time and wraps around when reaching T) is set to PN. Even if there are outstanding isochronous requests ready in the ingress port, they will not be served until next round of time-based WRR arbitration. In this manner, the time-based Port Arbitration Table serves for both isochronous bandwidth assignment and isochronous traffic regulation.

For a PCI Express Endpoint device serving as a Requester or a Completer, isochronous bandwidth allocation is accomplished through negotiation between system software and device driver, which is outside of the scope of this specification.

A.2.4. Isochronous Transaction Latency

Transaction latency is composed of the latency through the PCI Express fabric and the latency contributed by the rest of the system. For memory transactions, transaction latency is the accumulated delay across the PCI Express fabric plus the service delay of the Completer. Isochronous transaction latency is defined for each transaction and measured in units of virtual timeslot *t*.

- For a Requester in the Endpoint-to-Root-Complex model, 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).
- For a Requester in both Endpoint-to-Root-Complex and Peer-to-Peer models, 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.

As part of the isochronous contract, 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 using the minimum and maximum isochronous transaction latencies. As shown later, for most of common platforms, the minimum isochronous transaction latency is much smaller than the maximum isochronous transaction latency. As a conservative measure, we set the minimum isochronous transaction latency to zero and only provide guidelines on measuring the maximum isochronous transaction latency.

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.

Transaction latency for a PCI Express Link or a PCI Express fabric, L_{Fabric}, is defined as the delay from the time a transaction is posted at the transmission end to the time it is available at the receiving end. This applies to both read and write transactions. (Note that read transactions traverse PCI Express fabric twice, first time during the request phase and second time during the completion phase.) L_{Fabric} depends on the topology, latency due to each PCI Express Link and arbitration point in the path from 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.

As specified later, a restriction on the PCI Express topology is imposed for each targeted platform in order to provide a practically meaningful guideline for L_{Fabric}. The values of L_{Fabric} provided in the guideline should be reasonable and serve as practical upper limits under normal operating conditions.

The value of $L_{Completer}$ depends on the memory technology and specific memory configuration settings 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, two read requests and two write requests are issued by the Requester. 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.

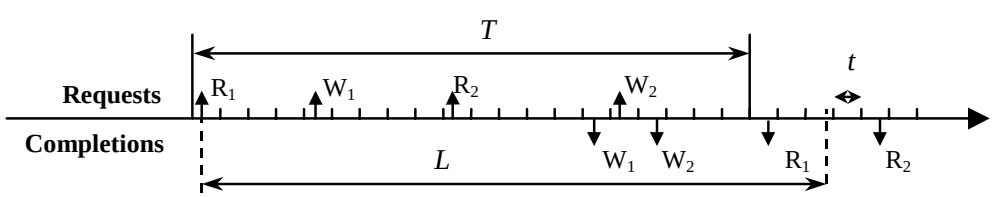

Figure A-3: A Simplified Example Illustrating PCI Express Isochronous Parameters

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 read requests must never exceed Max Payload Size.
- All read and write requests must never cross naturally aligned address boundaries.

A.4. Transaction Ordering

In general, isochronous transactions follow the same ordering rules as described in Section 2.5. The following ordering rules further illustrate and clarify the proper behavior of isochronous transactions:

- There is no ordering between isochronous transactions and other PCI Express transactions, since on each PCI Express Link isochronous transactions are mapped to a dedicated Virtual Channel and are not mixed with transactions of other Traffic Classes.
- Isochronous write requests are served on any PCI Express Link in strictly the same order as isochronous write requests are posted.
- Switches must allow isochronous write requests to pass isochronous read completions.

A.5. Isochronous Data Coherency

Cache coherency for isochronous transactions is not an I/O interconnect issue but rather an operating system software and Root Complex hardware issue. This specification provides the necessary mechanism to control Root Complex behavior in terms of enforcing hardware cache coherency on a 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 must have the "Snoop Not Required" Attribute bit set.

Root Complex must report the Root Complex's capability to the system software by setting the Snoop Transaction Permitted field in the VC Resource Capability Register (for the VC resource intended for isochronous traffic) in 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 device drive of Endpoint devices to set or unset the "Snoop Not Required" 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 (PCI Express Endpoint device or Root Complex) and PCI Express fabric components should implement proper sizing of buffers such that under normal operating conditions, no back-pressure 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 back-pressure 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 either due to excessive Data Link Layer Retry, flow control provides a natural mechanism to ensure functional correctness.

A.7. Topology Restrictions

Total service latency for a Requester depends on the position of that device within a particular PCI Express topology. In order to provide a realistic upper bound of such latency, it is necessary to establish topology restrictions for target platforms.

For desktop, volume workstation and mobile platforms, the worst case topology is 3-level deep as shown in Figure A-4. In other words, for Endpoint-to-Root-Complex communication, a PCI Express Endpoint device with isochronous service request needs to be able to work on a platform with two levels of Switches between it and the Root Complex. Peer-to-Peer communication should also work for the same 3-level deep PCI Express topology for two PCI Express Endpoint devices that support Peer-to-Peer communication.

For server, high-end workstation and embedded communication platforms, the worst case topology can go beyond 3-level deep. In these platforms, a PCI Express Endpoint device with isochronous service request may connect to a Root Complex or other peer Endpoint devices through cascaded Switches.

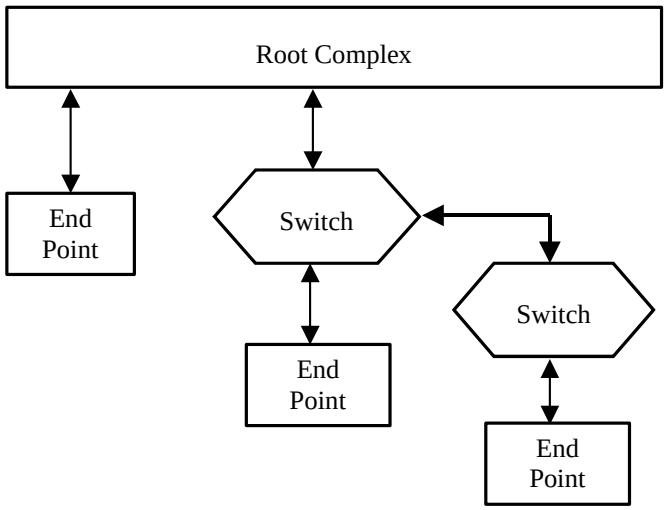

Figure A-4: An Example of PCI Express Topology Supporting Isochronous Applications

A.8. Transfer Reliability

Same as for non-isochronous traffic, reliable transfer is provided for isochronous traffic by PCI Express interconnect and Completer's memory subsystem. In other words, once an isochronous request is accepted in the PCI Express fabric, it will not be dropped by any PCI Express component. When the request requires completion, corresponding completion packet(s) will be returned to the requester. Requesters are responsible for shaping and conditioning isochronous traffic. With resource reservation and traffic regulation mechanism described above, guaranteed isochronous service is provided under normal operating conditions. When such conditions are not met, errors due to retry and flow control manifest in excessive latencies for isochronous transactions. In order to resolve the congestion caused by excessive retries and flow control (for example, one retry per isochronous period per Link may be budgeted in isochronous resource reservation managed by system software), a Requester may delay or drop non-committed isochronous requests. It may also drop late-received completions. For late-received data packets in the Completer's memory subsystem, it is up to the application and/or driver software to determine if data should be discarded.

A.9. Considerations for Bandwidth Allocation

A.9.1. 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, the speed, and the width of the Link.

Isochronous bandwidth allocation for a PCI Express Link is 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 timeslots allowed per Link (N_{link}) depends on the isochronous packet payload size and also the speed and width of the Link. Table A-2: shows N_{link} and Link utilization as functions of isochronous payload size and PCI Express Link width when the Link runs at 2.5 GHz and isochronous Link utilization limited to 50%. For low to medium PCI Express Links width (with number of Lanes between 1 and 8), the relatively slow Link bandwidth limits the isochronous resource (virtual timeslot) allocation. However, for wider PCI Express Links (with 12 or 16 Lanes), the relatively large virtual timeslot (at 100 n) limits the isochronous resource allocation.

Table A-2: Maximum Number of Virtual Timeslots Allowed for Different PCI Express Links at 2.5 GHz

# Lanes	1		2		4		8		12		16	
Y (Bytes)	N_{link}	% Util	N_{link}	% Util	N_{link}	% Util	N_{link}	% Util	N_{link}	% Util	N_{link}	% Util
128	11	50%	22	50%	44	50%	88	50%	128	48%	128	36%
256	5	43%	11	47%	23	49%	46	49%	70	50%	93	50%
512	3	50%	6	50%	12	50%	24	50%	36	50%	48	50%

As isochronous bandwidth allocation on a PCI Express Link is based on number of transactions N_{link} per isochronous period. There is no distinction between read requests and write requests in budgeting isochronous bandwidth on a PCI Express Link. In other words, even though a read request packet (without payload) can be much smaller than a write request packet (with payload), their Link utilization is accounted as the same according to the larger one (a write request). This is because for each read request in one direction of a PCI Express Link there will be one or more read completions with payload on the other direction of the PCI Express Link. Without differentiating between read and write request transactions, the allocated isochronous bandwidth for a PCI Express Link in the Endpoint-to-Root-Complex model is assumed to consume bandwidth in both directions. For the push-only Peer-to-Peer model, software may take advantage of the unidirectional isochronous traffic pattern in budgeting PCI Express Link resource.

A.9.2. Isochronous Bandwidth of Endpoint Devices

For Peer-to-Peer communication, when a PCI Express Endpoint device serves as the Completer of isochronous traffic, its device driver is responsible for reporting to the operating system-level PCI Express isochronous configuration software if the device is capable of being a Completer for isochronous transactions. In addition, the device driver must report if there is enough bandwidth to service the requests within the Completer's memory subsystem. The specifics of the reporting mechanism are outside of the scope of this specification.

A.9.3. 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.9.4. Isochronous Bandwidth of Root Complex

Isochronous bandwidth of Root Complex is reported to the software through RCRB Structure. Specifically, the Maximum Time Slots field of the VC Resource Capability Register in VC Capability Structure indicate 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.10. Considerations for PCI Express Components

A.10.1. A PCI Express Endpoint Device as a Requester

Before a PCI Express Endpoint device as a Requester can start issuing isochronous request transactions, the following configuration steps must be performed by software:

- Configuration of an Isochronous Virtual Channel that Isochronous Traffic Class is mapped to.
- Enabling of the Isochronous VC.

According to the rules stated in Chapter 2, an Endpoint Requester must issue isochronous transactions using Flow Control credits available for the corresponding Isochronous VC.

When isochronous transactions (requests) are injected uniformly, the receive Port, being a Switch Port or a Root Port, will issue Flow Control credit back promptly such that no back-pressure is applied to the Isochronous VC. Therefore, the Endpoint Requester can 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 Endpoint Requester gets back-pressured

on the Isochronous VC. This kind of Requester must size its buffer to account for the deviation of its injection pattern from uniformity.

A.10.2. A PCI Express Endpoint Device as a Completer

A PCI Express Endpoint device may serve as a Completer for isochronous Peer-to-Peer communication. Before a PCI Express Endpoint device starts serving isochronous transactions, its PCI Express Port must be configured by operating system-level configuration software to enable an Isochronous VC.

An Endpoint Completer must observe the maximum isochronous transaction latency ($L_{Completer}$). How an Endpoint Completer schedules memory cycles for PCI Express isochronous transactions and other memory transactions is outside of the scope of this specification as long as $L_{Completer}$ is met for PCI Express isochronous transactions.

An Endpoint Completer communicates with a Requester through PCI Express fabric components such as Switches. Since isochronous requests injected to an Endpoint Completer have already been regulated by Switches to conform to the isochronous contract, the Endpoint Completer does not have to regulate isochronous request traffic. However, an Endpoint Completer must size its internal buffer such that no back-pressure is applied to the Isochronous VC.

Since Switches do not check for the additional isochronous transactions rules stated in Section A.3, an Endpoint Completer 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.
- Return partial completions for read requests that cross naturally aligned address boundaries.
- Write partial data for write requests that cross naturally aligned address boundaries.

A.10.3. Switches

A Switch may have multiple ports capable of supporting isochronous transactions. Before a Switch starts serving isochronous transactions for a port, the following configuration steps must be performed by the software:

- Configuration of an Isochronous Virtual Channel that Isochronous Traffic Class is mapped to.
- Configuration of the port as an ingress port:
  - Configuration (or reconfiguration if the Egress Port Isochronous VC 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 the Isochronous VC's Port Arbitration Table with number of entries set according to the assigned isochronous bandwidth for all ingress ports with isochronous traffic targeting the Egress Port.
  - Select proper VC Arbitration such as strict-priority based VC Arbitration.
  - If required, configuration of the port's VC Arbitration Table with large weights assigned to the Isochronous VC.
- Enabling of the Isochronous VC for the port.

The Isochronous VC needs to 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. As the Isochronous VC is assigned with highest VC ID, for Switch port that supports priority-based VC arbitration, the Isochronous VC is served with the highest arbitration priority. For Switch port that supports WRR-based VC arbitration, software must program the weights for the Isochronous VC to be large enough so that the service is equivalent to a highest priority one.

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 an Isochronous VC 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 ingress port conforms to this specification (N_{link} transactions per isochronous period programmed in the target Switch Egress Port's Port Arbitration Table). With a such enforcement, normal isochronous transactions from compliant Requesters will not be impacted by ill behavior of any incompliant Requester.

Isochronous traffic regulation from any ingress port is implemented as part of the Port Arbitration of the target Egress Port. Specifically, a time-based WRR Port Arbitration is used to schedule isochronous read and/or write request transactions. The N_{max} virtual timeslots (t) within the isochronous time period (T) are represented by the time-based WRR Table in the PCI Express Virtual Channel Capability Structure detailed in Section 5.11. The table consists of N_{max} entries. A table entry represents one virtual timeslot. An ingress Port is assigned with N_{link} virtual timeslots when N_{link} entries in the target Egress Port's time-based WRR Port Table are set to the ingress port's Port Number.

The above isochronous traffic regulation mechanism only applies to request transactions but not to completion transactions. As read completion transactions only come from upstream port and go to downstream ports, no Port Arbitration is needed. When Endpoint-to-Root-Complex and Peer-to-Peer communications co-exist in a Switch, a downstream (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 meet latency requirement for isochronous requests.

A.10.4. 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 the operating system-level PCI Express configuration software to enable an Isochronous VC using the following configuration steps:

- Configuration of an Isochronous Virtual Channel that Isochronous Traffic Class is mapped to.
- Configuration of the Root Port as an Ingress Port:
    - Configuration (or reconfiguration if the Isochronous 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 ingress 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 Isochronous VC.
- Enabling of the Isochronous VC for the Root Port.

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 an Isochronous VC is enabled for a Root Port, the Root Complex must enforce proper traffic regulation to ensure that isochronous traffic from the Root Port confirms to

this specification (N_{link} transactions per isochronous period). With such enforcement, normal isochronous transactions from compliant Requesters will not be impacted by ill behavior of any incompliant Requesters. Isochronous traffic regulation is implemented using the time-based Port Arbitration Table in RCRB.

As Switches do not check for the additional isochronous transaction rules stated in Section A.3, 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.
- Return partial completions for read requests that cross naturally aligned address boundaries.
- Write partial data for write requests that cross naturally aligned address boundaries.

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	Bits HGF EDCBA	Current RD - abcdei fghj	Current RD + abcdei fghj
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	Bits HGF EDCBA	Current RD - abcdei fghj	Current RD + abcdei fghj
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
D24.1	38	001 11000	110011 1001	001100 1001

Data Byte Name	Data Byte Value	Bits HGF EDCBA	Current RD - abcdei fghj	Current RD + abcdei fghj
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
D25.2	59	010 11001	100110 0101	100110 0101
D26.2	5A	010 11010	010110 0101	010110 0101

401

Data Byte Name	Data Byte Value	Bits HGF EDCBA	Current RD - abcdei fghj	Current RD + abcdei fghj
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
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

Data Byte Name	Data Byte Value	Bits HGF EDCBA	Current RD - abcdei fghj	Current RD + abcdei fghj
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
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

Data Byte Name	Data Byte Value	Bits HGF EDCBA	Current RD - abcdei fghj	Current RD + abcdei fghj
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
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

Data Byte Name	Data Byte Value	Bits HGF EDCBA	Current RD - abcdei fghj	Current RD + abcdei fghj
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
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

Data Byte Name	Data Byte Value	Bits HGF EDCBA	Current RD - abcdei fghj	Current RD + abcdei fghj
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
D30.7	FE	111 11110	011110 0001	100001 1110
D31.7	FF	111 11111	101011 0001	010100 1110

406

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

408

C. Physical Layer Appendix

C.1. Data Scrambling

The following subroutines encode and decode an eight-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
    this advances the lfsr 8 bits every time it is called
    this fewer than 36 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  8   9   10  11  0   1   2   3   4   5   6   7
    9   10  11  12  9   10  11  12  12  13  14  15      8   9   8
    10  11  12  13  10  11  12  13  13  14  15          9   10  9
    12  13  14  15  13  14  15      14  15              10  11  11
    15              14  15                              12  13  14
                    15                                  15      15

*/
```

```
/* XOR required for creating the serial data is

bi  0   1   2   3   4   5   6   7

    15  14  13  12  11  10  9   8
        15  14  13  12  11  10  9
            15  14  13  12  11  10
                    15  14  13  12
                                15

  */

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] ^ bit[15];
        scrambit[2] ^= bit[13] ^ bit[14] ^ bit[15];
        scrambit[3] ^= bit[12] ^ bit[13] ^ bit[14];
        scrambit[4] ^= bit[11] ^ bit[12] ^ bit[13] ^ bit[15];
        scrambit[5] ^= bit[10] ^ bit[11] ^ bit[12] ^ bit[14];
        scrambit[6] ^= bit[9] ^ bit[10] ^ bit[11] ^ bit[13];
        scrambit[7] ^= bit[8] ^ bit[9] ^ bit[10] ^ bit[12] ^ bit[15];
    }
```

```
    outbyte = 0;

    for (i= 0; i<8; i++)            // convert data back to an integer
        outbyte += (scrambit[i] << i);

// Now advance the LFSR 8 serial clocks

    bit_out[0] = bit[8] ^ bit[9] ^ bit[10] ^ bit[12] ^ bit[15] ;
    bit_out[1] = bit[9] ^ bit[10] ^ bit[11] ^ bit[13];
    bit_out[2] = bit[10] ^ bit[11] ^ bit[12] ^ bit[14];
    bit_out[3] = bit[11] ^ bit[12] ^ bit[13] ^ bit[15];
    bit_out[4] = bit[8] ^ bit[9] ^ bit[10]^ bit[13] ^ bit[14] ^ bit[15] ;
    bit_out[5] = bit[9] ^ bit[10] ^ bit[11] ^ bit[14] ^ bit[15]  ;
    bit_out[6] = bit[10] ^ bit[11] ^ bit[12] ^ bit[15] ;
    bit_out[7] = bit[11] ^ bit[12] ^ bit[13];
    bit_out[8] = bit[0] ^ bit[12] ^ bit[13] ^ bit[14] ;
    bit_out[9] = bit[1] ^ bit[13] ^ bit[14] ^ bit[15] ;
    bit_out[10] = bit[2] ^ bit[14] ^ bit[15];
    bit_out[11] = bit[3] ^ bit[15];
    bit_out[12] = bit[4] ;
    bit_out[13] = bit[5] ^ bit[8] ^ bit[9] ^ bit[10] ^ bit[12] ^ bit[15] ;
    bit_out[14] = bit[6] ^ bit[9] ^ bit[10] ^ bit[11] ^ bit[13] ;
    bit_out[15] = bit[7] ^ bit[8] ^ bit[9] ^ bit[11] ^ bit[14] ^ bit[15] ;


    lfsr = 0;

    for (i=0; i <16; i++)       // the LFSR back to an integer
        lfsr += (bit_out[i] << i);

    return outbyte;
}

/*
    this routine implements the serial descrambling algorithm in parallel form
    this advances the lfsr 8 bits every time it is called
    this fewer than 36 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 l
                               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] ^ bit[15];
    descrambit[2] ^= bit[13] ^ bit[14] ^ bit[15];
    descrambit[3] ^= bit[12] ^ bit[13] ^ bit[14];
    descrambit[4] ^= bit[11] ^ bit[12] ^ bit[13] ^ bit[15];
    descrambit[5] ^= bit[10] ^ bit[11] ^ bit[12] ^ bit[14];
    descrambit[6] ^= bit[9] ^ bit[10] ^ bit[11] ^ bit[13];
    descrambit[7] ^= bit[8] ^ bit[9] ^ bit[10] ^ bit[12] ^ bit[15];
}

outbyte = 0;

for (i= 0; i<8; i++)                    // convert output back to int
    outbyte += (descrambit[i] << i);

// now step the LFSR 8 serial clocks
```


412

```
    bit_out[0] = bit[8] ^ bit[9] ^ bit[10] ^ bit[12] ^ bit[15] ;
    bit_out[1] = bit[9] ^ bit[10] ^ bit[11] ^ bit[13];
    bit_out[2] = bit[10] ^ bit[11] ^ bit[12] ^ bit[14];
    bit_out[3] = bit[11] ^ bit[12] ^ bit[13] ^ bit[15];
    bit_out[4] = bit[8] ^ bit[9] ^ bit[10]^ bit[13] ^ bit[14] ^ bit[15] ;
    bit_out[5] = bit[9] ^ bit[10] ^ bit[11] ^ bit[14] ^ bit[15]  ;
    bit_out[6] = bit[10] ^ bit[11] ^ bit[12] ^ bit[15] ;
    bit_out[7] = bit[11] ^ bit[12] ^ bit[13];
    bit_out[8] = bit[0] ^ bit[12] ^ bit[13] ^ bit[14] ;
    bit_out[9] = bit[1] ^ bit[13] ^ bit[14] ^ bit[15] ;
    bit_out[10] = bit[2] ^ bit[14] ^ bit[15];
    bit_out[11] = bit[3] ^ bit[15];
    bit_out[12] = bit[4] ;
    bit_out[13] = bit[5] ^ bit[8] ^ bit[9] ^ bit[10] ^ bit[12] ^ bit[15] ;
    bit_out[14] = bit[6] ^ bit[9] ^ bit[10] ^ bit[11] ^ bit[13] ;
    bit_out[15] = bit[7] ^ bit[8] ^ bit[9] ^ bit[11] ^ bit[14] ^ bit[15] ;

    lfsr = 0;

    for (i=0; i <16; i++)           // convert the LFSR back to integer
        lfsr += (bit_out[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	5FEF	BFDE	DFAD	1F4B	3E96	7D2C	FA58
08	54A1	A942	F295	453B	8A76	B4FD	C9EB	33C7
10	678E	CF1C	3E29	7C52	F8A4	5159	A2B2	E575
18	6AFB	D5F6	0BFD	17FA	2FF4	5FE8	BFD0	DFB1
20	1F73	3EE6	7DCC	FB98	5721	AE42	FC95	593B
28	B276	C4FD	29EB	53D6	A7AC	EF49	7E83	FD06
30	5A1D	B43A	C865	30DB	61B6	C36C	26C9	4D92
38	9B24	9659	8CA3	B957	D2BF	056F	0ADE	15BC
40	2B78	56F0	ADE0	FBD1	57B3	AF66	FEDD	5DAB
48	BB56	D6BD	0D6B	1AD6	35AC	6B58	D6B0	0D71
50	1AE2	35C4	6B88	D710	0E31	1C62	38C4	7188
58	E310	6631	CC62	38D5	71AA	E354	66B9	CD72
60	3AF5	75EA	EBD4	77B9	EF72	7EF5	FDEA	5BC5

	0,8	**1,9**	**2,A**	**3,B**	**4,C**	**5,D**	**6,E**	**7,F**
68	B78A	CF05	3E1B	7C36	F86C	50C9	A192	E335
70	667B	CCF6	39FD	73FA	E7F4	6FF9	DFF2	1FF5
78	3FEA	7FD4	FFA8	5F41	BE82	DD15	1A3B	3476

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	8D	76	D2	C2	68	B3	26	1F	6C	43	08	A5	54	D3	52	34
10	02	95	55	8A	1B	BE	1A	BB	3D	B7	56	FA	1B	0B	F6	53
20	41	9C	80	F0	4A	C3	7F	74	91	52	86	BE	B6	A7	6F	7A
30	D6	E6	63	31	0C	FE	36	F0	29	EA	F3	1E	94	26	34	EB
40	17	3B	85	53	4D	DC	4A	E9	88	0E	20	5D	D0	ED	01	69
50	DE	38	9E	FA	07	4B	DB	68	7B	43	0A	45	08	D7	0D	96
60	98	E6	35	3F	A5	98	76	FE	15	F7	0E	C8	AF	90	60	66
70	CB	D5	F0	FA	9F	00	82	2B	91	74	31	0E	1E	6A	F4	76
80	48	69	6D	F4	93	8A	CD	7B	7E	AD	13	15	EE	FE	72	1E
90	3B	AA	14	A0	E7	D4	AA	23	67	9E	DC	B0	FB	73	A5	E0
A0	4F	94	CA	06	12	92	E2	63	6D	62	78	45	93	0C	26	53
B0	02	22	59	3E	63	CA	6E	2B	1F	1F	6A	63	ED	A9	B5	35
C0	FD	A0	A2	4A	96	E1	AF	71	62	7B	D5	E1	8A	56	A0	55
D0	68	89	D1	82	FF	B4	4C	23	7F	1E	48	83	7F	E8	1A	B2
E0	CD	EA	C5	A9	C3	AC	01	62	CE	39	09	F6	7D	76	5F	39
B0	41	58	A5	2F	7A	4A	6D	83	7A	58	8D	38	5C	FF	3D	77
C0	B3	9C	23	3C	A0	91	4D	56	E1	0B	ED	43	A7	29	74	98
D0	A2	DB	2D	E5	7F	C8	8D	E7	69	C6	B8	0A	C9	83	D0	64
F0	3A	F9	3D	05	EA	D9	E9	EE	97	3B	BD	44	8C	4B	E2	0F

Scrambling produces the power spectrum shown in Figure C-1.

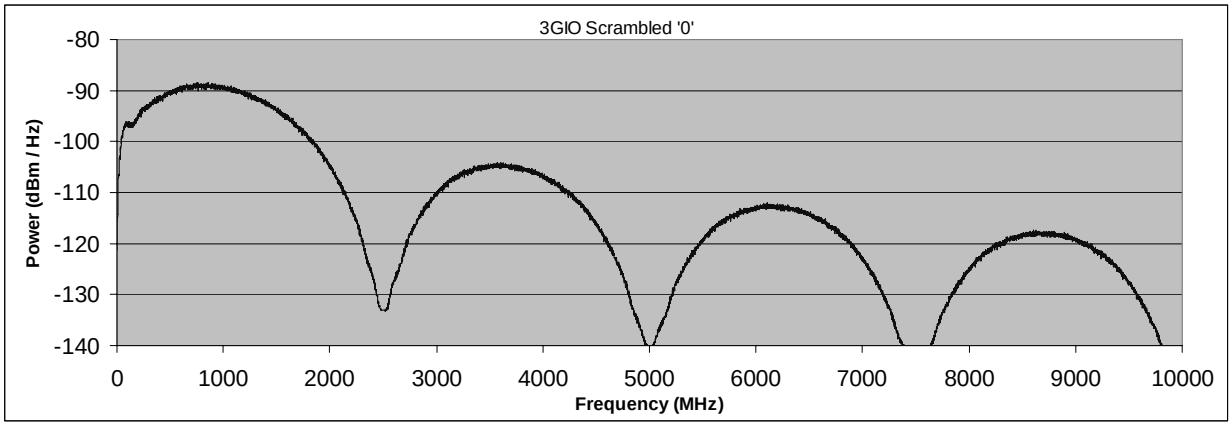

Figure C-1: Scrambling Spectrum for Data Value of 0

416